HP Virtual User Generator

for the Windows operating system

Software Version: 9.50

User Guide
Volume I - Using VuGen

Manufacturing Part Number: T7182-90019
Document Release Date: January 2009
Software Release Date: January 2009

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Third-Party Web Sites

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

Copyright Notices

Trademark Notices

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

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

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

UNIX® is a registered trademark of The Open Group.

Documentation Updates

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

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

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

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP Software Support web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software Support Online provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the HP Software Support web site to:

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

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

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Table of Contents

Welcome to This Guide 13
How This Guide Is Organized 14
Who Should Read This Guide 15
LoadRunner Online Documentation 15
Additional Online Resources 17
Documentation Updates 18

PART I: WORKING WITH VUGEN

Chapter 1: Developing Vuser Scripts 21
Introducing Vusers 22
Looking at Vuser Types 25
The Steps of Creating Vuser Scripts 27
HP LoadRunner Licensing 28
LoadRunner and Service Test 29

Chapter 2: Introducing VuGen 31
About VuGen 31
Starting VuGen 32
Customizing the Commands 32
Understanding the VuGen Environment Options 36
Setting the Environment Options 38
Viewing and Modifying Vuser Scripts 38
Running Vuser Scripts with VuGen 53
Understanding VuGen Code 53
Getting Help on Vuser Functions 56

Chapter 3: Using the Protocol Advisor 61
About the Protocol Advisor 61
Protocol Advisor Workflow 62
Notes and Limitations 65

Chapter 4: Viewing the VuGen Workflow 67
About Viewing the VuGen Workflow..67
Viewing the Task Pane 68
Recording Steps 69
Verifying the Script 70
Enhancing the Script..72
Prepare for Load 78
Finishing Your Script...79
Restoring the Workflow View 79

Chapter 5: Recording with VuGen...81
About Recording with VuGen...82
Vuser Script Sections 82
Creating New Virtual User Scripts...84
Adding and Removing Protocols 87
Choosing a Virtual User Category..88
User-Defined Template...89
Creating a New Script...91
Opening an Existing Script 92
Recording Your Application..93
Ending and Saving a Recording Session..97
Viewing the Recording Logs...98
Using Zip Files 100
Importing Actions 101
Providing Authentication Information..102
Regenerating a Vuser Script...104

Chapter 6: Enhancing Vuser Scripts...107
About Enhancing Vuser Scripts...108
Inserting Transactions into a Vuser Script......................................110
Inserting Rendezvous Points (LoadRunner only) 112
Inserting Comments into a Vuser Script...114
Obtaining Vuser Information 115
Sending Messages to Output..116
Handling Errors in Vuser Scripts During Execution 120
Synchronizing Vuser Scripts...121
Emulating User Think Time 122
Handling Command Line Arguments 123
Encrypting Text...124
Encoding Passwords Manually..125
Adding Files to the Script 126

Chapter 7: Creating Business Process Reports 127
About Exporting a Script to Word 127
Specifying the Report Details 128
Specifying Report Content 129

Chapter 8: Correlating Statements..131
About Correlating Statements..131
Using Correlation Functions for C Vusers 133
Using Correlation Functions for Java Vusers 135
Comparing Vuser Scripts using WDiff..136
Modifying Saved Parameters 138

Chapter 9: Running Vuser Scripts in Standalone Mode 139
About Running Vuser Scripts in Standalone Mode 139
Running a Vuser Script in VuGen 140
Replaying a Vuser Script...144
Using VuGen's Debugging Features 147
Using VuGen's Debugging Features for Web Vuser Scripts 152
Working with VuGen Windows..153
Find In Files 154
Running a Vuser Script from a Command Prompt...........................156
Running a Vuser Script from a UNIX Command Line 156
Integrating Scripts into Tests...159

Chapter 10: Viewing Test Results 163
About Viewing Test Results 164
The Test Results Window 165
Viewing the Results 168
Finding Results Steps..175
Printing Results..176
Exporting Test Results 179
Submitting Defects to Quality Center..180
Connecting to Quality Center from the Test Results Window 181
Customizing the Test Results Display 182
Viewing Web Services Reports 182
Sending Custom Information to the Report....................................185

Chapter 11: Managing Scripts Using Quality Center 187
About Managing Scripts Using Quality Center 187
Connecting to and Disconnecting from Quality Center 188
Opening Scripts from a Quality Center Project 192
Saving Scripts to a Quality Center Project 194
Managing Script Versions...195

Chapter 12: Managing Scripts with HP Performance Center 205
About Managing Scripts with HP Performance Center 205
Viewing the Vuser Scripts List in Performance Center.....................206
Connecting VuGen to Performance Center......................................206
Uploading Vuser Scripts 208
Downloading Vuser Scripts 211
Working with Downloaded Scripts 213

PART II: PARAMETERS

Chapter 13: Working with VuGen Parameters.................................217
About VuGen Parameters...218
Understanding Parameter Limitations...219
Creating Parameters 220
Understanding Parameter Types 223
Defining Parameter Properties 226
Using Existing Parameters..228
Using the Parameter List 230
Setting Parameterization Options 233

Chapter 14: File, Table, and XML Parameter Types 237
Selecting or Creating Data Files or Data Tables 238
Setting Properties for File Type Parameters......................................244
Setting Properties for Table Type Parameters...................................246
Choosing Data Assignment Methods for File/Table Parameters 248
Setting Properties for XML Parameters 253

Chapter 15: Setting Parameter Properties 263
About Setting Parameter Properties 263
Setting Properties for Internal Data Parameter Types.......................264
Setting Properties for User-Defined Functions.................................274
Customizing Parameter Formats..275
Selecting an Update Method 276
Simulating File Type Parameters..277
Using the File Parameter Simulator..279

PART III: RECORDING OPTIONS

Chapter 16: Recording Options for Selected Protocols 285
EJB Recording Options 286
RDP Recording Options..287
Understanding Citrix Recording Options 292
Setting the Citrix Recording Options...299
Database Recording Options 300
Database Advanced Recording Options 301
Setting the AMF Recording Mode 304
AMF Code Generation Options..308
Flex Code Generation Options 309
Microsoft .NET Recording Options 311
WCF Recording Options 318
RTE Configuration Options..320
RTE Recording Options 321
SAPGUI Recording Options..325
SAP-Web Recording Options..329
Setting the Correlation Recording Options 330
Web, Wireless, and Oracle NCA Recording Options 332

Chapter 17: Click and Script Recording 341
About Recording with Click and Script..341
Viewing Web (Click and Script) Vuser Scripts 343
Setting Click and Script Recording Options 344
Setting Advanced GUI Properties...345
Configuring Web Event Recording 348

Chapter 18: Java Recording Options..361
About Setting Java Recording Options..362
Java Virtual Machine (JVM) Recording Options..............................363
Setting Classpath Recording Options...365
Recorder Options 366
Serialization Options...368
Correlation Options 370
Log Options 371
CORBA Options 373

Chapter 19: Setting Script Generation Preferences 375
About Setting Script Generation Preferences...................................376
Selecting a Script Language 376
Applying the Basic Options..377
Understanding the Correlation Options..379
Setting Script Recording Options 380

Chapter 20: Setting Recording Options for Web Vusers 381
About Setting Recording Options 381
Understanding the Recording Levels 382
Setting the Recording Level...395

Chapter 21: Configuring the Port Mappings 397
About Configuring the Port Mappings 398
Defining Port Mappings...398
Adding a New Server Entry 401
Setting the Advanced Port Mapping Options..................................403
Setting the Port Mapping Recording Options..................................406

PART IV: RUN-TIME SETTINGS

Chapter 22: Configuring Run-Time Settings 413
About Run-Time Settings 414
Configuring Run Logic Run-Time Settings (multi-action) 415
Pacing Run-Time Settings..420
Configuring Pacing Run-Time Settings (multi-action) 421
Setting Pacing and Run Logic Options (single action) 422
Configuring the Log Run-Time Settings 424
Configuring the Think Time Settings 429
Configuring Additional Attributes Run-Time Settings 431
Configuring Miscellaneous Run-Time Settings................................432
Setting the VB Run-Time Settings 438

Chapter 23: Configuring Network Run-Time Settings 439
About Network Run-Time Settings 440
Setting the Network Speed 440
Setting Proxy Options 442
Setting Browser Emulation Properties...447
Setting Internet Preferences 452
Filtering Web Sites...461
Obtaining Debug Information 462
Performing HTML Compression 463
Checking Web Page Content 464

Chapter 24: Run-Time Settings for Selected Protocols 469
RDP Run-Time Settings 470
Citrix Run-Time Settings 475
.NET Environment Run-Time Settings 479
Oracle NCA Run-Time Settings 482
SAPGUI Run-Time Settings 485
Java and EJB Run-Time Settings 489
RTE Run-Time Settings 492
WAP Run-Time Settings 494
MMS Run-Time Settings 500
Flex Run-Time Settings 502

PART V: INFORMATION FOR ADVANCED USERS

Chapter 25: Creating Vuser Scripts in Visual Studio 505
About Creating Vuser Scripts in Visual Studio 505
Creating a Vuser Script with Visual C 506
Creating a Vuser Script with Visual Basic 508
Configuring Runtime Settings and Parameters 510

Chapter 26: Programming with the XML API 511
About Programming with the XML API 512
Understanding XML Documents 513
Using XML Functions 514
Specifying XML Function Parameters 517
Working with XML Attributes 519
Structuring an XML Script 519
Enhancing a Recorded Session 521
Using Result Parameters 526

Chapter 27: VuGen Debugging Tips 529
General Debugging Tip 530
Using C Functions for Tracing 530
Adding Additional C Language Keywords 530
Examining Replay Output 531
Debugging Database Applications 531
Working with Oracle Applications 533
Solving Common Problems with Oracle 2-Tier Vusers 534
Two-tier Database Scripting Tips 539
Running PeopleSoft-Tuxedo Scripts 548

Chapter 28: Advanced Topics 549
Files Generated During Recording 549
Files Generated During Replay...552
Running a Vuser from the Unix Command Line 553
Specifying the Vuser Behavior 555
Command Line Parameters..556
Recording OLE Servers..556
Examining the .dat Files..559
Adding a New Vuser Type 560

PART VI: APPENDIXES

Appendix A: Calling External Functions..569
Loading a DLL Locally..569
Loading a DLL Globally 571

Appendix B: Working with Foreign Languages...............................573
About Working with Foreign Languages 573
Manually Converting String Encoding 574
Converting String Encoding In Parameter Files...............................575
Setting the String Encoding for Web Record and Replay 577
Specifying a Language for the Accept-Language Header 580
Protocol Limitations...581
Quality Center Integration..582

Appendix C: Programming Scripts on UNIX Platforms....................583
About Programming Vuser Scripts to Run on UNIX Platforms........583
Generating Templates 584
Programming Vuser Actions 585
Configuring Vuser Run-Time Settings 587
Defining Transactions and Rendezvous Points...............................592
Compiling Scripts...592

Appendix D: Using Keyboard Shortcuts 595

Index...597

Welcome to This Guide

Welcome to the HP Virtual User Generator, *VuGen*, HP's tools for creating Vuser scripts. You use VuGen to develop a Vuser script by recording a user performing typical business processes. The scripts let you emulate real-life situations.

You use the scripts created with VuGen in conjunction with other products— HP LoadRunner, HP Performance Center, and HP Business Availability Center.

HP LoadRunner, a tool for performance testing, stresses your entire application to isolate and identify potential client, network, and server bottlenecks.

HP Performance Center implements the capabilities of LoadRunner on an enterprise level.

HP Business Availability Center helps you optimize the management and availability of business applications and systems in production.

This chapter includes:

- ➤ How This Guide Is Organized on page 14
- ➤ Who Should Read This Guide on page 15
- ➤ LoadRunner Online Documentation on page 15
- ➤ Additional Online Resources on page 17
- ➤ Documentation Updates on page 18

How This Guide Is Organized

This guide contains the following parts:

Part I Working with VuGen

Describes the Virtual User Generator interface and the recording and replaying of scripts. It also describes standard run-time settings, using data parameters and customizing a script.

Part II Parameters

Describes the process of creating parameters. Provides information about parameter types and properties.

Part III Recording Options

Provides information about the recording options relevant to all protocols as well as selected individual protocols.

Part IV Run-Time Settings

Provides information about the run-time settings relevant to all protocols as well as selected individual protocols.

Part V Information for Advanced Users

Provides information for advanced users such as general debugging tips, the files generated by VuGen, and how to program scripts in Visual C and Visual Basic.

Part VI Appendixes

Contains technology overviews and information about other advanced topics. Learn about calling external functions, programming in UNIX, working with foreign languages, and keyboard shortcuts.

Note: The online version of the *Virtual User Generator* guide is a single volume, while the printed version consists of two volumes, Volume I - *Using VuGen* and Volume II - *Protocols*.

Who Should Read This Guide

This guide is for the following users:

- Script developers
- Functional Testers
- Load Testers

This document assumes that you are moderately knowledgeable about your enterprise application.

LoadRunner Online Documentation

LoadRunner includes a complete set of documentation describing how to use the product. The documentation is available from the help menu and in PDF format. PDFs can be read and printed using Adobe Reader, which can be downloaded from the Adobe Web site *(*http://www.adobe.com*)*. Printed documentation is also available on demand.

Accessing the Documentation

You can access the documentation as follows:

- From the **Start** menu, click **Start** > **LoadRunner** > **Documentation** and select the relevant document.
- From the **Help** menu, click **Documentation Library** to open the merged help.

Getting Started Documentation

- **Readme.** Provides last-minute news and information about LoadRunner. You access the Readme from the **Start** menu.
- **HP LoadRunner Quick Start** provides a short, step-by-step overview and introduction to using LoadRunner. To access the Quick Start from the Start menu, click **Start** > **LoadRunner** > **Quick Start.**
- **HP LoadRunner Tutorial.** Self-paced printable guide, designed to lead you through the process of load testing and familiarize you with the LoadRunner testing environment. To access the tutorial from the Start menu, click **Start** > **LoadRunner** > **Tutorial.**

LoadRunner Guides

- **HP Virtual User Generator User Guide.** Describes how to create scripts using VuGen. The printed version consists of two volumes, Volume I - *Using VuGen* and Volume II - *Protocols*, while the online version is a single volume. When necessary, supplement this user guide with the online *HP LoadRunner Online Function Reference*.
- **HP LoadRunner Controller User Guide.** Describes how to create and run LoadRunner scenarios using the LoadRunner Controller in a Windows environment.
- **HP LoadRunner Monitor Reference.** Describes how to set up the server monitor environment and configure LoadRunner monitors for monitoring data generated during a scenario.
- **HP LoadRunner Analysis User Guide.** Describes how to use the LoadRunner Analysis graphs and reports after running a scenario to analyze system performance.
- **HP LoadRunner Installation Guide.** Explains how to install LoadRunner and additional LoadRunner components, including LoadRunner samples.

LoadRunner References

- **LoadRunner Function Reference.** Gives you online access to all of LoadRunner's functions that you can use when creating Vuser scripts, including examples of how to use the functions.

- **Analysis API Reference.** This Analysis API set can be used for unattended creating of an Analysis session or for custom extraction of data from the results of a test run under the Controller. You can access this reference from the Analysis Help menu.
- **LoadRunner Controller Automation COM and Monitor Automation Reference.** An interface with which you can write programs to run the LoadRunner Controller and perform most of the actions available in the Controller user interface. You access this reference (**automation.chm)** from the **<LoadRunner Installation>/bin** directory.
- **Error Codes and Troubleshooting.** Provides clear explanations and troubleshooting tips for Controller connectivity and Web protocol errors. It also provides general troubleshooting tips for Winsock, SAPGUI, and Citrix protocols.

Additional Online Resources

For additional help, refer to one of the following resources:

Troubleshooting & Knowledge Base accesses the Troubleshooting page on the HP Software Support Web site where you can search the Self-solve knowledge base. Choose **Help** > **Troubleshooting & Knowledge Base**. The URL for this Web site is http://h20230.www2.hp.com/troubleshooting.jsp.

HP Software Support accesses the HP Software Support Web site. This site enables you to browse the Self-solve knowledge base. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. Choose **Help** > **HP Software Support**. The URL for this Web site is www.hp.com/go/hpsoftwaresupport.

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

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

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

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

Documentation Updates

HP Software is continually updating its product documentation with new information.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to the HP Software Product Manuals Web site (http://h20230.www2.hp.com/selfsolve/manuals).

Part 1

Working with VuGen

1

Developing Vuser Scripts

When testing or monitoring an environment, you need to emulate the true behavior of users on your system. HP testing tools emulate an environment in which users concurrently work on, or access your system.

To do this emulation, the human was replaced with a virtual user, or a *Vuser*. The actions that a Vuser performs are described in a *Vuser script*. The primary tool for creating Vuser scripts is the Virtual User Generator, *VuGen*.

This chapter includes:

➤ Introducing Vusers on page 22

➤ Looking at Vuser Types on page 25

➤ The Steps of Creating Vuser Scripts on page 27

➤ HP LoadRunner Licensing on page 28

➤ LoadRunner and Service Test on page 29

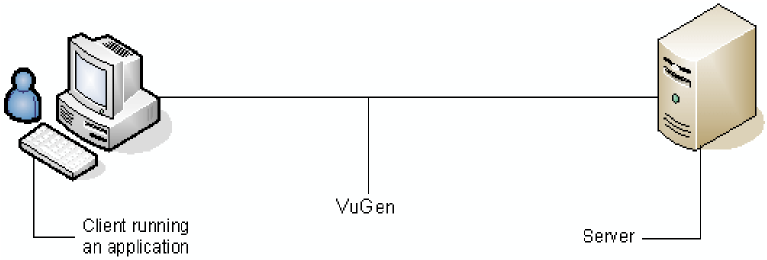
Note: The online version of the *Virtual User Generator* guide is a single volume, while the printed version consists of two volumes, *Volume I-Using VuGen* and *Volume II-Protocols*.

Introducing Vusers

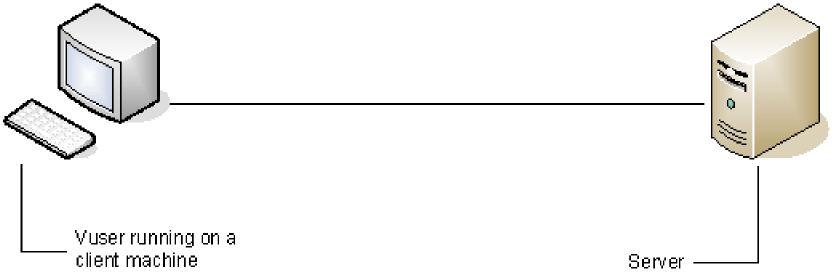
Vusers emulate the actions of human users by performing typical business processes in your application. The actions that a Vuser performs during the recording session are described in a *Vuser script*.

HP's tool for creating Vuser scripts is the Virtual User Generator, *VuGen*. You use VuGen to develop a Vuser script by recording a user performing typical business processes on a client application. VuGen records the actions that you perform during the recording session, recording only the activity between the client and the server. Instead of having to manually program the application's API function calls to the server, VuGen automatically generates functions that accurately model and emulate real world situations.

During recording VuGen monitors the client end of the database and traces all the requests sent by the user and received from the user, to the server.



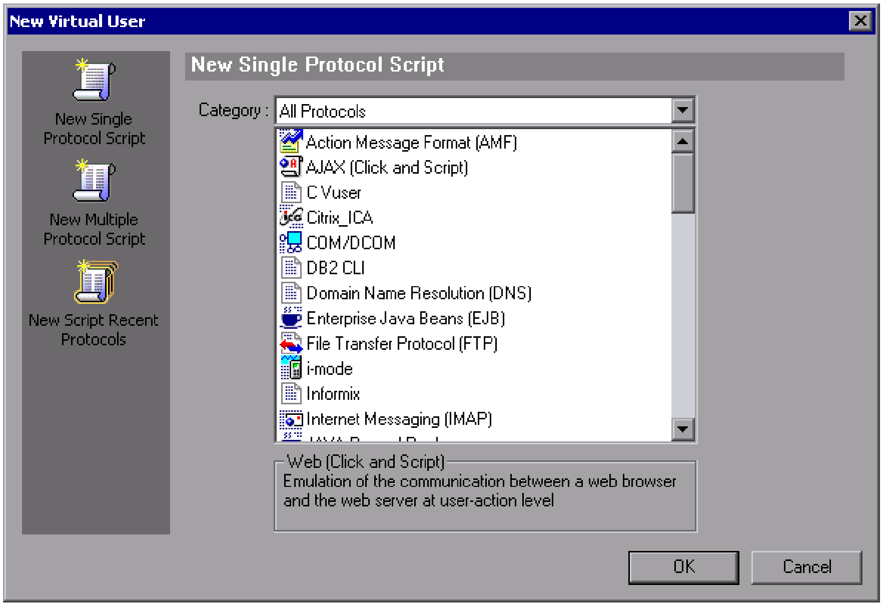
During playback, Vuser scripts communicate directly with the server by executing calls to the server API. When a Vuser communicates directly with a server, system resources are not required for the client interface. This lets you run a large number of Vusers simultaneously on a single workstation, and enables you to use only a few testing machines to emulate large server loads.



In addition, since Vuser scripts do not rely on client software, you can use Vusers to check server performance even before the user interface of the client software has been fully developed.

Using VuGen, you can run scripts as standalone tests. Running scripts from VuGen is useful for debugging as it enables you to see how a Vuser will behave and which enhancements need to be made.

VuGen enables you to record a variety of Vuser types, each suited to a particular load testing environment or topology. When you open a new test, VuGen displays a complete list of the supported protocols.



New Virtual User

New Single Protocol Script

Category : All Protocols

Action Message Format (AMF)
AJAX (Click and Script)
C Vuser
Citrix_ICA
COM/DCOM
DB2 CLI
Domain Name Resolution (DNS)
Enterprise Java Beans (EJB)
File Transfer Protocol (FTP)
i-mode
Informix
Internet Messaging (IMAP)

Web (Click and Script)
Emulation of the communication between a web browser and the web server at user-action level

New Single Protocol Script
New Multiple Protocol Script
New Script Recent Protocols
OK
Cancel

While running the Vusers, you gather information about the system's response. Afterwards, you can view this information with the Analysis tool. For example, you can observe how a server behaved when one hundred Vusers simultaneously withdrew cash from a bank's ATM.

Looking at Vuser Types

VuGen provides a variety of Vuser technologies that allow you to emulate your system. Each technology is suited to a particular architecture and results in a specific type of Vuser script. For example, you use Web Vuser Scripts to emulate users operating Web browsers. You use FTP Vusers to emulate an FTP session. The various Vuser technologies can be used alone or together, to create effective load tests or Business Process Monitor profiles.

The Vuser types are divided into the following categories:

- **All Protocols.** A list of all supported protocols in alphabetical order
- **Application Deployment Solution.** For the Citrix and Microsoft Remote Desktop Protocol (RDP) protocols
- **Client/Server.** For DB2 CLI, DNS, Informix, Microsoft .NET, MS SQL, ODBC, Oracle (2-tier), Sybase Ctlib, Sybase Dblib, and Windows Sockets protocols
- **Custom.** For C template, Java template, Javascript, VB script, VBNet, and VB template type scripts
- **Distributed Components.** For COM/DCOM, and Microsoft .NET protocols
- **E-business.** For AJAX (Click and Script), AMF, Flex, FTP, LDAP, Microsoft .NET, Web (Click and Script), Web (HTTP/HTML), and Web Services protocols
- **ERP/CRM.** For Oracle NCA, Oracle Web Applications 11i, Peoplesoft Enterprise, Peoplesoft-Tuxedo, SAP-Web, SAPGUI, SAP (Click and Script), and Siebel Web protocols
- **Enterprise Java Beans.** For EJB Testing
- **Java.** For the Java Record/Replay protocol
- **Legacy.** For Terminal Emulation (RTE)
- **Mailing Services.** Internet Messaging (IMAP), MS Exchange (MAPI), Post Office Protocol (POP3), and Simple Mail Protocol (SMTP)
- **Middleware.** For the Tuxedo protocol
- **Streaming.** For MediaPlayer (MMS) and RealPlayer protocols

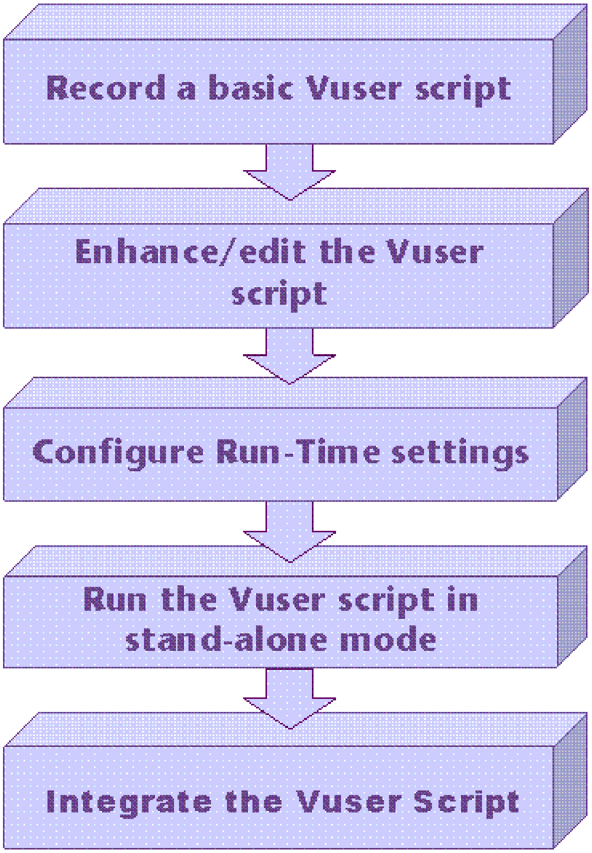
- **Wireless.** For Multimedia Messaging Service (MM) and WAP protocols

To view a list of all supported protocols in alphabetical order, select **File** > **New** and select *All Protocols* in the **Category** list box.

Note: In order to run the various protocols, you must have either a global license or licenses for the desired protocols. For more information, select **Configuration** > **LoadRunner License** in the LoadRunner Launcher (**Start** > **Programs** > **LoadRunner** > **LoadRunner**).

The Steps of Creating Vuser Scripts

The following diagram outlines the process of developing a Vuser script:


Record a basic Vuser script
Enhance/edit the Vuser script
Configure Run-Time settings
Run the Vuser script in stand-alone mode
Integrate the Vuser Script

The process of creating a Vuser script is as follows:

1. Record a basic script using VuGen. If you are testing Windows-based GUI applications or complex Web environments such as applets and Flash, you may need to use HP's GUI-based tools such as WinRunner and QuickTest Professional.
2. Enhance the basic script by adding control-flow statements and other LoadRunner API functions into the script.
3. Configure the run-time settings. These settings include iteration, log, and timing information, and define the Vuser behavior during a script run.

4 Verify the script’s functionality, run it in standalone mode.

5 After you verify that the script is functional, you integrate it into your environment: a LoadRunner scenario, Performance Center load test, or Business Process Monitor profile. For more information, see the *HP LoadRunner Controller*, *HP Performance Center*, or *HP Business Availibility Center User Guides*.

HP LoadRunner Licensing

When you purchase LoadRunner, you receive a custom license for your configuration and needs. You can access this information at any time through the LoadRunner Launcher. To preview your license key information, choose **Start** > **Programs** > **LoadRunner** to open the LoadRunner Launcher. Select **LoadRunner License** from the Launcher’s **Configuration** menu.

When you purchase a license for Service Test, you install it through the License Manager (**Start** > **Programs** > **LoadRunner** > **Service Test** > **License Manager**). The license can be a Seat license (host lock) or a Concurrent license (site license). For more information, see the *HP LoadRunner Installation Guide*.

HP Service Test is available as a standalone product or as an extension to HP LoadRunner's Virtual User Generator, VuGen. Service Test contains all of the capabilities of VuGen with added functionality in the area of SOA and Web Service testing.

HP Service Test is a functional testing tool and therefore provides more functional testing features than VuGen. In addition, Service Test contains several utilities that allow working with WSDLs, XML, and SOAP.

All scripts created in HP Service Test can run in HP LoadRunner and HP Performance Center. This compatibility exists only when Service Test and LoadRunner are of the same version.

LoadRunner and Service Test

The following features exist only in HP Service Test (standalone) or LoadRunner with Service Test:

Functional Testing Utilities

- **XML Validation.** checks if XML is well-formed, complies with a schema, and uses expected values.
- **Checkpoints.** allows you to compare the response with expected values.
- **Negative Testing.** the ability to define SOAP faults as the desired response, enabling you to test error cases.
- **WS-I Validation.** the ability to check if the WSDL and SOAP are WS-I compliant
- **Test Generation Wizard.** a wizard guiding you in creating aspect-based tests

Integration with Quality Center

- **BPT.** Business Process Testing. Service Test can create components for Quality Center's BPT and run them in a business scenario.
- **Remote execution from QC.** Service Test scripts can be launched directly from Quality Center. You can define the parameters and runtime settings, run the test, and inspect the results from within Quality Center.

Other Tools

- **Service Emulation.** a tool to create an emulated service to simulate both a client and server
- **Command line invocation.** the ability to invoke a script from the command line

2

Introducing VuGen

The Virtual User Generator, also known as VuGen, helps you develop Vuser scripts for a variety of application types and communication protocols.

This chapter includes:

- About VuGen on page 31
- Starting VuGen on page 32
- Customizing the Commands on page 32
- Understanding the VuGen Environment Options on page 36
- Setting the Environment Options on page 38
- Viewing and Modifying Vuser Scripts on page 38
- Running Vuser Scripts with VuGen on page 53
- Understanding VuGen Code on page 53
- Getting Help on Vuser Functions on page 56

About VuGen

The Virtual User Generator, also known as VuGen, is the primary tool for developing Vuser scripts.

VuGen not only records Vuser scripts, but also runs them. Running scripts from VuGen is useful for debugging. It enables you to emulate how a Vuser script will run when executed as part of a larger test.

When you record a Vuser script, VuGen generates various functions that define the actions that you perform during the recording session. VuGen inserts these functions into the VuGen editor to create a basic Vuser script.

VuGen records Vuser scripts on Windows platforms only. However, a recorded Vuser script can be run on both Windows and UNIX platforms.

Starting VuGen

VuGen can be run from a number of different applications such as HP Business Availability Center, HP LoadRunner, and HP Performance Center.

To start VuGen, select **Start** > **Programs** > *<App_Name>* > **Applications** > **Virtual User Generator**. The Virtual User Generator Start Page opens.

The following actions are available from the start page:

Action	Procedure
To open an existing script	Click **Open Existing Script**
To open an existing script from a zip archive	Select **File** > **Zip Operations** > **Import From Zip File**
To create a new script	Click **New Vuser Script**

Customizing the Commands

You can customize the appearance of VuGen by configuring toolbars and shortcuts from the Customize dialog box using one of the following tabs:

- Commands Tab
- Toolbars Tab
- Tools Tab
- Keyboard Tab
- Options Tab

Commands Tab

Select a toolbar in the **Categories** section (left pane). The toolbar's buttons are displayed in the right pane, **Commands**. Click on a command to view its description. Drag the desired items from the right pane into the desired open toolbar.

Toolbars Tab

The **Toolbars** tab lets you indicate which toolbars to display. Select the check box for each toolbar you want to display. The available toolbars are Standard, Edit, View, Record, Vuser, Debug, Tools, Tree, and Advanced.

Additionally, the following buttons in this tab allow you to configure the toolbar settings:

- **Reset.** Returns to the default toolbar view settings for toolbar selected in the left pane.
- **Reset All.** Returns to the default toolbar view settings for all toolbars. VuGen issues a warning before reverting back to the default settings.
- **New.** Adds a custom toolbar to the list of shown toolbars. Drag this toolbar to the desired location in the VuGen window. Afterwards, use the Commands tab to add icons to the toolbar.
- **Rename.** Allows you to rename the selected toolbar. This button is only enabled when you select a custom toolbar.
- **Delete.** Deletes the selected toolbar. This button is only enabled when you select a custom toolbar.
- **Show text labels.** Shows text labels for the selected toolbar. For example, if you enable this option for the Record toolbar, the Stop button will display the word Stop and the Run button will display Run.

Tools Tab

The **Tools** tab lets you add custom commands to VuGen toolbars. The command is usually an executable, batch, or *.com* file. In addition you can indicate a specific document or PDF file that you may want to open from within VuGen. You define one or more tools in the Tools tab Tool list. VuGen lists these tools under the Commands tab **Tools** menu. You select the desired command from the Tool menu, and drag it onto the desired toolbar. For example, you can specify *calc.exe*, the Windows calculator, to allow you to open it from VuGen.

Menu Contents

- **New.** (left button) Add an item to the Menu Contents list. VuGen creates a new item in the Menu contents list and the cursor moves to the beginning of the line. Type in the name of the command.
- **Delete.** (second button) Deletes the selected item from the Menu Contents list.

Up Arrow / Down Arrow. Move the selected item up or down.

Item Details

- **Command.** Type in the name of an executable or batch file. Alternatively, click the Browse button to the right of the field, and locate the desired file. If you specify a non-executable file, make sure that the file extension is associated with a program that is installed on the machine.
- **Arguments.** Specify run-time arguments to be executed with the specified file.
- **Initial Directory.** The initial directory in which to run the custom tool.

Keyboard Tab

The Keyboard tab lets you assign keyboard shortcuts to menu commands. You can assign shortcuts to both standard and custom commands.

To assign a keyboard shortcut:

1. **Select a menu category from the Category box.**
2. Select a command within the chosen category from the **Commands** box.

3 Click in the **Press New Shortcut Key** box and press the shortcut key or key combination. VuGen displays the key or key combination in the box.

4 Click **Assign**.

5 To remove an assignment, select the command and view its assigned shortcuts. Select the shortcut you want to remove in the **Current Keys** box, and click **Remove**.

6 To restore all default assignments, click **Reset All**. Note that this also deletes all custom keyboard assignments.

Options Tab

Toolbar

You can set several display options that apply to all of the toolbars:

- ➤ **Show ScreenTips on toolbars:** Show the screen tips when moving the cursor over the toolbar buttons (enabled by default).
  - ➤ **Show Shortcut Keys in Screen Tips:** Show the keyboard shortcut in the tooltips (disabled by default).
- ➤ **Large Icons:** Display large buttons on the toolbar (disabled by default).

Personalized Menu and Toolbars

You can personalize the way commands are shown in the menus and toolbars:

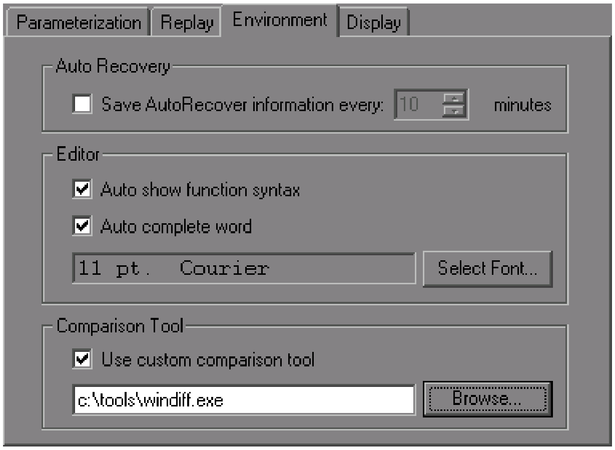
- ➤ **Menus show recently used commands first:** The menus show the commands used most recently, at the top of the menu (enabled by default).
  - ➤ **Show full menu after a short delay:** For expandable menus, show the full menu after a short delay.

Restore Defaults

➤ **Reset my usage data.** Delete all of the custom commands and restore the default set of visible commands to the menus and toolbars. Does not undo any explicit customization.

Understanding the VuGen Environment Options

You can set up your VuGen working environment in order to customize the auto recovery and editor settings. You set these options from the Tools > General Options > Environment tab.



□ Save AutoRecover information every: 10 minutes

☑ Auto show function syntax

☑ Auto complete word

11 pt. Courier

☑ Use custom comparison tool

c:\tools\windiff.exe

Auto Recovery

The auto recovery options allow you to restore your script's settings in the event of a crash or power outage. To allow auto recovery, select the **Save AutoRecover Information** check box and specify the time between the saves in minutes.

Editor

You can set the editor options to select a font and enable VuGen's Intellisense capabilities which automatically fill in words and function syntax.

- **Auto show function syntax.** When you type the opening parenthesis of a function, VuGen shows the syntax of the function with its arguments and prototypes. To enable this function globally, select the check box. If you do not check this box, you can still enable this function manually by pressing Ctrl+Shift+Space or choosing **Edit** > **Show Function Syntax** after typing the opening parenthesis in the editor.
- **Auto complete word.** When you type the first underscore of a function, VuGen opens a list of functions allowing you to select the exact function without having to manually type in the entire function. To enable this function globally, select the check box. If you do not check this box, you can enable this function manually by pressing Ctrl+Space or choosing **Edit** > **Complete Word** while typing in the editor.
- **Select Font.** To set the editor font, click **Select Font**. The Font Configuration dialog box opens. Select the desired font, style, and size and click **OK**. Note that only fixed size fonts (Courier, Lucida Console, FixedSys, etc.) are available.

Default Environment Settings

Show Function Syntax and **Auto complete word** are enabled globally by default. Auto Recovery is set to 10 seconds. If you manually changed these settings, you can restore the default values by clicking **Use Defaults**.

Comparison Tool

Vuser scripts can be compared and displayed side by side using the comparison tool.

To compare Vuser scripts:

1. Open the first Vuser script that you want to compare.
2. Select **Tools** > **Compare with Script.**
3. Select the second Vuser script. The Vuser scripts are displayed in a new window, side by side. Differences are highlighted in yellow.

Setting the Environment Options

To set the environment-related options:

1. Select **Tools** > **General Options** and click the Environment tab.
2. To save the current script information for auto recovery, select the **Save AutoRecover Information** option and specify the time in minutes between the saves.
3. To set the editor font, click **Select Font**. The Font dialog box opens. Select the desired font, style, and size and click **OK**. Note that only fixed size fonts (Courier, Lucida Console, FixedSys, and so on) are available.
4. To use a comparison tool other than WDiff, select **Use custom comparison tool** and then specify or browse for the desired tool. Make sure that the executable file that you specify is a valid comparison tool and that the path is correct. An executable file which is not a comparison tool will lead to unexpected results.
5. Click **OK** to accept the settings and close the General Options dialog box.

Viewing and Modifying Vuser Scripts

VuGen provides several views for examining the contents of your script: a text-based Script view, an icon-based Tree view with snapshots, or a icon-based Thumbnail view.

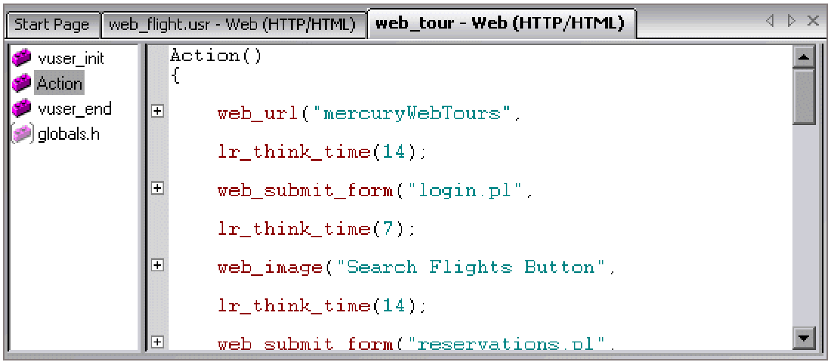
The Script and Tree views are available for most Vuser types. Many protocols also support the Thumbnail view.

Viewing the Code in Script View

The Script view lets you view the actual API functions that were recorded or inserted into the script. This view is for advanced users who want to edit the script by adding "C" or Vuser API functions as well as control flow statements.

To Display the Script View:

From the VuGen main menu, select **View** > **Script View**, or click the **Script** toolbar button. The script is displayed in the text-based Script view. If you are already in the Script view, the menu item is disabled.



```
Action()
{
	web_url("mercuryWebTours",
	lr_think_time(14);
	web_submit_form("login.pl",
	lr_think_time(7);
	web_image("Search Flights Button",
	lr_think_time(14);
	web_submit_form("reservations.pl",
```

⊞ ⊟ You can expand and collapse the functions by clicking the minus or plus sign in the margin to the left of the script. This make the script neater and easier to read.

When working in Script view, you can add steps to the script using the **Insert** > **New Step** command. Alternatively, you can manually enter functions using the Complete Word and Show Function Syntax features. For more information, see "Getting Help on Vuser Functions" on page 56.

Note: If you make changes to a Vuser script while in the Script view, VuGen makes the corresponding changes in the Tree view of the Vuser script. If VuGen is unable to interpret the text-based changes that were made, it will not convert the Script view into Tree or Thumbnail view.

Viewing a Script in Tree View

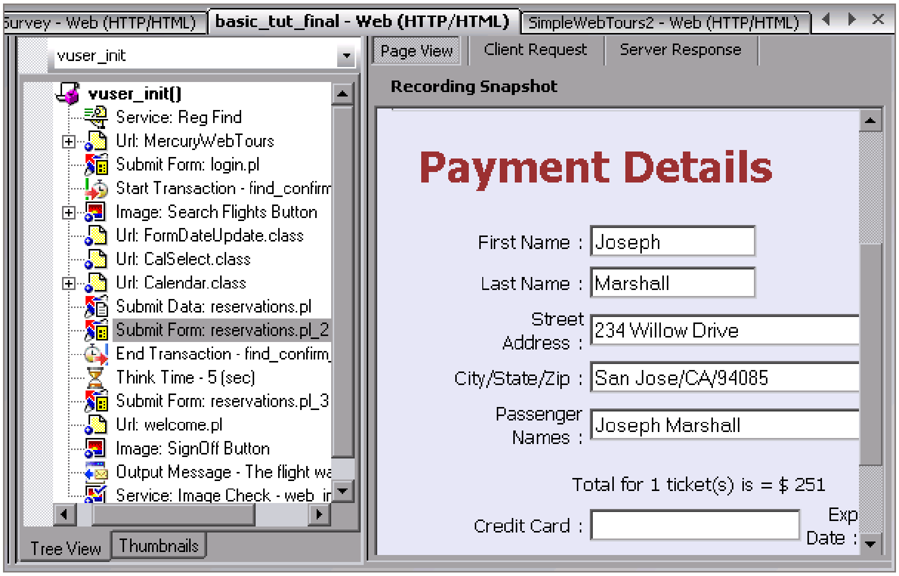
VuGen's Tree view shows the Vuser script in an icon-based format, with each step represented by a different icon.

To display the Tree view:

From VuGen's main menu, select **View** > **Tree View**, or click the **View Tree** button.

The Actions section of the Vuser script is displayed in the icon-based Tree view. To display a different section, select that section in the drop-down list, above the tree.

If you are already in Tree view, the menu item is disabled.



vuser_init()
- Service: Reg Find
- Url: MercuryWebTours
- Submit Form: login.pl
- Start Transaction - find_confirm
- Image: Search Flights Button
- Url: FormDateUpdate.class
- Url: CalSelect.class
- Url: Calendar.class
- Submit Data: reservations.pl
- Submit Form: reservations.pl_2
- End Transaction - find_confirm
- Think Time - 5 (sec)
- Submit Form: reservations.pl_3
- Url: welcome.pl
- Image: SignOff Button
- Output Message - The flight wa
- Service: Image Check - web_ir

Payment Details

First Name : Joseph
Last Name : Marshall
Street Address : 234 Willow Drive
City/State/Zip : San Jose/CA/94085
Passenger Names : Joseph Marshall
Total for 1 ticket(s) is = $ 251
Credit Card :
Exp Date :

Within the Tree view, you can manipulate steps by dragging them to the desired location. You can also add additional steps between existing steps in the tree hierarchy.

To insert a step in Tree view:

1 Click on a step.

2 Select **Insert Before** or **Insert After** from the right-click menu. The Add Step dialog box opens.

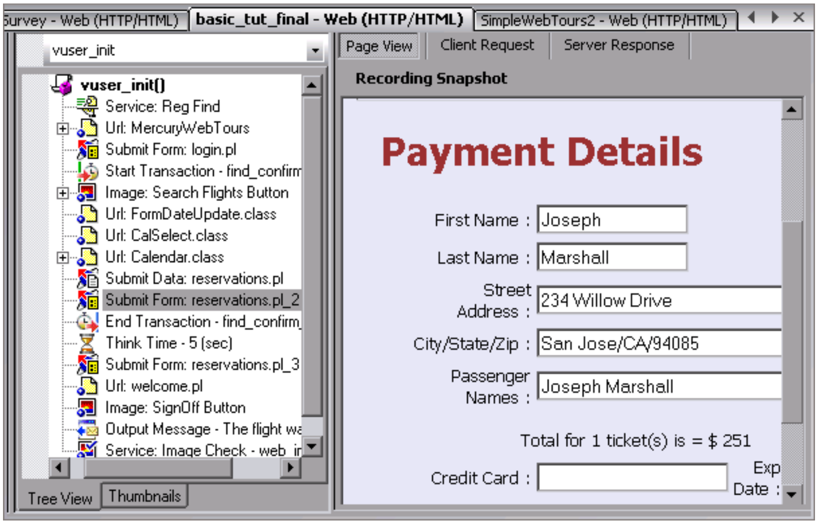
3 Select a step and click **OK**. The Properties dialog box opens.

4 Specify the properties and click **OK**. VuGen inserts the step before or after the current step.

Note: Inserting a step after a parent node results in the detachment of the children nodes from the parent node.

Understanding Snapshots

A snapshot is a graphical representation of the current step. When working in Tree view, VuGen displays the snapshot of the selected step in the right pane. The snapshot shows the client window after the step was executed.



vuser_init()
Service: Reg Find
Url: MercuryWebTours
Submit Form: login.pl
Start Transaction - find_confirm
Image: Search Flights Button
Url: FormDateUpdate.class
Url: CalSelect.class
Url: Calendar.class
Submit Data: reservations.pl
Submit Form: reservations.pl_2
End Transaction - find_confirm
Think Time - 5 (sec)
Submit Form: reservations.pl_3
Url: welcome.pl
Image: SignOff Button
Output Message - The flight wa
Service: Image Check - web_ir

Payment Details

First Name : Joseph
Last Name : Marshall
Street Address : 234 Willow Drive
City/State/Zip : San Jose/CA/94085
Passenger Names : Joseph Marshall
Total for 1 ticket(s) is = $ 251
Credit Card :
Exp Date :

VuGen captures a base snapshot during recording and another one during replay. You compare the Record and Replay snapshots to determine the dynamic values that need to be correlated in order to run the script. For more information, see the section on correlation after recording.

The following toolbar buttons let you show or hide the various snapshot windows.

Show a full window of the recorded snapshot

Show a split window of the recorded and replayed snapshot

Show a full window of the replayed snapshot

To view or hide snapshots:

1. Make sure you are in Tree view. If not, then switch to Tree view (**View** > **Tree View**).
2. Select **View** > **Snapshot** > **View Snapshot**. VuGen shows the snapshot of the client window. If the snapshot is already visible, VuGen hides it.
3. Use the expanded menu of **View** > **Snapshot** to view the recorded and/or replayed snapshots. You can also use the shortcut toolbar buttons to display the desired view:

Each time you replay the script, VuGen saves another Replay snapshot to the script's result directory: **Iteration1**, **Iteration2**, and so forth.

By default, VuGen compares the recording snapshot to the first replay snapshot. You may, however, select a different snapshot for comparison. To select a specific replay snapshot, select the expanded menu of **View** > **Snapshot** > **Select Iteration**. Select a set of results and click **OK.**

Multiple Snapshots

In several protocols such as Microsoft Remote Desktop Protocol (RDP), you can view multiple snapshots for a single step. This occurs when a mismatch occurs during replay and you choose to append the new image to the step. For more information, see the RDP protocol section.

Troubleshooting Snapshots

If you encounter a step without a snapshot, follow these guidelines to determine why it is not available. Note that not all steps are associated with snapshots—only steps with screen operations or for Web, showing browser window content, have snapshots.

Several protocols allow you to disable the capturing of snapshots during recording using the Recording options.

If there is no **Record** snapshot displayed for the selected step, it may be due to one of the following reasons:

- The script was recorded with a VuGen version 6.02 or earlier.
- Snapshots are not generated for certain types of steps.
- The imported actions do not contain snapshots.

If there is no **Replay** snapshot displayed for the selected step, it may be due to one of the following reasons:

- The script was recorded with VuGen version 6.02 or earlier.
- The imported actions do not contain snapshots.
- The Vuser files are stored in a read-only directory, and VuGen could not save the replay snapshots.
- The step represents navigation to a resource.

Snapshot Files

Each time you replay the script, VuGen saves the snapshots in the *script* directory with an *.inf* extension. The replay snapshots are located in the script's result directory: **Iteration1**, **Iteration2**, and so forth, for each set of results.

To determine the name of the snapshot file for a Web Vuser, switch to Script view (**View** > **Script View**). In the following example, the snapshot information is represented by t1.inf.

```
web_url("WebTours",
        "URL=http://localhost/WebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);
```

For Citrix Vuser scripts, VuGen saves snapshots as .png files in the script's directory. To determine the name of the snapshot file, check the function's arguments in Script view.

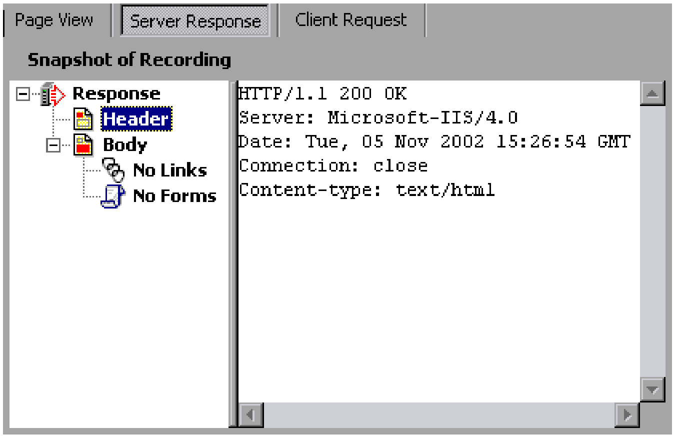
```
ctrx_sync_on_window("ICA Administrator Toolbar", ACTIVATE, 768, 0, 33,
              573, "snapshot12", CTRX_LAST);
```

Web Vuser Snapshot Tabs

In the Snapshot window for Web Vusers, the following tabs are available:

- **Page View.** Display the snapshot in HTML as it would appear in a browser. This button is available for both the recording and replay snapshots. Use this view to make sure you are viewing the correct snapshot. In this view, however, you do not see the values that need to be correlated.
- **Server Response.** Displays the server response HTML code of the snapshot. This button is available for both the recorded and replayed snapshots. The HTML view also shows a tree hierarchy of the script in the left pane, with a breakdown of the document's components: Header and Body with the title, links, forms, and so forth.

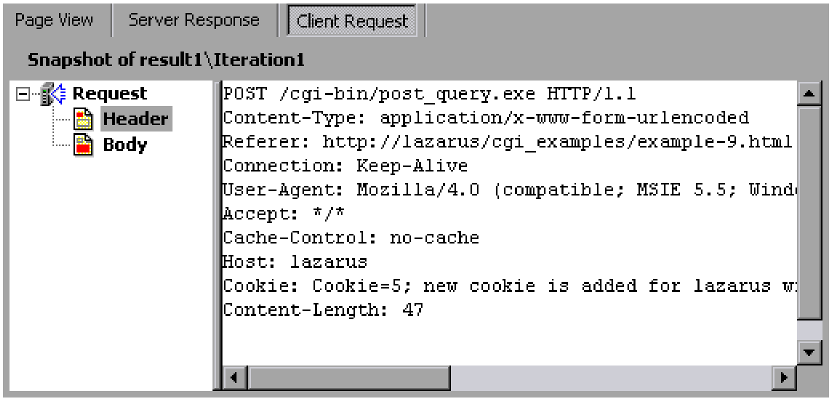
To find the HTML text of an element in the source in the right pane, select the element in the left pane of the Server Response, and select **Find Element** from the right-click menu.



Snapshot of Recording

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Tue, 05 Nov 2002 15:26:54 GMT
Connection: close
Content-type: text/html
```

- ➤ **Client Request.** Displays the client request HTML code of the snapshot. This button is available for both the recorded and replayed snapshots. The HTML view also shows a tree hierarchy of the script in the left pane, with a breakdown of the document's components: Header and Body and their sub-components.



Snapshot of result1\Iteration1

```
POST /cgi-bin/post_query.exe HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://lazarus/cgi_examples/example-9.html
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Wind
Accept: */*
Cache-Control: no-cache
Host: lazarus
Cookie: Cookie=5; new cookie is added for lazarus w
Content-Length: 47
```

Viewing Script Thumbnails

For several Vuser types such as Web, SAPGUI and Citrix, you can view thumbnail representations of the snapshots. You can view thumbnails in either Tree view or through the Transaction Editor.

Viewing Thumbnails in Tree View

In Tree view, the **Thumbnail** tab appears at the bottom of the Tree view window.

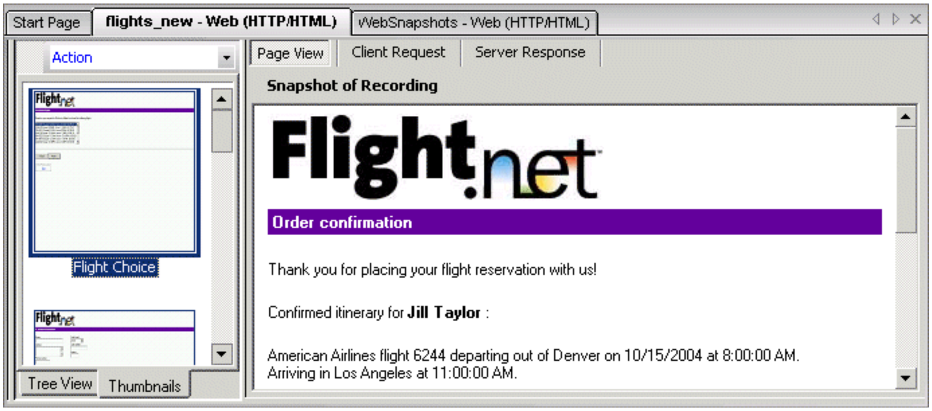
By default, the thumbnail view only shows primary steps in your script. To show all thumbnails, select **View** > **Show All Thumbnails**. VuGen shows the thumbnails for all of the steps in the script.

Note: For multiple iterations, the VuGen shows the replay thumbnails for the last iteration. To show the thumbnails of a specific iteration, select **View** > **Snapshot** > **Select Iteration** and select the desired iteration.

To view the thumbnails from Tree View:

1 Click the **Thumbnail** tab at the bottom of the left pane.

2 Click the desired thumbnail image to open the thumbnail's snapshot in the right pane.



Snapshot of Recording

Order confirmation

Thank you for placing your flight reservation with us!

Confirmed itinerary for **Jill Taylor** :

American Airlines flight 6244 departing out of Denver on 10/15/2004 at 8:00:00 AM.
Arriving in Los Angeles at 11:00:00 AM.

3 Double-click on a thumbnail image to view a larger image. A separate window opens showing a larger view of the thumbnail.

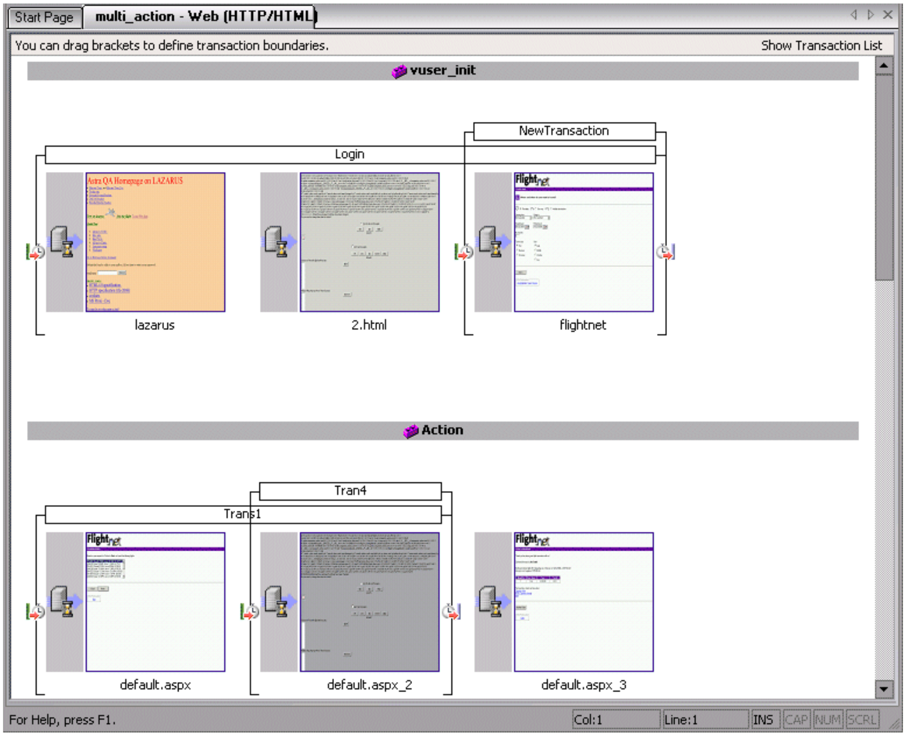
Setting the Mode for Viewing Actions

You can instruct VuGen how to view the script by its actions (**View** > **Actions**). By default, VuGen displays the script actions in the left pane when in Script view only. You can keep the default setting (**Open Automatically**), or open the action pane in the Tree view and/or Script view.

- **Open Automatically.** In Script view only, VuGen displays the script actions in the left pane. This is the default setting. Clear the selection to hide the actions pane.
- **Open in Tree Mode.** View the actions in Tree view. VuGen opens three panes: the actions, the steps, and the snapshot (if Snapshot view is enabled). You can adjust the width of each of these panes by dragging its border in the desired direction. Clear the selection to hide the actions pane.
- **Open in Script Mode.** View the actions in Script view. VuGen opens two panes: the actions, and the script view. You can adjust the width of the actions pane by dragging its border in the desired direction. Clear the selection to hide the actions pane.

Viewing Thumbnails in the Workflow Wizard

You can view the snapshots through the Transaction Editor. This view sorts the thumbnails by actions and provides you with an flat thumbnail view of all of the script's steps.

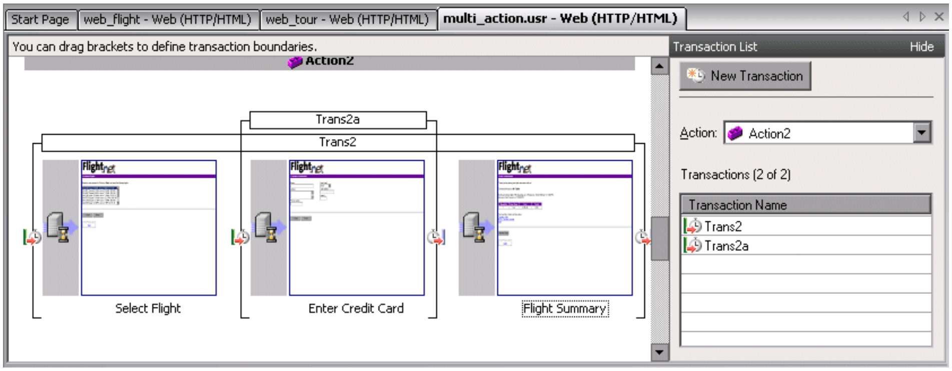


To view thumbnails in the Transaction Editor:

1. Click the **Tasks** button on the toolbar to open the task list pane.
2. Click the **Enhancements** > **Transactions** link. The Transaction Editor opens in the middle and right panes.

   For a more encompassing view, click **Tasks** to hide the Task list.
3. In the right pane, select the action that you want to view. VuGen displays the action that you selected.

In the following example, **Action2** was selected.



You can drag brackets to define transaction boundaries.

Action: Action2

Transactions (2 of 2)

Transaction Name
Trans2
Trans2a

Working with Thumbnails

VuGen lets you work with thumbnails by renaming them, annotating them, and viewing them in a larger size.

To view a thumbnail as a larger image:

Select **View Larger Image** from the right-click menu or press **Alt+F6**. A separate window opens showing a larger view of the thumbnail.

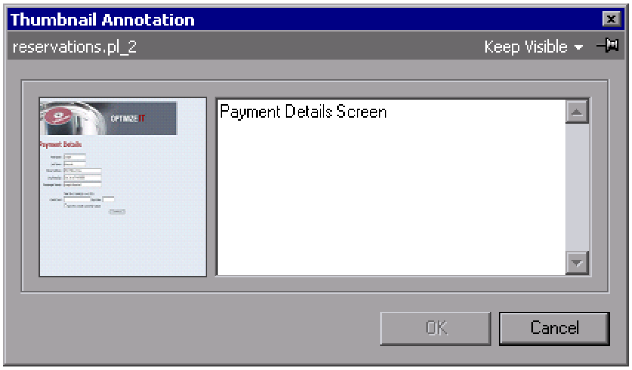
To rename a thumbnail:

1 Select the thumbnail and select **Rename** from the right-click menu or press F2.

2 Type in the desired text.

To annotate a thumbnail:

1 Select a thumbnail and select **Annotate** from the right-click menu or press Alt+F2. The Thumbnail Annotation dialog box opens.



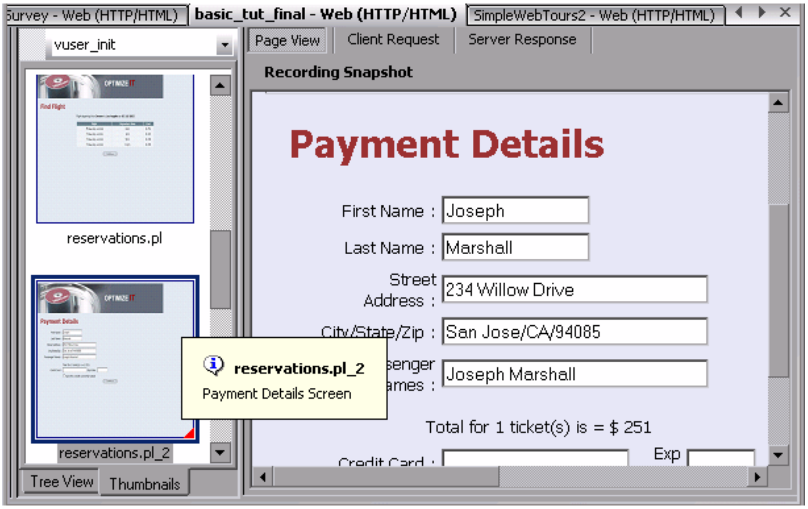
Payment Details Screen

2 Type text into the right pane of the Thumbnail Annotation dialog box.

3 Click **OK** to save the annotation and close the dialog box.

To leave the Annotation box open in order to add more text or work with other snapshots, select **Keep Visible** from the upper right corner. The **OK** button changes to **Apply**.

After you insert an annotation for a thumbnail, VuGen places a red mark in the bottom right corner to indicate that an annotation exists. If you move your mouse over the thumbnail, VuGen shows a popup of the annotation text.



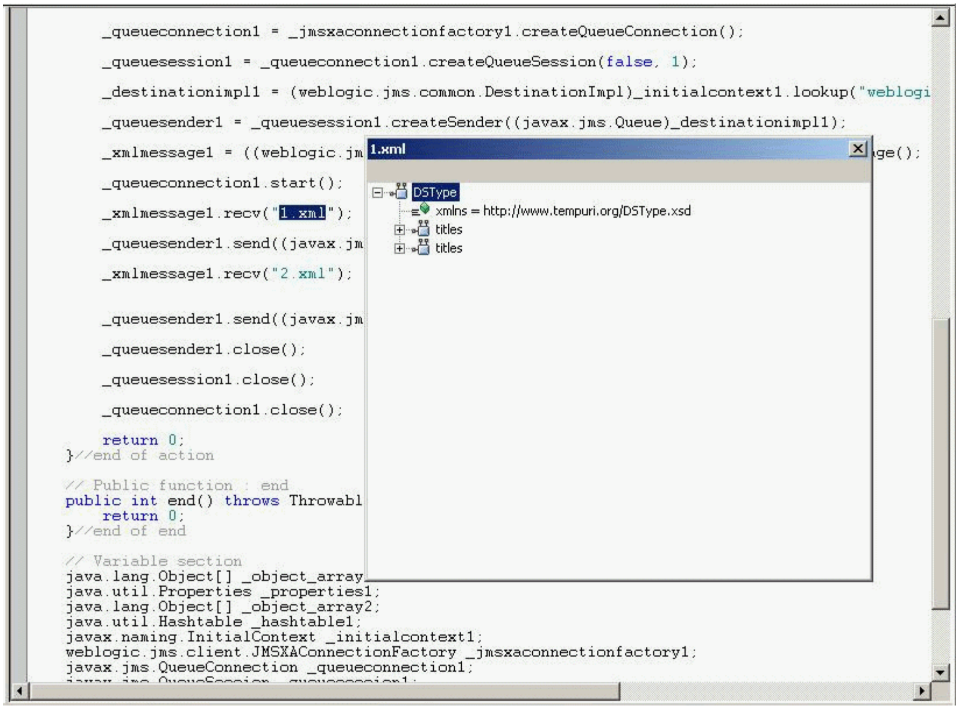
reservations.pl_2
Payment Details Screen

Payment Details

First Name : Joseph
Last Name : Marshall
Street Address : 234 Willow Drive
City/State/Zip : San Jose/CA/94085
Passenger Names : Joseph Marshall
Total for 1 ticket(s) is = $ 251
Credit Card : Exp

Using the XML Viewer

For certain protocols using XML files such as JMS, VuGen lets you view an XML structure directly from the editor window. A viewer displays the XML elements, and allows you to collapse or expand each of the nodes.



```
        _queueconnection1 = _jmsxaconnectionfactory1.createQueueConnection();

        _queuesession1 = _queueconnection1.createQueueSession(false, 1);

        _destinationimpl1 = (weblogic.jms.common.DestinationImpl)_initialcontext1.lookup("weblogi

        _queuesender1 = _queuesession1.createSender((javax.jms.Queue)_destinationimpl1);

        _xmlmessage1 = ((weblogic.jm                                                  ge();

        _queueconnection1.start();

        _xmlmessage1.recv("1.xml");

        _queuesender1.send((javax.jm

        _xmlmessage1.recv("2.xml");


        _queuesender1.send((javax.jm

        _queuesender1.close();

        _queuesession1.close();

        _queueconnection1.close();

        return 0;
    }//end of action

    // Public function : end
    public int end() throws Throwabl
        return 0;
    }//end of end

    // Variable section
    java.lang.Object[] _object_array
    java.util.Properties _properties1;
    java.lang.Object[] _object_array2;
    java.util.Hashtable _hashtable1;
    javax.naming.InitialContext _initialcontext1;
    weblogic.jms.client.JMSXAConnectionFactory _jmsxaconnectionfactory1;
    javax.jms.QueueConnection _queueconnection1;
```

To view an file in the XML viewer:

1 In Tree view, select a step with the XML file and select **View** > **XML**.

2 In Script view, right-click the XML file name and select **View XML**.

Running Vuser Scripts with VuGen

In order to perform testing or monitor an application with your Vuser script, you need to incorporate it into a LoadRunner scenario or Business Process Monitor profile. Before doing this, you check the script's functionality by running it from VuGen. For more information, see Chapter 9, "Running Vuser Scripts in Standalone Mode."

If the script replay is successful, you can then integrate it into your environment: a LoadRunner scenario, Performance Center load test, or Business Process Monitor profile. For more information, see the *HP LoadRunner Controller*, *HP Performance Center*, or *HP Business Availibility Center User Guides*.

Before you run a Vuser script, you can modify its run-time settings. These settings include the number of iterations that the Vuser performs, and the pacing and the think time that will be applied to the Vuser when the script is run. For more information on configuring run-time settings, see Chapter 22, "Configuring Run-Time Settings."

When you run a Vuser script, it is processed by an interpreter and then executed. You do not need to compile the script. If you modify a script, any syntax errors introduced into the script are noted by the interpreter. You can also call external functions from your script that can be recognized and executed by the interpreter. For more information, see Appendix A, "Calling External Functions."

Advanced users can compile a recorded script to create an executable program. For more information, see Chapter 6, "Enhancing Vuser Scripts."

Understanding VuGen Code

When you record a Vuser script, VuGen generates Vuser functions and inserts them into the script. There are two types of Vuser functions:

- General Vuser Functions
- Protocol-Specific Vuser Functions

The *general* Vuser functions and the *protocol-specific* functions together form the LoadRunner API. This API enables Vusers to communicate directly with a server. VuGen displays a list of all of the supported protocols when you create a new script. For syntax information about all of the Vuser functions, see the *Online Function Reference* (**Help** > **Function Reference**).

General Vuser Functions

The general Vuser functions are also called LR functions because each LR function has an **lr** prefix. The LR functions can be used in any type of Vuser script. The LR functions enable you to:

- Get run-time information about a Vuser, its Vuser Group, and its host.
- Add transactions and synchronization points to a Vuser script. For example, the **lr_start_transaction** (**lr.start_transaction** in Java) function marks the beginning of a transaction, and **the lr_end_transaction** (**lr.end_transaction** in Java) function marks the end of a transaction. See Chapter 6, "Enhancing Vuser Scripts" for more information.
- Send messages to the output, indicating an error or a warning.

See "Getting Help on Vuser Functions" on page 56 for a list of LR functions, and for details see the *Online Function Reference* (**Help** > **Function Reference**).

Protocol-Specific Vuser Functions

In addition to the general Vuser functions, VuGen also generates and inserts protocol-specific functions into the Vuser script while you record.

The protocol-specific functions are particular to the type of Vuser that you are recording. For example, VuGen inserts LRD functions into a database script, LRT functions into a Tuxedo script, and LRS functions into a Windows Sockets script.

By default, VuGen's automatic script generator creates Vuser scripts in C for most protocols, and in Java for Java type protocols. You can instruct VuGen to generate code in Visual Basic or Javascript. For more information, see Chapter 19, "Setting Script Generation Preferences."

All standard conventions apply to the scripts, including control flow and syntax. You can add comments and conditional statements to the script just as you do in other programming languages.

The following segment from a Web Vuser script shows several functions that VuGen recorded and generated in a script:

```
#include "as_web.h"


Action1()
{

    web_add_cookie("nav=140; DOMAIN=dogbert");

    web_url("dogbert",
        "URL=http://dogbert/",
        "RecContentType=text/html",
        LAST);

    web_image("Library",
        "Alt=Library",
        LAST);

    web_link("1 Book Search:",
        "Text=1 Book Search:",
        LAST);

    lr_start_transaction("Purchase_Order");
…
```

For more information about using C functions in your Vuser scripts, see Chapter 6, "Enhancing Vuser Scripts." For more information about modifying a Java script, see "Programming Java Scripts" in *Volume II-Protocols*.

Note: The C Interpreter used for running Vuser scripts written in C, only supports the ANSI C language. It does not support the Microsoft extensions to ANSI C.

Getting Help on Vuser Functions

You can add API Vuser functions to any script in order to enhance its capabilities. VuGen generates Vuser functions while you record. If required, you can manually insert additional functions into a script after recording. For information about typical enhancements, see Chapter 6, "Enhancing Vuser Scripts."

You can get help for VuGen’s API functions in several ways:

- Online Function Reference
- Word Completion
- Show Function Syntax
- Header File

In addition, you can use the standard Search feature (**Edit** > **Find**) to locate functions within a script, or the Find In Files feature on page 154 to search all of the files in the script.

Online Function Reference

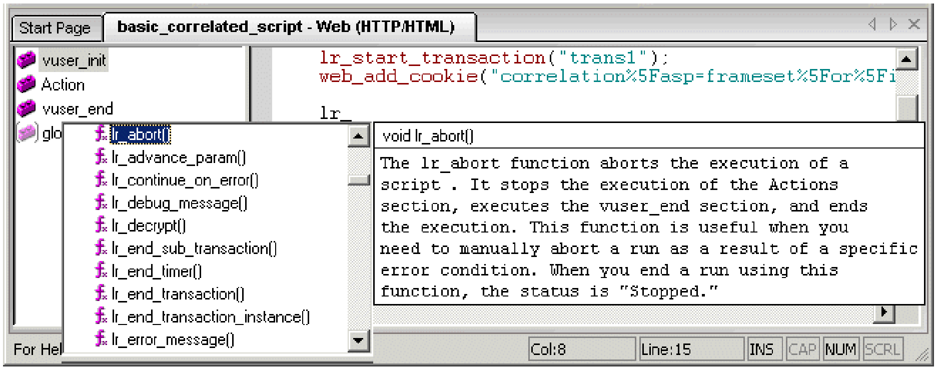
The *Online Function Reference* contains detailed syntax information about all of the VuGen functions. It also provides examples for the functions. You can search for a function by its name, or find it through a categorical or alphabetical listing.

To open the *Online Function Reference*, select **Help** > **Function Reference** from the VuGen interface. Then select a protocol and the desired category.

To obtain information about a specific function that is already in your script, place your cursor on the function in the VuGen editor, and press the F1 key.

Word Completion

As part of the *IntelliSense* enhancements, the VuGen editor incorporates the *Word Completion* feature. When you begin typing a function, after you type the first underscore, VuGen opens a list box displaying all available matches to the function prefix, along with the function's syntax and description.



```
lr_start_transaction("trans1");
web_add_cookie("correlation%5Fasp=frameset%5For%5Fi

lr_
```

void lr_abort()

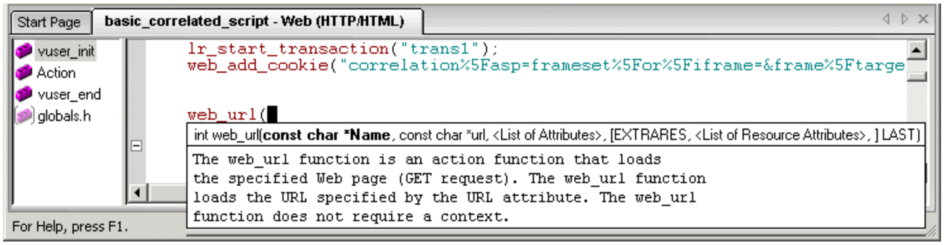
The lr_abort function aborts the execution of a script . It stops the execution of the Actions section, executes the vuser_end section, and ends the execution. This function is useful when you need to manually abort a run as a result of a specific error condition. When you end a run using this function, the status is "Stopped."

To use one of the displayed functions, select it, or scroll to the desired item and then select it. VuGen inserts the function at the location of the cursor. To close the list box, press the Esc key.

By default, VuGen uses word completion globally. To disable word completion, select **Tools** > **General Options** and select the Environment tab. Clear the check box adjacent to the **Auto complete word** option. If you disable word completion globally, you can still bring up the list box of functions by pressing Ctrl+Space or choosing **Edit** > **Complete Word** while typing in the editor.

Show Function Syntax

An additional feature of VuGen's Intellisense, is **Show Function Syntax**. When you type the opening parenthesis of a function, VuGen shows the syntax of the function with its arguments and prototypes and a brief description.



```
lr_start_transaction("trans1");
web_add_cookie("correlation%5Fasp=frameset%5For%5Fiframe=&frame%5Ftarge

web_url(
```

int web_url(**const char *Name**, const char *url, <List of Attributes>, [EXTRARES, <List of Resource Attributes>,] LAST)

```
The web_url function is an action function that loads
the specified Web page (GET request). The web_url function
loads the URL specified by the URL attribute. The web_url
function does not require a context.
```

By default, **Show Function Syntax** is enabled globally. To disable this feature, select **Tools** > **General Options** and select the Environment tab. Clear the check box adjacent to the **Auto show function syntax** option.

If you disable **Show Function Syntax** globally, you can still bring up the syntax by pressing Ctrl+Shift+Space or choosing **Edit** > **Show Function Syntax** after typing the opening parenthesis in the editor.

Header File

All of the non-Java function prototypes are listed in the library header files. The header files are located within the *include* directory of the product installation. They include detailed syntax information and return values. They also include definitions of constants, availability, and other advanced information that may not have been included in the Function Reference.

In most cases, the name of the header file corresponds to the prefix of the protocol. For example, Database functions that begin with an **lrd** prefix, are listed in the **lrd.h** file.

The following table lists the header files associated with the most commonly used protocols:

Protocol	**File**
AJAX (Click and Script)	**web_ajax.h**
Citrix	**ctrxfuncs.h**
COM/DCOM	**lrc.h**
Database	**lrd.h**
FTP	**mic_ftp.h**
General C function	**lrun.h**
IMAP	**mic_imap.h**
LDAP	**mic_mldap.h**
MAPI	**mic_mapi.h**
Oracle NCA	**orafuncs.h**
POP3	**mic_pop3.h**
RDP	**lrrdp.h**
SAPGUI	**as_sapgui.h**
SAP (Click and Script)	**sap_api.h**
Siebel	**lrdsiebel.h**
SMTP	**mic_smtp.h**
Terminal Emulator	**lrrte.h**
WAP	**as_wap.h**
Web (HTML\HTTP)	**as_web.h**
Web (Click and Script)	**web_api.h**
Web Services	**wssoap.h**
Windows Sockets	**lrs.h**

3

Using the Protocol Advisor

VuGen provides a variety of protocols that can be used to create Vuser scripts. The Protocol Advisor analyzes your business processes and provides guidance for selecting a protocol to record your script.

This chapter includes:

- About the Protocol Advisor on page 61
- Protocol Advisor Workflow on page 62
- Notes and Limitations on page 65

About the Protocol Advisor

You use the Protocol Advisor to help you determine an appropriate protocol for recording a Vuser script. The Protocol Advisor scans your application for elements of different protocols and displays a list of the detected protocols. These protocols should be used as guidelines and as a starting point for finding the optimal protocol for your application.

In most cases, more than one protocol is suggested and displayed, along with combinations of protocols. The following section contains guidelines on how to use the list of suggested protocols.

Protocol Advisor Workflow

The following steps represent a typical workflow for finding the optimal protocol to record your application using the Protocol Advisor:

1 Run the Protocol Advisor

a From the start page, select **File** > **Protocol Advisor** > **Analyze Application** or click the **Protocol Advisor** button. The Protocol Advisor dialog box opens.

b Fill in the fields as follows:

- ➤ **Application type.** Select the type of application.
- ➤ **Program to analyze.** Select the program to analyze.
- ➤ **URL Address.** This field is for Internet Applications only. Specify the starting URL address.
- ➤ **Program Arguments.** This field is for Win32 applications only. Specify command line arguments for the executable specified above. For example, if you specify **plus32.exe** with the program arguments **peter@neptune**, it connects the user **Peter** to the server **Neptune** when starting **plus32.exe**.
- ➤ **Working Directory.** For applications that require you to specify a working directory, specify it here. The required information differs, depending on the type of Vuser script.

c Click **OK**. A box is displayed showing the progress of the analysis.

d Perform typical business processes in your application. Try to walk through a variety of business processes to make sure that your results are comprehensive.

e Click **Stop Analyzing** to end the analysis.

The Protocol Advisor Results page displays the results.

2 Save the results.

Select **File** > **Protocol Advisor** > **Save Results**. Enter a name and select the directory.

3 Select the protocol to use.

We recommend selecting the protocols using the following order of priority:

- ➤ Multi/Combination protocol
- ➤ The highest level application protocol (see the diagram in step 6 on page 64)
- ➤ The first protocol on the list

4 Create a new Vuser script.

Create a new Vuser script according to the sections about the selected protocol.

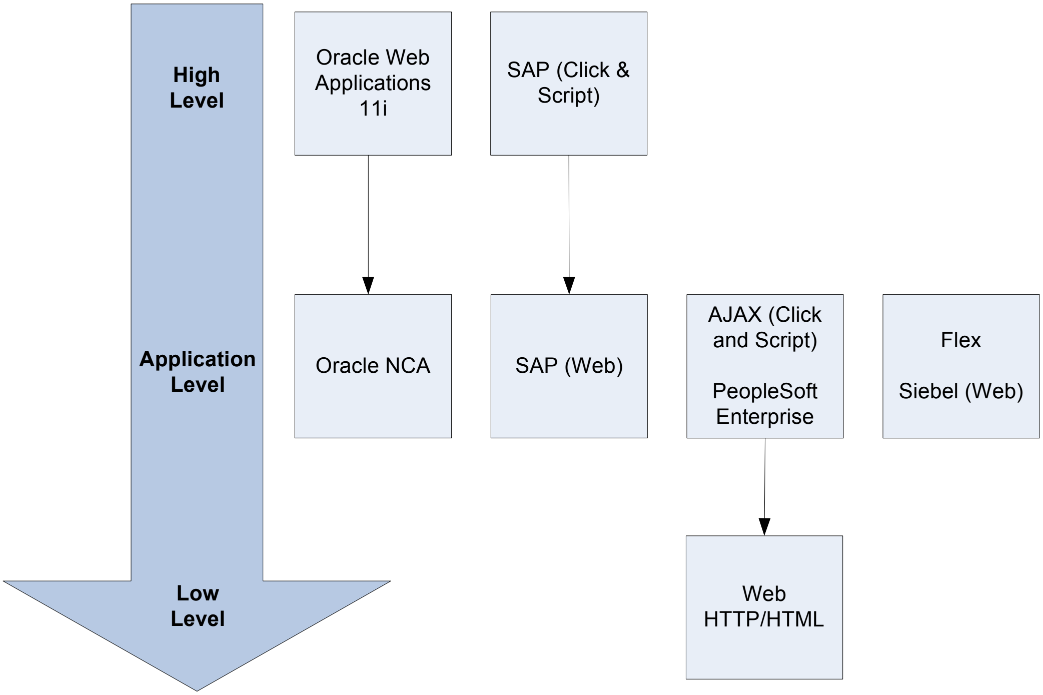
5 Enhance, debug, and verify replay.

Enhance and debug your script until you can replay it successfully. If, after enhancing and debugging the Vuser script, you cannot replay it successfully, proceed to the next step.

6 If you cannot replay your script, select a different protocol.

➤ **For all non Web-based protocols:** Return to the Protocol Advisor Results page and select the next protocol on the list and continue from step 2. If there are no more protocols listed, contact HP Customer Support.

➤ **For Web-based protocols:** The following diagram illustrates the Web-based protocols arranged according to their application level. The arrows indicate the order in which you should attempt a new protocol. For example, if the first protocol you used was Oracle Web Applications 11i and it failed to successfully replay, you would then try the Oracle NCA protocol.



Notes and Limitations

The following notes and limitations should be considered:

- This feature will only detect protocols supported by LoadRunner.
- Some Web protocols are detected based on the URL. For example, the URL may contain keywords such as SAP. Therefore, if you use a different URL or a different application with the same URL, the results may differ.
- The following protocols are frequently detected but are not always appropriate for use. You should only use them if there are no other detected protocols.
  - COM/DCOM
  - Java
  - .Net
  - WinSocket
  - LDAP

4

Viewing the VuGen Workflow

VuGen’s workflow screens guide you through the creation of a Vuser script that can be used for load testing or monitoring.

This chapter includes:

- About Viewing the VuGen Workflow on page 67
- Viewing the Task Pane on page 68
- Recording Steps on page 69
- Verifying the Script on page 70
- Enhancing the Script on page 72
- Prepare for Load on page 78
- Finishing Your Script on page 79
- Restoring the Workflow View on page 79

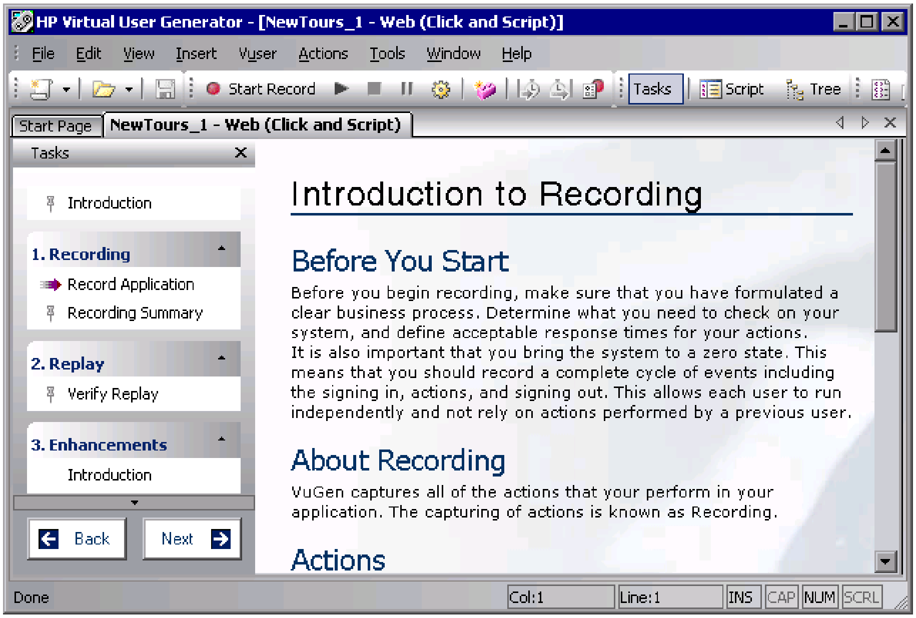
About Viewing the VuGen Workflow

VuGen’s workflow screens walks you through the different steps of creating a script.

You can also work in Script View or Tree View. The next time you start VuGen and open a script, it opens to the view that you had open when you exited VuGen. You can switch back to the wizard view by clicking on any task in the Task Pane.

Viewing the Task Pane

VuGen's left pane shows a list of the tasks required in order to create a functional script. Click on any task within the list to open that step in the wizard. VuGen indicates the current task with an arrow.



Introduction to Recording

Before You Start

Before you begin recording, make sure that you have formulated a clear business process. Determine what you need to check on your system, and define acceptable response times for your actions.
It is also important that you bring the system to a zero state. This means that you should record a complete cycle of events including the signing in, actions, and signing out. This allows each user to run independently and not rely on actions performed by a previous user.

About Recording

VuGen captures all of the actions that your perform in your application. The capturing of actions is known as Recording.

Actions

The list of tasks is divided into five parts: **Recording (Script Creation** in C and Web Services Vusers), **Replay**, **Enhancements**, **Prepare for Load**, and **Finish**.

The initial task differs slightly between Web, Web Services and non-recordable protocols, such as Custom C Vusers.

Many of the tasks have sub-tasks. The following table lists them.

Task	Sub Tasks
Recording	Record Application, Recording Summary (recordable protocols)
Script Creation	Create Script, Creation Summary (for Web Services, C))
Replay	Verify Replay
Enhancements	Introduction, Transactions, Parameterization, Content Checks (for Web Vusers)
Prepare for Load	Introduction, Iterations, Concurrent Users

Recording Steps

The Recording Section (excluding Web Services, C, and non-recordable protocols) has two steps: Record Application and Recording Summary.

Record Application

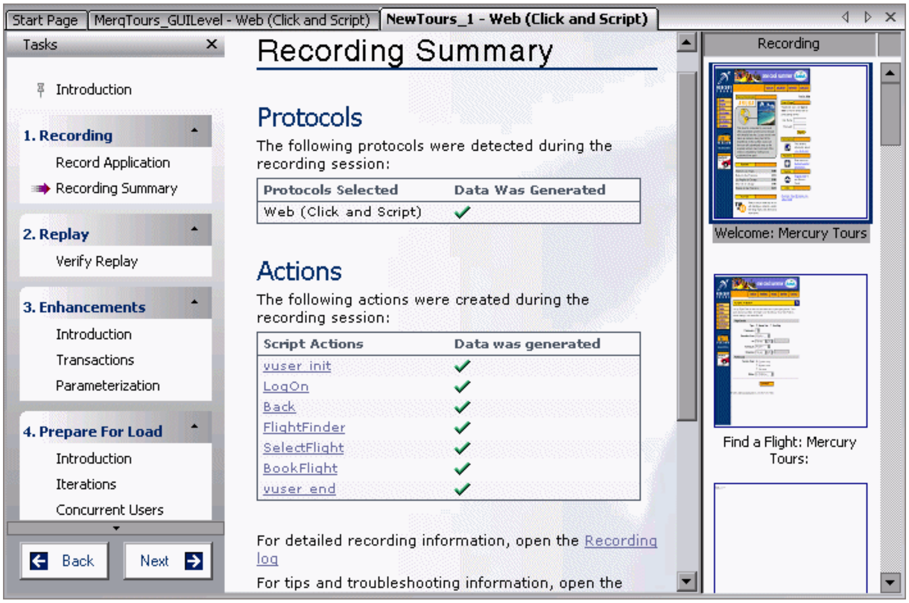
This wizard step provides an introduction to the recording process and contains the following sections:

- Before you Start
- About Recording
- Actions
- Recording Options
- Start Recording

Recording Summary

This wizard step provides a summary of the recording including the protocol information and the actions into which the session was recorded.

This step also provides thumbnails of the recorded snapshots.



Recording Summary

Protocols

The following protocols were detected during the recording session:

Protocols Selected	Data Was Generated
Web (Click and Script)	✓

Actions

The following actions were created during the recording session:

Script Actions	Data was generated
vuser_init	✓
LogOn	✓
Back	✓
FlightFinder	✓
SelectFlight	✓
BookFlight	✓
vuser_end	✓

For detailed recording information, open the Recording log

For tips and troubleshooting information, open the

Verifying the Script

The Verification section contains the **Verify Replay** step. Note that once you replay the script, you can view a Replay summary at any time by clicking **Replay Summary** in the **General** section, below the **Finish** step.

Verify Replay

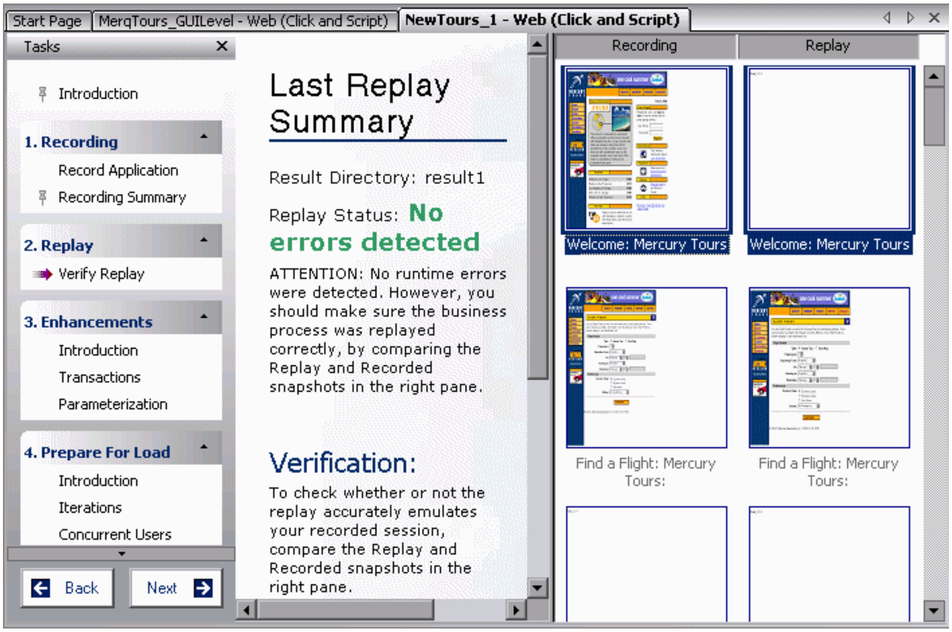
This wizard step provides an introduction to verification and contains the following sections.

- About Replay
- Run-Time Settings
- Before Replay (what to look for during replay)

Replay Summary

This wizard step provides a summary of the replay. It list the errors and provides a link to the error in the replay log.

The Replay summary also shows thumbnails of the Recording and Replay snapshots. You can visually compare the snapshots and look for discrepancies.



Last Replay Summary

Result Directory: result1

Replay Status: No errors detected

ATTENTION: No runtime errors were detected. However, you should make sure the business process was replayed correctly, by comparing the Replay and Recorded snapshots in the right pane.

Verification:

To check whether or not the replay accurately emulates your recorded session, compare the Replay and Recorded snapshots in the right pane.

Note: For multiple iterations, the Replay Summary window shows the replay thumbnails for the last iteration. To show the thumbnails of a specific iteration, select **View** > **Tree View** to switch to Tree view. Then select **View** > **Snapshot** > **Select Iteration** and select the desired iteration.

Enhancing the Script

The Enhancement Section has three primary steps:

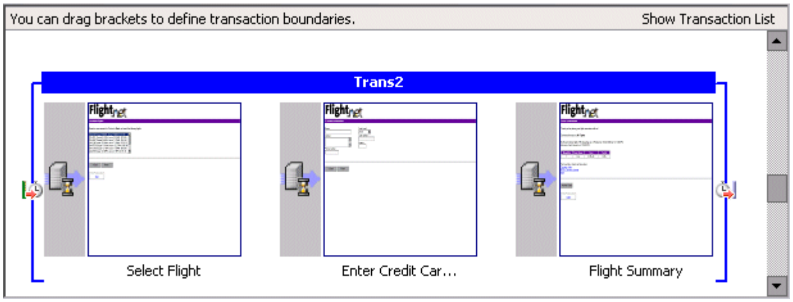
- Transactions
- Parameterization
- Content Check.

Transactions

VuGen uses the **Transaction Editor** to allow you to add and manage transactions directly from a thumbnail view of the script.

By default, VuGen only displays thumbnails for the primary steps in your script. To display thumbnails for all of the steps in your script, select **View** > **Show All Thumbnails**.

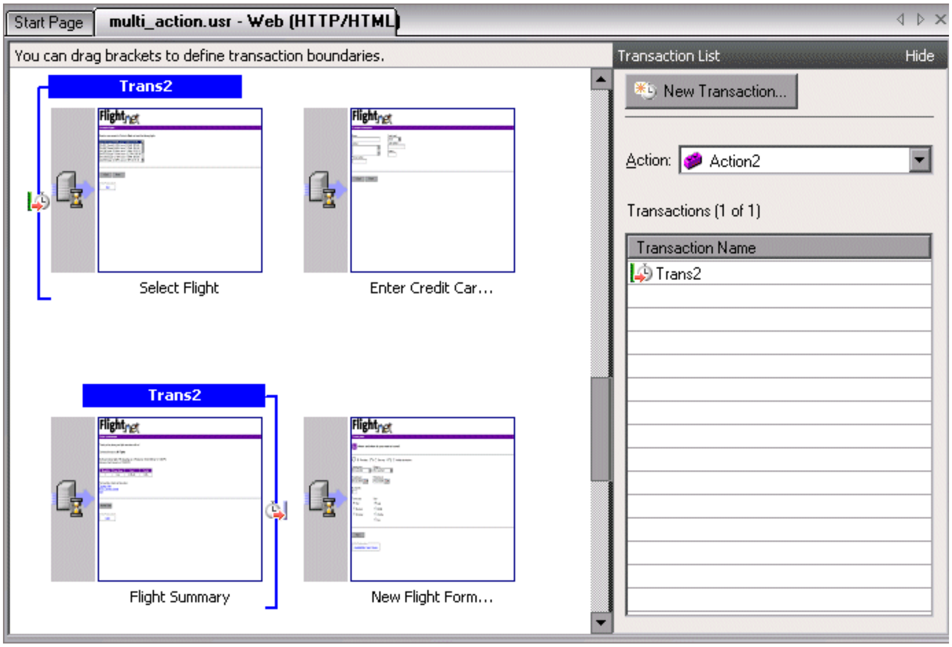
In the following example, *Trans2* measures the time for the three steps: **Select Flight**, **Enter Credit Card**, and **Flight Summary**.



You can drag brackets to define transaction boundaries.

Working with the Transactions List

The transaction list, in the right pane of the Transaction editor, shows a list of the transactions in the script. You can view a complete list of the transactions in the script, or only those in a specific action.

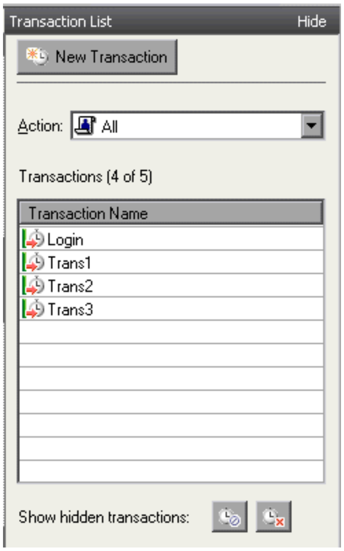


Action: Action2

Transactions (1 of 1)

Transaction Name
Trans2

To view all transactions, select **All** in the **Action** box. To view the transactions in a specific action, select the action name in the **Action** box.



To show and hide the Transaction list, click the **Hide** or **Show Transaction List** button in the upper right corner of the VuGen window.

By default, the transaction list only shows transactions without errors that measure the server response for primary steps in the script. It does not show:

- Non-primary steps
- Client side transactions
- Transactions with errors

Therefore, you might see a caption similar to the following caption above the transaction list: Transactions (2 of 4).

To show the hidden transactions—the non-primary and client side transactions—click the button adjacent to **Show hidden transactions** at the bottom of the transaction list. VuGen lists the hidden transactions in gray. To hide them, click the button again.

Transactions with errors are those that do not measure any server steps, or those with illegal names. To show the transactions with errors, click the **Show transactions with errors** button. VuGen lists the transactions with errors in red. To hide them, click the button again.

To show the transactions for non-primary steps, you need to display all of the thumbnails. Select **View** > **Show All Thumbnails**. The Transaction Editor shows the thumbnails of all the steps in the script and their transactions.

Defining Transactions in the Transaction Editor

You define a transaction in the Transaction Editor by marking the starting and ending thumbnails of the transaction.

To define a transaction:

1. Click **Transactions** in the Task list to open the Transaction Editor.
2. In the thumbnail area (middle pane), scroll down to the steps that you want to mark as a a transaction.

   Tip: To see more thumbnails per page, click the toolbar's **Tasks** button to hide the Tasks list.

3. To mark a single step as a transaction, click on a thumbnail and select **New Single-Step Transaction** from the right-click menu. VuGen prompts you to provide a name for the new transaction. If you want to expand the transactions at a later time, you can drag the transaction brackets to include additional steps.
4. To mark multiple steps as a transaction, click in the thumbnail area and select **New Transaction** from the right-click menu or click the **New Transaction** button in the top of the right pane.
5. VuGen displays instructions in the status area above the thumbnails as follows:

   Step 1 of 3: Select a starting point for the new transaction.

   Click the thumbnail of the transaction's first step.

Step 2 of 3: Select where the new transaction should end.

Click the thumbnail of the transaction's last step.

Step 3 of 3: Specify a name for the new transaction.

Type in a transaction name in the bracket directly above the transaction's first step.

To complete the transaction, press the Enter key.

To exit a transaction during the above sequence, press the Esc key.

6 To change the starting point of a transaction, drag the transaction opening bracket to a new location. To change the ending point of a transaction, drag the transaction closing bracket to a new location.

7 Repeat the above steps for additional transactions.

- To rename a transaction, select its title in the right pane and select **Rename Transaction** from the right-click menu. Type in the new name.
- To delete a transaction, select its title in the right pane and select **Delete Transaction** from the right-click menu.

Guidelines for the Transaction Editor

Follow these guidelines when creating and defining transactions in the Transaction Editor:

- Transactions must begin and end within a single action—they may not extend over multiple actions.
- Transaction names must be unique within your script, even between actions.
- Use the right-click menu to add, rename, or delete transactions.
- You can create transactions within an existing transaction. These are called *nested* transactions.

Note: If you nest transactions, close the second transaction before or at the same time as you close the first one—otherwise it won't be analyzed properly.

Parameterization

The Parameterization screen provides you with an overview of parameterizing values in your script. It guides you through the steps of parameterization:

- Locate the argument you want to parameterize
- Give the parameter a name
- Select a parameter type
- Define properties for the parameter type
- Replace the argument with a parameter

After you parameterize an argument in your script, you can return to the **Enhancements** step or replay the script.

Content Check

The Content Check wizard step lets you check for specific text or content in replayed web pages.

Using a **Text Check**, you can search for a text string during replay.

Using **Content Checks**, you can instruct VuGen to search for server strings automatically using predefined rules, even if you don't know the exact text that will be returned by the server.

Prepare for Load

The fourth part of the Workflow Wizard is primarily for running load tests on your system to check the response and capacity of your machine. This part has two primary steps:

- Iterations
- Concurrent Users

Iterations

This wizard step provides an introduction to iterations and allows you to open the Run-Time settings for setting their values.

To set the number of iterations:

1. Open the Run-Time settings (F4).
2. Select the Run Logic node.
3. Specify the number of iterations.

Concurrent Users

This step guides you through the process of creating a scenario using the LoadRunner Controller.

In a scenario, you can specify the number of users to run concurrently and you can observe the behavior of your system with multiple users.

Finishing Your Script

The final step of the Workflow wizard is **Finish**.

It contains the following sections:

- **Create a Scenario.** To run a load test on your system using the LoadRunner Controller.
- **Upload to Performance Center.** To run a test through a Performance Center server installation.
- **Upload to Quality Center.** To add a test to the test repository.
- **Create Business Process Report.** Create a Microsoft Word document containing a summary of the VuGen script.

Restoring the Workflow View

If your VuGen view no longer display the Workflow views, you can easily return to it:

- Make sure the **Tasks** Pane is visible in the left pane. If not, click the **Tasks** button on the toolbar.
- Click the top link, **Introduction**. The right pane displays the first Workflow screen.

Tip: To instruct VuGen to display the Workflow view after every replay, open the **Replay** tab (**Tools** > **General Options**) and select the **Replay summary** option in the After Replay section.

5

Recording with VuGen

VuGen creates a Vuser script by recording the communication between a client application and a server.

This chapter includes:

- ➤ About Recording with VuGen on page 82
- ➤ Vuser Script Sections on page 82
- ➤ Creating New Virtual User Scripts on page 84
- ➤ Adding and Removing Protocols on page 87
- ➤ Choosing a Virtual User Category on page 88
- ➤ User-Defined Template on page 89
- ➤ Creating a New Script on page 91
- ➤ Opening an Existing Script on page 92
- ➤ Recording Your Application on page 93
- ➤ Ending and Saving a Recording Session on page 97
- ➤ Viewing the Recording Logs on page 98
- ➤ Using Zip Files on page 100
- ➤ Importing Actions on page 101
- ➤ Providing Authentication Information on page 102
- ➤ Regenerating a Vuser Script on page 104

About Recording with VuGen

VuGen creates a Vuser script by recording the actions that you perform on a client application. When you run the recorded script, the resulting Vuser emulates the user activity between the client and server.

Each Vuser script that you create contains at least three sections: *vuser_init*, one or more *Actions*, and *vuser_end*. During recording, you can select the section of the script into which VuGen will insert the recorded functions. In general, you record a login to a server into the *vuser_init* section, client activity into the *Actions* sections, and the logoff procedure into the *vuser_end* section.

After creating a test, you can save it to a zip archive and send it as an email attachment.

While recording, you can insert transactions, comments, and rendezvous points into the script. For details, see Chapter 6, "Enhancing Vuser Scripts."

Vuser Script Sections

Each Vuser script contains at least three sections: vuser_init, one or more Actions, and vuser_end. Before and during recording, you can select the section of the script into which VuGen will insert the recorded functions. The following table shows what to record into each section, and when each section is executed.

Script Section	Used when recording...	Is executed when...
vuser_init	a login to a server	the Vuser is initialized (loaded)
Actions	client activity	the Vuser is in **Running** status
vuser_end	a logoff procedure	the Vuser finishes or is stopped

When you run multiple iterations of a Vuser script, only the *Actions* sections of the script are repeated—the *vuser_init* and *vuser_end* sections are not repeated. For more information on the iteration settings, see Chapter 22, "Configuring Run-Time Settings."

You use the VuGen script editor to display and edit the contents of each of the script sections. You can display the contents of only a single section at a time. To display a section, highlight its name in the left pane.

When working with Vuser scripts that use Java classes, you place all your code in the Actions class. The Actions class contains three methods: init, action, and end. These methods correspond to the sections of scripts developed using other protocols—you insert initialization routines into the init method, client actions into the action method, and log off procedures in the end method. For more information, see "Programming Java Scripts" in *Volume II-Protocols*.

```
public class Actions{
    public int init() {
        return 0;}
    public int action() {
        return 0;}
    public int end() {
        return 0;}
}
```

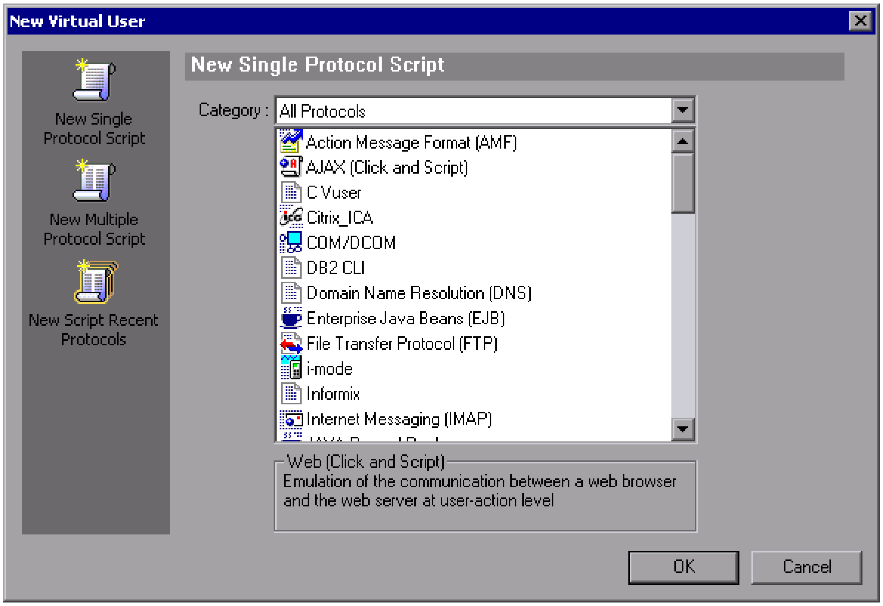
Note: Transaction Breakdown for Oracle DB is not available for actions recorded in the vuser_init section.

Creating New Virtual User Scripts

VuGen allows you to create new scripts by recording in either single or multi-protocol mode. It also provides a solution for creating scripts in a Service Oriented Architecture (SOA) environment.

The New Virtual User window opens whenever you click **New**. This dialog box provides a shortcut to the following:

New Single Protocol Script. Creates a single protocol Vuser script. Select a category from the Category list (see "Choosing a Virtual User Category" on page 88), and select a protocol in the protocol list under that category.

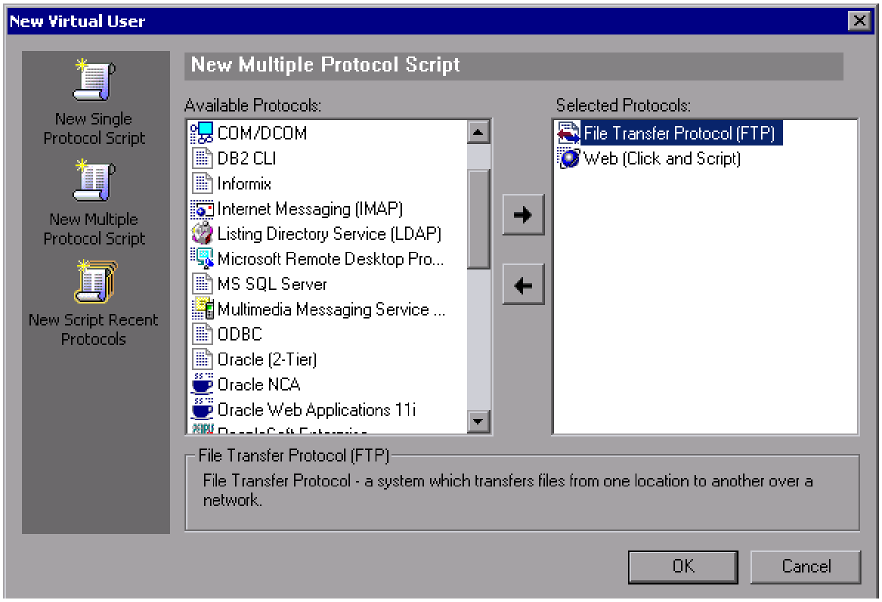

New Virtual User
New Single Protocol Script
New Multiple Protocol Script
New Script Recent Protocols
New Single Protocol Script
OK
Cancel

Category : All Protocols

- Action Message Format (AMF)
- AJAX (Click and Script)
- C Vuser
- Citrix_ICA
- COM/DCOM
- DB2 CLI
- Domain Name Resolution (DNS)
- Enterprise Java Beans (EJB)
- File Transfer Protocol (FTP)
- i-mode
- Informix
- Internet Messaging (IMAP)

Web (Click and Script)
Emulation of the communication between a web browser and the web server at user-action level

New Multiple Protocol Script. Creates a multiple protocol Vuser script. VuGen displays all of the available protocols and allows you to specify which protocols to record. To create a multiple protocol script, select a protocol and click the right arrow to move it into the Selected Protocols section.


New Virtual User
New Single Protocol Script
New Multiple Protocol Script
New Script Recent Protocols
New Multiple Protocol Script
Available Protocols:
COM/DCOM
DB2 CLI
Informix
Internet Messaging (IMAP)
Listing Directory Service (LDAP)
Microsoft Remote Desktop Pro...
MS SQL Server
Multimedia Messaging Service ...
ODBC
Oracle (2-Tier)
Oracle NCA
Oracle Web Applications 11i
Selected Protocols:
OK
Cancel

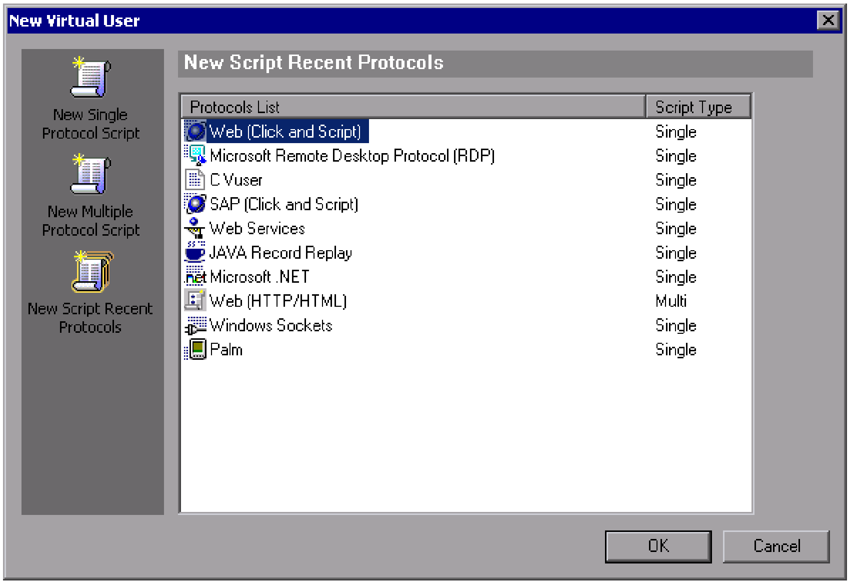
Selected Protocols:

- File Transfer Protocol (FTP)
- Web (Click and Script)

File Transfer Protocol (FTP)

File Transfer Protocol - a system which transfers files from one location to another over a network.

New Script Recent Protocols. Lists the most recent protocols that were used to create new Vuser scripts and indicates whether they were single or multi protocol. Select a protocol from the list and click **OK** to create a new script for that protocol.


New Virtual User
New Single Protocol Script
New Multiple Protocol Script
New Script Recent Protocols
New Script Recent Protocols
OK
Cancel

Protocols List	Script Type
Web (Click and Script)	Single
Microsoft Remote Desktop Protocol (RDP)	Single
C Vuser	Single
SAP (Click and Script)	Single
Web Services	Single
JAVA Record Replay	Single
Microsoft .NET	Single
Web (HTTP/HTML)	Multi
Windows Sockets	Single
Palm	Single

When you record a single protocol, VuGen only records the specified protocol. When you record in multi-protocol mode, VuGen records the actions in several protocols. Multi-protocol scripts are supported for the following protocols: COM, FTP, IMAP, Oracle NCA, POP3, RealPlayer, Window Sockets (raw), SMTP, and Web.

Another variation between Vuser types is multiple-action support. Most protocols support more than one action section. Currently, the following protocols support multi-actions: Oracle NCA, Web, RTE, General (C Vusers), WAP, i-Mode, and VoiceXML.

For most Vuser types, you create a new Vuser script each time you record—you cannot record into an existing script. However, when recording a Java, Web, WAP, i-mode, Oracle NCA, or RTE Vuser script, you can also record within an existing script.

Since VuGen supports a large variety of protocols, some of the recording steps that follow apply only to specific protocols.

For all Java language Vusers (CORBA, RMI, Jacada, and EJB) see "Java Protocols - Recording" in *Volume II-Protocols*.

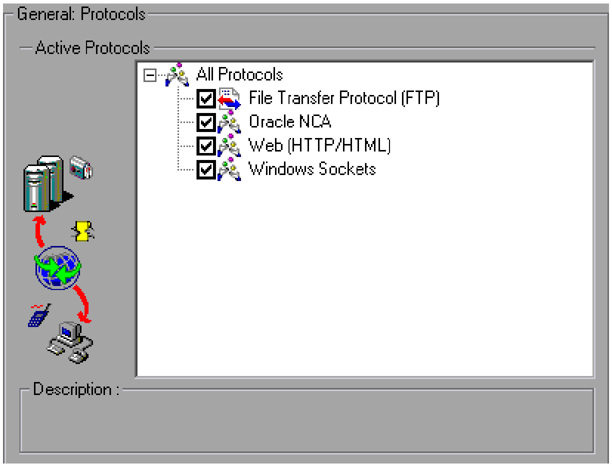
In SOA (Service Oriented Architecture) systems, it is essential that you test the stability of your applications and services before deployment. **LoadRunner VuGen** allows you to create basic Web Service scripts. **HP LoadRunner with Service Test**, HP's SOA testing tool, contains additional features that help you create a comprehensive testing solution for your SOA environment. For more information about **Service Test**, contact an HP representative.

Adding and Removing Protocols

Before recording a multi-protocol session, VuGen lets you modify the protocol list for which to generate code during the recording session. If you specified certain protocols when you created the script, you can enable or disable them using the Protocol Recording options.

To open the recording options, select **Tools** > **Recording Options** or press Ctrl+F7. Select the **General:Protocols** node.

Select the check boxes adjacent to the protocols you want to record in the next recording session. Clear the check boxes adjacent to the protocols you do not want to record in the next recording session.



General: Protocols

Active Protocols

- All Protocols
  - ☑ File Transfer Protocol (FTP)
  - ☑ Oracle NCA
  - ☑ Web (HTTP/HTML)
  - ☑ Windows Sockets

Description :

For information on setting the Protocol options for the AMF and Web protocols, see "Setting the AMF Recording Mode" on page 304.

Choosing a Virtual User Category

The Vuser types are divided into the following categories:

- **All Protocols.** A list of all supported protocols in alphabetical order
- **Application Deployment Solution:** For the Citrix and Microsoft Remote Desktop Protocol (RDP) protocols
- **Client/Server.** For DB2 CLI, DNS, Informix, Microsoft .NET, MS SQL, ODBC, Oracle 2-Tier, DB2 CLI, Sybase Ctlib, Sybase Dblib, and Windows Sockets, protocols
- **Custom.** For C Template, Visual Basic template, Java template, VBNet, Javascript, and VBscript type scripts
- **Distributed Components.** For COM/DCOM and Microsoft .NET protocols

- **E-Business.** For AMF, AJAX (Click and Script), Flex, FTP, LDAP, Microsoft .NET, Web (Click and Script), Web (HTTP/HTML), and the Web Services/SOA protocols
- **ERP/CRM.** For Oracle NCA, Oracle Web Applications 11i, Peoplesoft Enterprise, Peoplesoft-Tuxedo, SAP-Web, SAPGUI, SAP (Click and Script), and Siebel Web protocols
- **Enterprise Java Beans.** For the EJB Testing protocol
- **Java.** For recording and replaying Java type protocols such as CORBA, RMI-Java, and JMS
- **Legacy.** For Terminal Emulation (RTE)
- **Mailing Services.** Internet Messaging (IMAP), MS Exchange (MAPI), POP3, and SMTP
- **Middleware.** For the Tuxedo protocol
- **Streaming.** For Real and Media Player (MMS) protocols
- **Wireless.** For Multimedia Messaging Service (MMS) and WAP protocols

User-Defined Template

The User-Defined Template enables you to save a script with a specific configuration as a template. You can then use this template as a basis for creating future scripts.

The template supports the following files and data:

- Run-Time Settings
- parameters
- extra files
- actions
- snapshots

Recording and General options are not supported as they are generic settings and are not relevant to a specific script.

Saving and Creating Templates

You can save a script as a template or open a script from a saved template.

To save a script as a template:

1. Open the script in VuGen.
2. Select **File** > **User-Defined Template** > **Save As Template**. The Save Script As Template dialog opens. There is an icon by all existing template folders.
3. Enter the name of the template to be saved in the **File name** area and click **Save**. The script is saved as a template in the location you specified, and is added to the list.

To open a script from a template:

1. Select **File** > **User-Defined template** > **Create Script From Template**. The Create Script From Template dialog opens. There is an icon by all existing template folders.
2. Select the template you want to use as a basis for your script and click **Open**. A new script, based on the template, opens in VuGen.

Notes and Limitations

- Once you have configured a script for a specific protocol and then save the script as a template, further scripts based on that template will only work with that same protocol. For example, if you configured your script for the Web (Click and Script) protocol, then created a template from that script, the template can only be used with Web (Click and Script).
- Once you have created your template, you cannot edit it directly in VuGen. To make any changes, you open the template and save it again with another name or overwrite the existing template.
- If you regenerate an original script from a template, you will lose all of your manual changes.

Creating a New Script

This section explains how to invoke VuGen and create a new script.

To create a new Vuser script:

1. Select **Start** > **Programs** > *application_name* > **Applications** > **Virtual User Generator** to start VuGen. The startup screen opens.
2. To create a single protocol script, make a selection from the Category list and select one of the protocols.
3. To create a multi-protocol script, allowing you to record two or more protocols in a single recording session, click the **New Multiple Protocol Script** button in the left pane to enable the Protocol Selection window.

   Select the desired protocol from the **Available Protocols** list. Click the right-facing arrow to move the selection into the **Selected Protocols** list. Repeat this step for all of the desired protocols.

   Note: When recording certain Oracle NCA applications, you only need to select **Oracle NCA**—not **Web Protocol**. For details, see "Oracle NCA Protocol" in *Volume II-Protocols*.

4. Click **OK** to close the dialog box and begin generating the Vuser script.

Tip: Alternatively, you can open a script from a User-Defined Template. For details, see "User-Defined Template" on page 89.

Opening an Existing Script

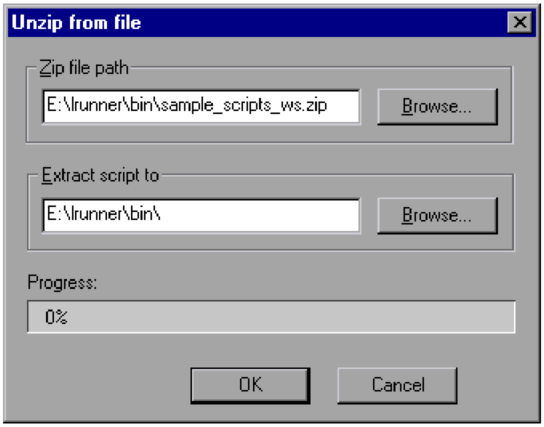
If you already have a script on the local machine or network, you can modify it and record additional actions.

To open an existing script:

1 To open a script stored on your local machine or a network drive, select **File** > **Open**.

2 To open a file from a Quality Center repository (LoadRunner only), see "Opening Scripts from a Quality Center Project" on page 192.

3 To open a script stored in a compressed zip file, select **File** > **Zip Operations** > **Import from Zip File**. After you select a zip file, VuGen prompts you for a location at which to store the unzipped files.



Zip file path: E:\lrunner\bin\sample_scripts_ws.zip

Extract script to: E:\lrunner\bin\

Progress: 0%

4 To work from a zip file, while not expanding or saving the script files, select **File** > **Zip Operations** > **Work from Zip File**. When you modify the script and save it, the changes are stored directly in the zip file.

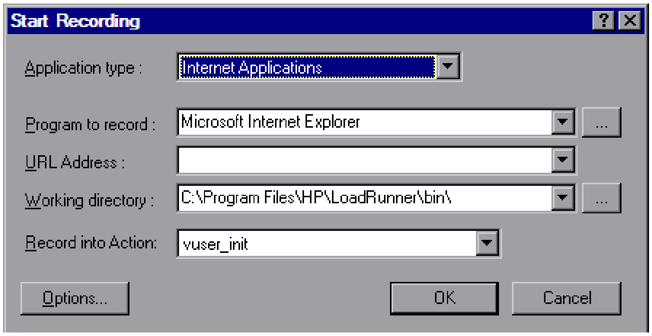
Recording Your Application

For most Vuser script types, VuGen automatically opens the Start Recording dialog box when you create the new script.

To begin recording:

1. If the Start Recording dialog box was not opened, click the **Start Recording** button. The Start Recording dialog box opens. This dialog box differs slightly for each protocol.

2. For most Client/Server protocols, the following dialog box opens:



Start Recording

Application type : Internet Applications

Program to record : Microsoft Internet Explorer

URL Address :

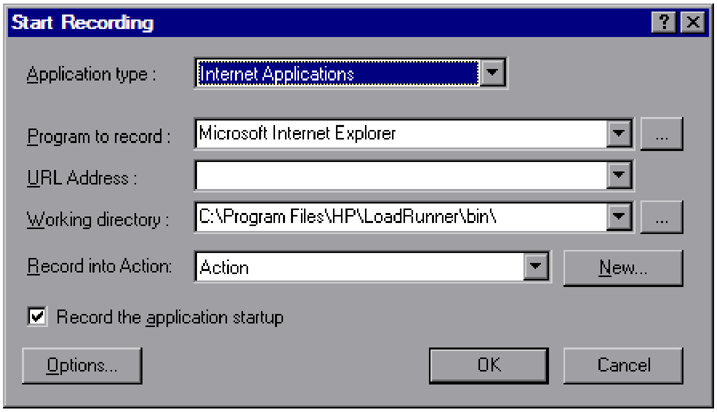
Working directory : C:\Program Files\HP\LoadRunner\bin\

Record into Action: vuser_init

Enter the program to record, the working directory, (optional) and the Action. If applicable, click **Options** to set the recording options.

3. For non-Internet applications, select the application type: Win32 Applications or Internet Applications. For example, Web and Oracle NCA scripts record Internet Applications, while Windows Socket Vusers records a Win32 application. For Citrix ICA Vusers, VuGen automatically records the Citrix client—you only need to specify the action in **Record into Action**.

4 For Internet-based applications, fill in the relevant information:



Start Recording

Application type : Internet Applications

Program to record : Microsoft Internet Explorer

URL Address :

Working directory : C:\Program Files\HP\LoadRunner\bin\

Record into Action: Action

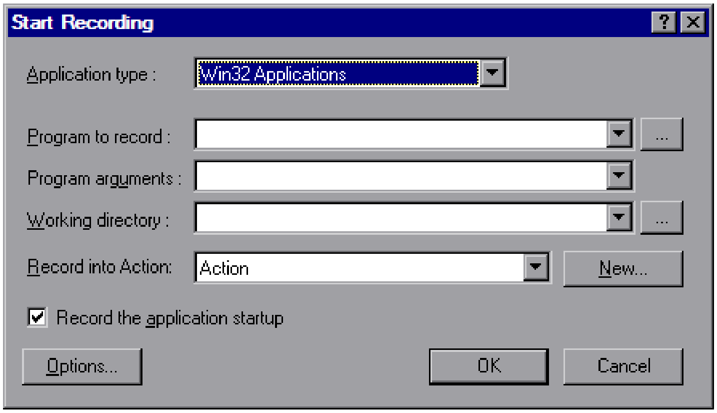
☑ Record the application startup

➤ **Program to record.** Select the browser or Internet application to record.

➤ **URL Address.** Specify the starting URL address.

➤ **Working Directory.** For applications that require you to specify a working directory, specify it here. The required information differs, depending on the type of Vuser script.

5 For Win32 Applications, fill in the relevant information:



Start Recording

Application type : Win32 Applications

Program to record :

Program arguments :

Working directory :

Record into Action: Action

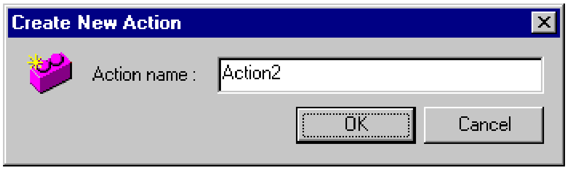
☑ Record the application startup

➤ **Program to record.** Enter the Win 32 application to record.

- **Program Arguments.** Specify command line arguments for the executable specified above. For example, if you specify *plus32.exe* with the command line options peter@neptune, it connects the user *Peter* to the server *Neptune* when starting *plus32.exe*.

- **Working Directory.** For applications that require you to specify a working directory, specify it here.

6 In the **Record into Action** box, select the section into which you want to record. Initially, the available sections are *vuser_init*, *Action*, and *vuser_end*. For single-protocol Vuser scripts that support multiple actions (Oracle NCA, Web, RTE, C Vusers, WAP, i-Mode, and VoiceXML), you can add a new section by selecting **Actions** > **Create New Action** and specify a new action name.



Create New Action

Action name : Action2

7 To record the application startup, select **Record the application startup** (not applicable to Java type Vuser script). To instruct VuGen not to record the application startup, clear the check box. In the following instances, it may not be advisable to record the startup:

- If you are recording multiple actions, in which case you only need to perform the startup in one action.

- In cases where you want to navigate to a specific point in the application before starting to record.

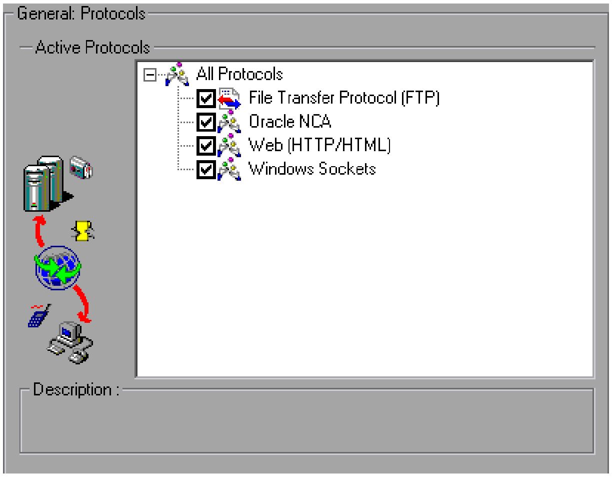
- If you are recording into an existing script.

8 Click **Options** or the **Recording Options** button to open the Recording options dialog box and set the recording options. The available options may vary, depending on the recorded protocol. For more information, see their respective chapters.

9 To select a language for code generation and to set the scripting options, click the **Script** tab. For details, see Chapter 19, "Setting Script Generation Preferences."

10 To specify port information, click the **Port Mapping** tab. This is useful when recording SSL applications on a non-standard port. Review the list of ports. If the port you are using is not on the list, you can specify the information using the Port Mapping options. For more information, see Chapter 21, "Configuring the Port Mappings."

11 For a multi-protocol recording: To modify the list of protocols that you want to record, click the **Protocol** tab. Expand the node and select the desired protocols.



General: Protocols

Active Protocols

- All Protocols
  - ☑ File Transfer Protocol (FTP)
  - ☑ Oracle NCA
  - ☑ Web (HTTP/HTML)
  - ☑ Windows Sockets

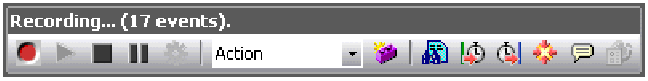
Description :

You are now ready to begin recording.

12 Click **OK** to close the dialog box and begin recording.

13 If you cleared the **Record the application startup** check box, the Recording Suspended dialog box appears. When you reach the point at which you want to start recording, click **Record**. If you decide not to record, click **Abort**.

14 VuGen starts your application and the Recording toolbar opens.



Recording... (17 events).

Action

Perform typical actions within your application. VuGen simultaneously fills in the selected action section of the Vuser script. Use the floating toolbar to switch between sections during recording.

If your application or server requires authentication, VuGen will prompt you to enter a user name and password. For more information about authentication, see the appropriate section.

Ending and Saving a Recording Session

After you record a typical business process, you complete the recording session by performing the closing steps of your business process and saving the Vuser script.

To complete the recording:

1 Switch to the *vuser_end* section in the floating toolbar, and perform the log off or cleanup procedure.

2 Click the **Stop Recording** button on the Recording toolbar. The VuGen editor displays all the recorded steps, (or the recorded functions if you began in script view).

3 Click **Save** to save the recorded session. The Save Test dialog box opens (for new Vuser scripts only). Specify a script name. **Note:** Do not name the script *init*, *run* or *end*, since these names are used by VuGen.

4 To save the entire script directory as a zip file, select **File** > **Zip Operations** > **Export to Zip File**.

Specify which files to save. To save only runtime files, select **Runtime files** in the **Files to zip** section. By default, VuGen saves all files to the archive.

Select a **compression ratio**: maximum, normal, fast, super fast, or none. The greater the compression ratio, the longer VuGen will take to create the archive.

Click **OK**.

Tip: You can save the script as a User-Defined Template. For details, see "User-Defined Template" on page 89.

5 To create a zip file and send it as an email attachment, select **File** > **Zip Operations** > **Zip and Email**.

Click **OK**. An email compose form opens.

Enter an email address and send your email.

Viewing the Recording Logs

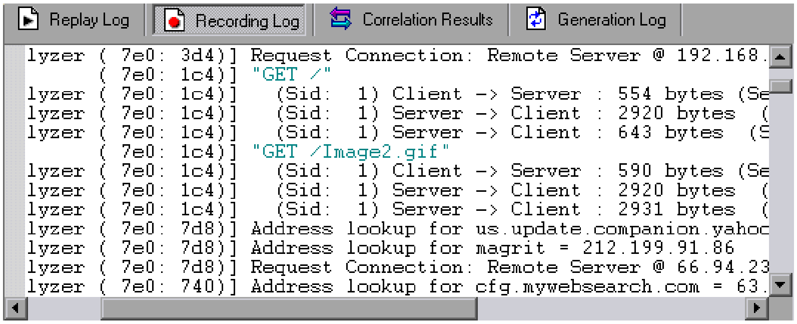
After recording, you can view the contents of the *vuser_init*, *Actions*, and *vuser_end* sections in the VuGen script editor. To display an action, select the action name in the left pane.

While you record, VuGen creates a series of configuration, data, and source code files. These files contain Vuser run-time and setup information. VuGen saves these files together with the script. To open the script folder, switch to Script view (**View** > **Script View**) and select **Open Script Directory** from the right-click menu.

You can view information about the recording and the script generation by viewing the logs in the bottom window. To open the Output window, select **View** > **Output Window** and select the Recording Log or Generation Log tabs.

Recording Log

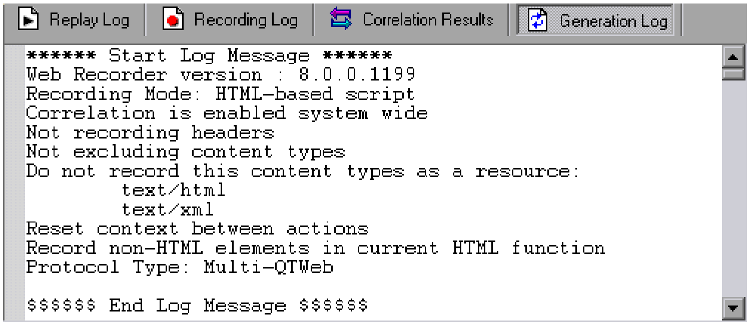
To view a log of the messages that were issued during recording, click the **Recording Log** tab. You can set the level of detail for this log in the **Advanced** tab of the Recording options.



```
lyzer ( 7e0: 3d4)] Request Connection: Remote Server @ 192.168.
      ( 7e0: 1c4)] "GET /"
lyzer ( 7e0: 1c4)]   (Sid:  1) Client -> Server : 554 bytes (Se
lyzer ( 7e0: 1c4)]   (Sid:  1) Server -> Client : 2920 bytes  (
lyzer ( 7e0: 1c4)]   (Sid:  1) Server -> Client : 643 bytes  (S
      ( 7e0: 1c4)] "GET /Image2.gif"
lyzer ( 7e0: 1c4)]   (Sid:  1) Client -> Server : 590 bytes (Se
lyzer ( 7e0: 1c4)]   (Sid:  1) Server -> Client : 2920 bytes  (
lyzer ( 7e0: 1c4)]   (Sid:  1) Server -> Client : 2931 bytes  (
lyzer ( 7e0: 7d8)] Address lookup for us.update.companion.yahoc
lyzer ( 7e0: 7d8)] Address lookup for magrit = 212.199.91.86
lyzer ( 7e0: 7d8)] Request Connection: Remote Server @ 66.94.23
lyzer ( 7e0: 740)] Address lookup for cfg.mywebsearch.com = 63.
```

Generation Log

To view a summary of the script's settings used for generating the code, select the **Generation Log** tab. This view shows the recorder version, the recording option values, and other additional information.



```
****** Start Log Message ******
Web Recorder version : 8.0.0.1199
Recording Mode: HTML-based script
Correlation is enabled system wide
Not recording headers
Not excluding content types
Do not record this content types as a resource:
        text/html
        text/xml
Reset context between actions
Record non-HTML elements in current HTML function
Protocol Type: Multi-QTWeb

$$$$$$ End Log Message $$$$$$
```

Using Zip Files

VuGen allows you to work with zip file in several ways. The advantages of working with zip files is that you conserve disk space, and it allows your scripts to be portable. Instead of copying many files from machine to machine, you only need to copy one zip file.

Importing from a Zip file

To open a script stored in a compressed zip file, select **File** > **Zip Operations** > **Import from Zip File**. After you select a zip file, VuGen prompts you for a location at which to store the unzipped files.

Working from a Zip file

To work from a zip file, while not expanding or saving the script files, select **File** > **Zip Operations** > **Work from Zip File**. When you modify the script and save it, the changes are stored directly in the zip file.

Exporting to a Zip File

To save the entire script directory as a zip file, select **File** > **Zip Operations** > **Export to Zip File**.

You can indicate whether to save all files of only runtime. By default, VuGen saves all files to the archive. To save only runtime files, select the **Runtime files** option.

You can also select a **compression ratio**: Maximum, Normal, Fast, Super fast, or None. The greater the compression ratio, the longer VuGen takes to create the archive. The *Maximum* compression option, therefore, is the slowest.

Zip and Email

To create a zip file and send it as an email attachment, select **File** > **Zip Operations** > **Zip and Email**. When you click **OK** in the Zip To File dialog box, VuGen compresses the file according to your settings and opens an email compose form with the zip file as an attachment.

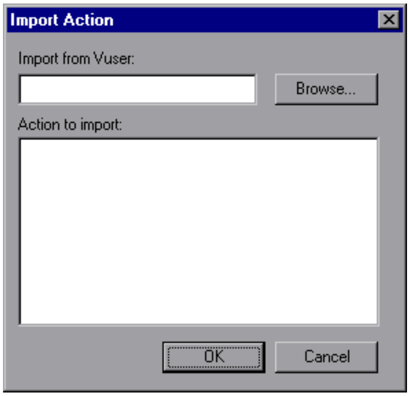
Importing Actions

For Vuser types that support multiple actions, you can import actions into your script from another Vuser script. You can only import actions from Vusers of the same type. Note that any parameters associated with the imported action, will be merged with the script. The available options are:

- **Import From Vuser.** Enter or Browse for the Vuser script from which you want to import.
- **Action to Import.** Select the Action you want to import.

To import actions into the current script:

1 Select **Actions** > **Import Action into Vuser**, or right-click the Task pane and select **Import Action into Vuser** from the right-click menu. The Import Action dialog box opens.



2 Click **Browse** to select a Vuser script. A list of the script's actions appears in the **Actions to Import** section.

3 Highlight an action and click **OK**. The action appears in your script.

4 To rearrange the order of actions, you must first enable action reordering. Right-click on any action and select **Enable Action Reorder**. Then drag the actions to the desired order. Note that when you reorder actions in the left pane of VuGen, it does not affect the order in which they are executed. To change the order of execution, use the Pacing node of the Run-Time settings as described in Chapter 22, "Configuring Run-Time Settings."

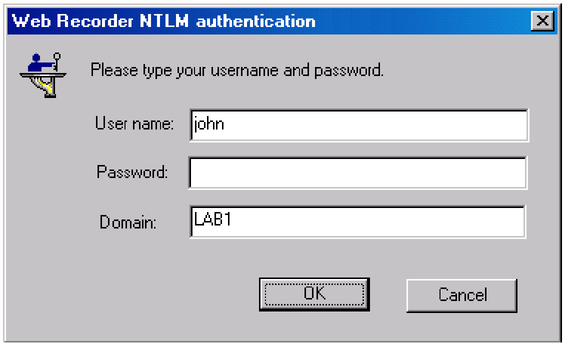
Providing Authentication Information

The following section only applies to multi-protocol recording.

When recording a Web session that uses NTLM authentication, your server may require you to enter details such as a user name and password.

Initially, IE (Internet Explorer) tries to use the NT authentication information of the current user:

➤ If IE succeeds in logging in using this information and you record a script —then, at the end of the recording VuGen prompts you to enter a password. VuGen retrieves the user name and domain information automatically. If necessary, you can also edit the user name in the Web Recorder NTLM Authentication dialog box.



Web Recorder NTLM authentication

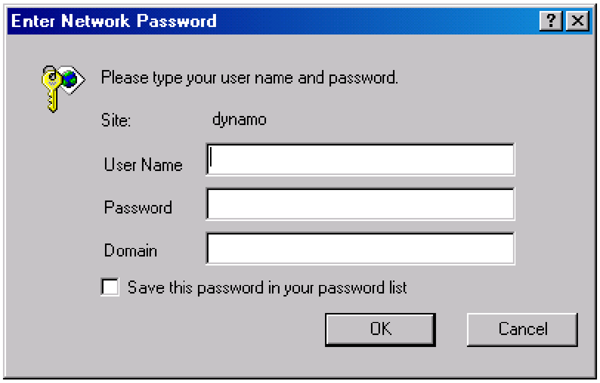
Please type your username and password.

User name: john

Password:

Domain: LAB1

➤ If IE is unable to log in with the current user's information, it prompts you to enter a user name and password using the standard browser authentication dialog box.



Enter Network Password

Please type your user name and password.

Site: dynamo

User Name

Password

Domain

□ Save this password in your password list

Generating a web_set_user function

When performing NTLM authentication, VuGen adds a **web_set_user** function to the script.

- If the authentication succeeds, VuGen generates a **web_set_user** function with your user name, encrypted password, and host.

```
web_set_user("domain1\\dashwood",
        lr_decrypt("4042e3e7c8bbbcfde0f737f91f"),
       "sussex:8080");
```

- If you cancel the Web Recorder NTLM Authentication dialog box without entering information, VuGen generates a **web_set_user** function for you to edit manually.

```
web_set_user("domain1\\dashwood,
        "Enter NTLM Password Here",
        "sussex:8080");
```

Note: If you enter a password manually, it will appear in the script as-is, presenting a security issue. To encrypt the password, right-click the password and select **Encrypt string**. VuGen encrypts the string and generates an **lr_decrypt** function, used to decode the password during replay. For more information about encrypting strings, see "Encrypting Text" on page 124.

Regenerating a Vuser Script

After recording a script, you can enhance it by adding transactions, rendezvous, messages, or comments. For more information, see Chapter 6, "Enhancing Vuser Scripts."

In addition, you can parameterize the script and correlate variables. For more information, see Chapter 13, "Working with VuGen Parameters."

If you need to revert back to your originally recorded script, you can regenerate it. This feature is ideal for debugging, or fixing a corrupted script. When you regenerate a script, it removes all of the manually added enhancements to the recorded actions. If you added parameters to your script, VuGen restores the original values. The parameter list, however, is not deleted; you can reinsert parameters that you created earlier. Note that regeneration, only cleans up the recorded actions, but not those that were manually added.

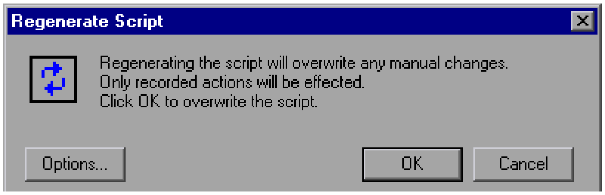
The following buttons are available from the Regenerate Script dialog box.

OK: Regenerates the Vuser script from the original Recording log. Regeneration removes all correlations and parameterizations that you performed on the script manually.

Options: When working with multi-protocol scripts, you can indicate which protocols to regenerate. To customize the regeneration, click the **Options** button in the Regenerate Vuser dialog box to open the recording options. Select the **Protocols** tab and indicate which protocols to regenerate and which to leave as is. Select the check boxes of the protocols you want to regenerate. Clear the check boxes of those you do not wish to regenerate.

To regenerate a multi-protocol Vuser script:

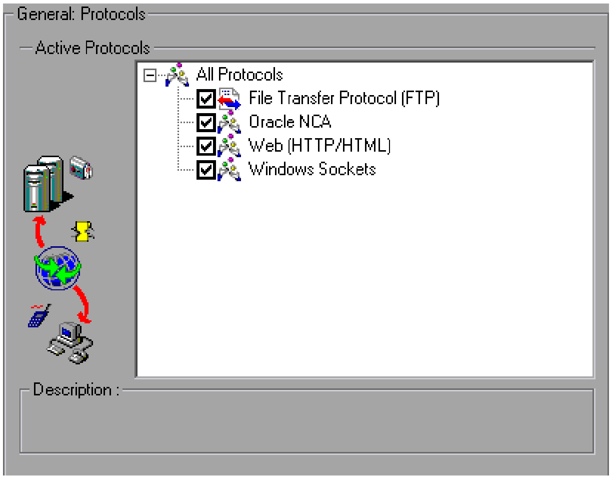
1 Select **Tools** > **Regenerate Script**. VuGen issues a warning indicating that all manual changes will be overwritten.



Regenerating the script will overwrite any manual changes.
Only recorded actions will be effected.
Click OK to overwrite the script.

2 Click **Options** to open the Regenerate Options dialog box.

3 Select the **General:Protocols** node. Indicate which protocols to regenerate and which to leave as is. Select the check boxes of the protocols you want to regenerate. Clear the check boxes of the protocols you want to leave unchanged.



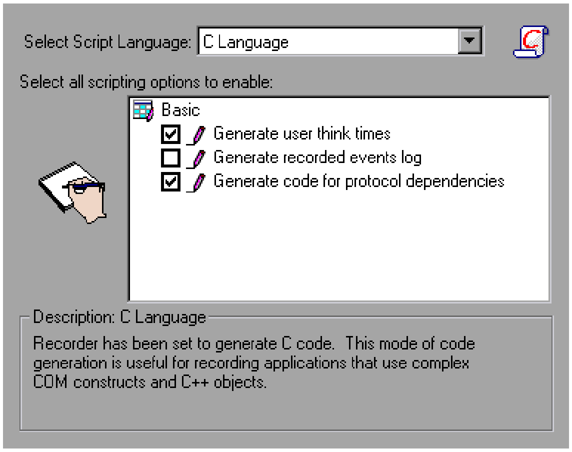
General: Protocols

Active Protocols

- All Protocols
  - ☑ File Transfer Protocol (FTP)
  - ☑ Oracle NCA
  - ☑ Web (HTTP/HTML)
  - ☑ Windows Sockets

Description :

4 To change the Script options, select the **General:Script** node and select or clear the appropriate check box.



Select Script Language: C Language

Select all scripting options to enable:

- Basic
  - ☑ Generate user think times
  - □ Generate recorded events log
  - ☑ Generate code for protocol dependencies

Description: C Language

Recorder has been set to generate C code. This mode of code generation is useful for recording applications that use complex COM constructs and C++ objects.

6

Enhancing Vuser Scripts

You can enhance a Vuser script—either during or after recording—by adding General Vuser functions, Protocol-Specific Vuser functions, and Standard ANSI C functions.

This chapter includes:

- About Enhancing Vuser Scripts on page 108
- Inserting Transactions into a Vuser Script on page 110
- Inserting Rendezvous Points (LoadRunner only) on page 112
- Inserting Comments into a Vuser Script on page 114
- Obtaining Vuser Information on page 115
- Sending Messages to Output on page 116
- Handling Errors in Vuser Scripts During Execution on page 120
- Synchronizing Vuser Scripts on page 121
- Emulating User Think Time on page 122
- Handling Command Line Arguments on page 123
- Encrypting Text on page 124
- Encoding Passwords Manually on page 125
- Adding Files to the Script on page 126

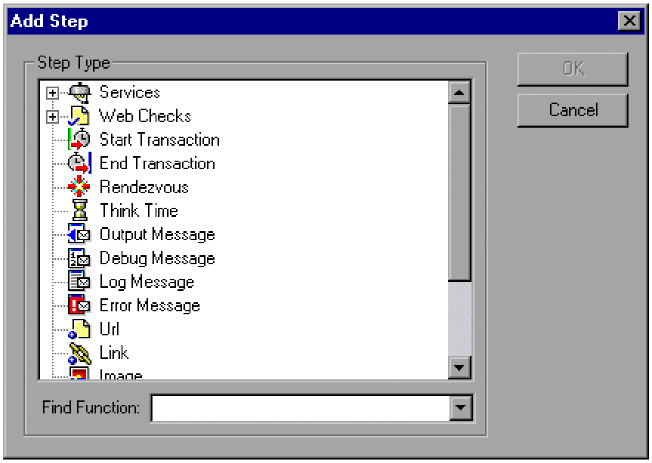
About Enhancing Vuser Scripts

While you are recording a Vuser script, or after you record it, you can enhance its capabilities by manually adding a step, also known as a function.

To add a new step to your script.

1 Place the cursor at the desired location.

2 Select **Insert** > **New Step**. The Add Step dialog box opens with the relevant steps for the current protocol.



3 Select a step and click **OK**. VuGen inserts the step (or function in Script view) at the location of the cursor.

The following types of functions are available from the Add Steps dialog box:

- General Vuser Functions
- Protocol-Specific Vuser Functions
- Standard ANSI C Functions

General Vuser Functions

General Vuser functions greatly enhance the functionality of any Vuser script. For example, you can use General Vuser functions to measure server performance, control server load, add debugging code, or retrieve run-time information about the Vusers participating in the test.

You can use General Vuser functions in any type of Vuser script. All General Vuser functions have an **LR** prefix. VuGen generates some General Vuser functions and inserts them into a Vuser script during recording. To use additional functions that were not automatically generated, select **Insert** > **New Step** from VuGen's main window and select the desired function.

This chapter discusses the use of only the most common General Vuser functions. For additional information about Vuser functions, see the *Online Function Reference* (**Help** > **Function Reference**).

Protocol-Specific Vuser Functions

There are several libraries of functions that you can use to enhance a Vuser script. Each library is specific to a type of Vuser. For example, you use the **LRS** functions in a Windows Sockets Vuser script and **LRT** functions in a Tuxedo Vuser script. For details on the protocol-specific Vuser functions, see the *Online Function Reference* (**Help** > **Function Reference**).

Standard ANSI C Functions

You can enhance your Vuser scripts by adding standard ANSI C functions. ANSI C functions allow you to add comments, control flow statements, conditional statements, and so forth to your Vuser scripts. You can add standard ANSI C functions to any type of Vuser script. For details, see "Programming a Script in the VuGen Editor" in *Volume II-Protocols*.

Inserting Transactions into a Vuser Script

You define *transactions* to measure the performance of the server. Each transaction measures the time it takes for the server to respond to specified Vuser requests. These requests can be simple tasks such as waiting for a response for a single query, or complex tasks, such as submitting several queries and generating a report.

To measure a transaction, you insert Vuser functions to mark the beginning and the end of a task. Within a script, you can mark an unlimited number of transactions, each transaction with a different name.

For LoadRunner, the Controller measures the time that it takes to perform each transaction. After the test run, you analyze the server's performance per transaction using the Analysis' graphs and reports.

You can create transactions either during or after recording. To add transactions after recording, use the Transaction editor to graphically mark the steps of a transaction, as described in "Transactions" on page 72. Alternatively, use the **Insert** menu to add **Start Transaction** and **End Transaction** markers.

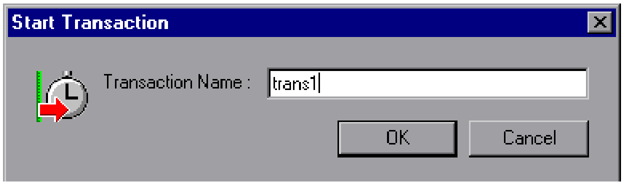
The following sections describe how to create a transaction during recording.

Marking the Beginning of a Transaction

Before creating a script, you should determine which business processes you want to measure. You then mark each business process or sub-process as a transaction.

To mark the start of a transaction:

1 While recording a Vuser script, click the **Start Transaction** button on the Recording toolbar. The Start Transaction dialog box opens.



Start Transaction

Transaction Name : trans1

2 Type a transaction name in the Transaction Name box. Transaction names must begin with a letter or number and may contain letters or numbers.

Note: Do not use the following characters: . , : # / \ " <

Click OK to accept the transaction name. VuGen inserts an lr_start_transaction statement into the Vuser script. For example, the following function indicates the start of the trans1 transaction:

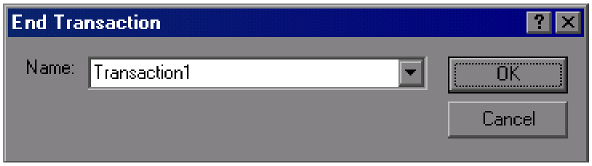
```
lr_start_transaction("trans1");
```

Marking the End of a Transaction

You mark the end of a business process with an end transaction statement.

To mark the end of a transaction:

1 While recording a script, click the **End Transaction** button on the Recording toolbar. The End Transaction dialog box opens.



End Transaction

Name: Transaction1

2 Click the arrow for a list of open transactions. Select the transaction to close.

Click OK to accept the transaction name. VuGen inserts an **lr_end_transaction** statement into the Vuser script. For example, the following function indicates the end of the trans1 transaction:

```
lr_end_transaction("trans1", LR_AUTO);
```

Note: You can create *nested* transactions—transactions within transactions. If you nest transactions, close the inner transactions before closing the outer ones—otherwise the transactions won’t be analyzed properly.

Inserting Rendezvous Points (LoadRunner only)

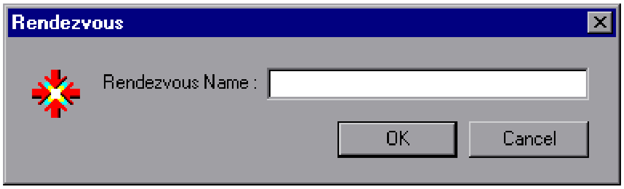
When performing load testing, you need to emulate heavy user load on your system. To accomplish this, you synchronize Vusers to perform a task at exactly the same moment. You configure multiple Vusers to act simultaneously by creating a *rendezvous point*. When a Vuser arrives at the rendezvous point, it waits until all Vusers participating in the rendezvous arrive. When the designated number of Vusers arrive, the Vusers are released.

You designate the meeting place by inserting a rendezvous point into your Vuser script. When a Vuser executes a script and encounters the rendezvous point, script execution is paused and the Vuser waits for permission from the Controller to continue. After the Vuser is released from the rendezvous, it performs the next task in the script.

Note: Rendezvous points are only effective in *Action* section(s)—not *init* or *end*.

To insert a rendezvous point:

1 While recording a Vuser script, click the **Rendezvous** button on the Recording toolbar. The Rendezvous dialog box opens.



2 Type a name for the rendezvous point in the **Rendezvous Name** box.

Note: The name for the rendezvous point is not case sensitive. For example, the Vuser recognizes Rendezvous1 and rendezvous1 as the same point.

Click OK. VuGen inserts lr_rendezvous into the Vuser script. For example, the following function defines a rendezvous point named rendezvous1:

```
lr_rendezvous("rendezvous1");
```

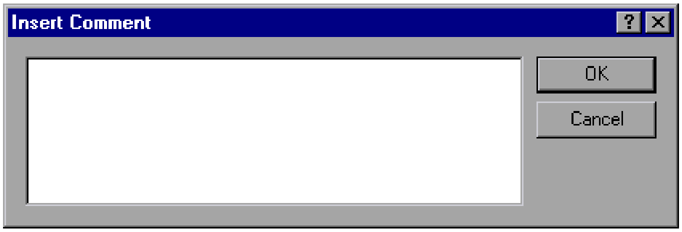
3 To insert rendezvous points into your script after the recording session, select **Insert** > **Rendezvous** from the VuGen menu.

Inserting Comments into a Vuser Script

VuGen allows you to insert comments between Vuser activities. You can insert a comment to describe an activity or to provide information about a specific operation. For example, if you are recording database actions, you could insert a comment to mark the first query, such as "This is the first query."

To insert a comment:

1 While recording a script, click the **Comment** button on the Recording tool bar. The Insert Comment dialog box opens.



2 Type the comment into the text box.

3 Click OK to insert the comment and close the dialog box. The text is placed at the current point in the script, enclosed by comment markers. The following script segment shows how a comment appears in a Vuser script:

```
/*
* This is the first query
*/
```

Note: You can insert comments into your script after you complete a recording session, by selecting **Insert** > **Comment** from the VuGen menu.

Obtaining Vuser Information

You can add the following functions to your Vuser scripts to retrieve Vuser information:

Function	Description
lr_get_attrib_string	Returns a command line parameter string.
lr_get_host_name	Returns the name of the machine running the Vuser script.
lr_get_master_host_name	Returns the name of the machine running the Controller. Not applicable when working with the HP Business Availability Center.
lr_whoami	Returns the name of a Vuser executing the script. Not applicable when working with the HP Business Availability Center.

In the following example, the **lr_get_host_name** function retrieves the name of the computer on which the Vuser is running.

```
my_host = lr_get_host_name( );
```

For more information about the above functions, see the *Online Function Reference* (**Help** > **Function Reference**).

Sending Messages to Output

Using the Message type functions in your Vuser script, you can send customized error and notification messages to the output and log files, and to the Test Report summary. For example, you could insert a message that displays the current state of the client application. The LoadRunner Controller displays these messages in the Output window. You can also save these messages to a file.

When working with HP Business Availability Center, you can use Message type functions to send error and notification messages to the Web site or Business Process Monitor log files. For example, you could insert a message that displays the current state of the Web-based application.

Note: Do not send messages from within a transaction as this may lengthen the transaction execution time and skew the transaction results.

You can use the following message functions in your Vuser scripts:

Funtion	Description
lr_debug_message	Sends a debug message to the Output window or the Business Process Monitor log file.
lr_error_message	Sends an error message to the Output window, Test Results report, or the Business Process Monitor log files.
lr_get_debug_message	Retrieves the current message class.
lr_log_message	Sends an output message directly to the log file, *output.txt*, located in the Vuser script directory. This function is useful in preventing output messages from interfering with TCP/IP traffic.
lr_output_message	Sends a message to the Output window, Test Results report, or the Business Process Monitor log files.

Funtion	Description
lr_set_debug_message	Sets a message class for output messages.
lr_vuser_status_message	Sends a message to the Vuser status area in the Controller. Not applicable when working with the HP Business Availability Center.
lr_message	Sends a message to the Vuser log and Output window or the Business Process Monitor log files.

The behavior of the **lr_message**, **lr_output_message**, and **lr_log_message** functions are not affected by the script's debugging level in the Log run-time settings—they will always send messages.

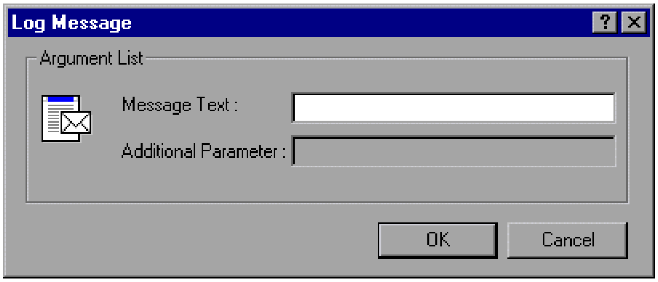
Using the **lr_output_message**, and **lr_error_message** functions, you can also send meaningful messages to the Test Results summary report. For information, see Chapter 10, "Viewing Test Results."

Log Messages

You can use VuGen to generate and insert **lr_log_message** functions into a Vuser script. For example, if you are recording database actions, you could insert a message to indicate the first query, "This is the first query."

To insert an lr_log_message function:

1 Select **Insert** > **Log Message**. The Log Message dialog box opens.



2 Type the message into the **Message Text** box.

3 Click **OK** to insert the message and close the dialog box. An **lr_log_message** function is inserted at the current point in the script.

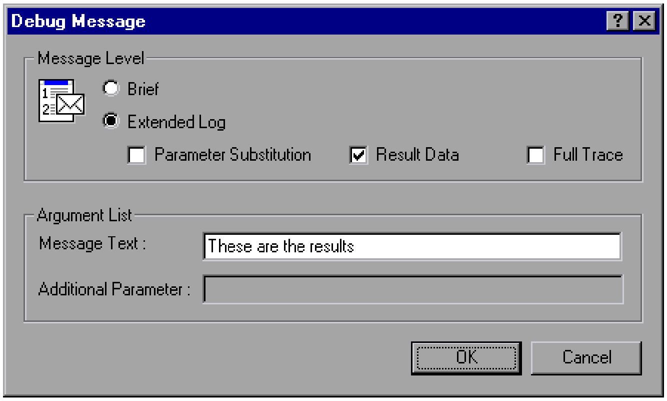
Debug Messages

You can add a debug or error message using VuGen's user interface. For debug messages you can indicate the level of the text message—the message is only issued when your specified level matches the message class. You set the message class using **lr_set_debug_message**.

To insert a debug function:

1 Select **Insert** > **New Step**. The Add Step dialog box opens.

2 Select the **Debug Message** step and click **OK**. The Debug Message dialog box opens.



Debug Message

Message Level

○ Brief

● Extended Log

□ Parameter Substitution ☑ Result Data □ Full Trace

Argument List

Message Text : These are the results

Additional Parameter :

3 Select a message level, **Brief** or **Extended Log**. If you select Extended Log, indicate the type of information to log: **Parameter Substitution**, **Result Data**, or **Full Trace**.

4 Type the message into the **Message Text** box.

5 Click **OK** to insert the message and close the dialog box. An **lr_debug_message** function is inserted at the current point in the script.

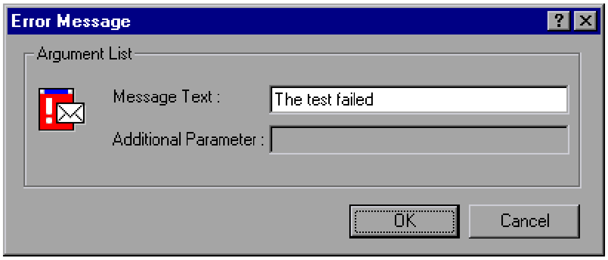
Error and Output Messages

For protocols with a Tree view representation of the script, such as Web, Winsock, and Oracle NCA, you can add an error or output message using the user interface. A common usage of this function is to insert a conditional statement, and issue a message if the error condition is detected.

To insert an error or output message function:

1 Select **Insert** > **New Step**. The Add Step dialog box opens.

2 Select the **Error Message** or **Output Message** step and click **OK**. The Error Message or Output Message dialog box opens.



Error Message

Argument List

Message Text : The test failed

Additional Parameter :

3 Type the message into the **Message Text** box.

4 Click **OK** to insert the message and close the dialog box. An **lr_error_message** or **lr_output_message** function is inserted at the current point in the script.

For more information about the message functions, see the *Online Function Reference* (**Help** > **Function Reference**).

Handling Errors in Vuser Scripts During Execution

You can specify how a Vuser handles errors during script execution. By default, when a Vuser detects an error, the Vuser stops executing the script. You can instruct a Vuser to continue with the next iteration when an error occurs using one of the following methods:

- Using run-time settings. You can specify the **Continue on Error** run-time setting. The **Continue on Error** run-time setting applies to the entire Vuser script. You can use the **lr_continue_on_error** function to override the **Continue on Error** run-time setting for a portion of a script. For details, see "Error Handling" on page 433.
- Using the **lr_continue_on_error** function. The **lr_continue_on_error** function enables you to control error handling for a specific segment of a Vuser script. To mark the segment, enclose it with lr_continue_on_error(1); and lr_continue_on_error(0); statements. The new error settings apply to the enclosed Vuser script segment. See the paragraphs below for details.

For example, if you enable the Continue on Error run-time setting and a Vuser encounters an error during replay of the following script segment, the Vuser continues executing the script.

```
web_link("EBOOKS",
    "Text=EBOOKS",
    "Snapshot=t2.inf",
    LAST);

web_link("Find Rocket eBooks",
    "Text=Find Rocket eBooks",
    "Snapshot=t3.inf",
    LAST);
```

To instruct the Vuser to continue on error for a specific segment of the script, enclose the segment with the appropriate lr_continue_on_error statements:

```
lr_continue_on_error(1);
    web_link("EBOOKS",
        "Text=EBOOKS",
        "Snapshot=t2.inf",
        LAST);

    web_link("Find Rocket eBooks",
        "Text=Find Rocket eBooks",
        "Snapshot=t3.inf",
        LAST);
lr_continue_on_error(0);
```

Synchronizing Vuser Scripts

You can add synchronization functions to synchronize the execution of the Vuser script with the output from your application. Synchronization applies to RTE Vuser scripts only.

The following is a list of the available synchronization functions:

Function	Description
TE_wait_cursor	Waits for the cursor to appear at a specified location in the terminal window.
TE_wait_silent	Waits for the client application to be silent for a specified number of seconds.
TE_wait_sync	Waits for the system to return from X-SYSTEM or Input Inhibited mode.
TE_wait_text	Waits for a string to appear in a designated location.
TE_wait_sync_transaction	Records the time that the system remained in the most recent X SYSTEM mode.

For details on using synchronization functions in RTE Vuser scripts, see "RTE - Synchronization" in *Volume II-Protocols*.

Emulating User Think Time

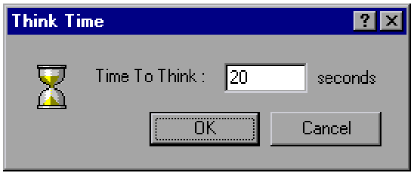
The time that a user waits between performing successive actions is known as the *think time*. Vusers use the **lr_think_time** function to emulate user think time. When your record a Vuser script, VuGen records the actual think times and inserts appropriate **lr_think_time** statements into the Vuser script. You can edit the recorded **lr_think_time** statements, and manually add more **lr_think_time** statements to a Vuser script.

To manually add a think time statement:

1 Place the cursor at the desired location.

2 Select **Insert** > **Add Step**. The Add Step dialog box opens.

3 Select **Think Time** and click **OK**. The Think Time dialog box opens.



Time To Think : 20 seconds

4 Specify the desired think time in seconds and click **OK**.

Note: When you record a Java Vuser script, **lr_think_time** statements are not generated in the Vuser script.

You can use the think time settings to influence how the **lr_think_time** statements operate when you execute a Vuser script. To access the think time settings, select **Vuser** > **Run-time Settings** from the VuGen main menu, and then click the **Think Time** tab. For more information, see the *Online Function Reference* (**Help** > **Function Reference**).

Handling Command Line Arguments

You can pass values to a Vuser script at run-time by specifying command line arguments when you run the script. You specify the command line arguments within the Run-Time settings dialog box. For more information, see "Configuring Additional Attributes Run-Time Settings" on page 431.

There are three functions that allow you to read the command line arguments, and then to pass the values to a Vuser script:

Function	Description
lr_get_attrib_double	Retrieves double precision floating point type arguments
lr_get_attrib_long	Retrieves long integer type arguments
lr_get_attrib_string	Retrieves character strings

Your command line should have one of the following two formats where the arguments and their values are listed in pairs, after the script name:

```
script_name   -argument argument_value   -argument argument_value
```

```
script_name   -argument argument_value   -argument argument_value
```

The following example shows the command line string used to repeat script1 five times on the load generator pc4:

```
script1  -host pc4  -loop 5
```

For more information on the command line parsing functions, or for details on including arguments on a command line, see the *Online Function Reference* (**Help** > **Function Reference**).

Encrypting Text

You can encrypt text within your script to protect your passwords and other confidential text strings. You can perform encryption both automatically, from the user interface, and manually, through programming. When you encrypt a string, it appears in the script as a coded string. Note that VuGen uses 32-bit encryption.

In order for the script to use the encrypted string, it must be decrypted with **lr_decrypt**.

```
lr_start_transaction(lr_decrypt("3c29f4486a595750"));
```

You can restore the string at any time, to determine its original value.

To encrypt a string:

1 For protocols that have tree views, view the script in script view. Select **View** > **Script View**.

2 Select the text you want to encrypt.

3 Select **Encrypt string (***string***)** from the right-click menu.

To restore an encrypted string:

1 For protocols that have tree views, view the script in script view. Select **View** > **Script View**.

2 Select the string you want to restore.

3 Select **Restore encrypted string (***string***)** from the right-click menu.

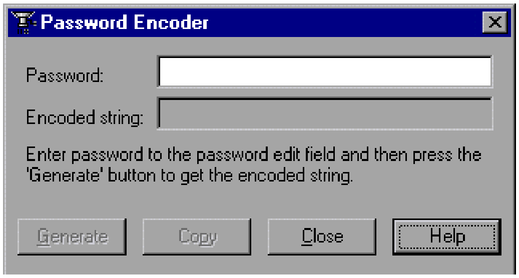
For more information on the **lr_decrypt** function, see the *Online Function Reference* (**Help** > **Function Reference**).

Encoding Passwords Manually

You can encode passwords in order to use the resulting strings as arguments in your script or 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 protect the integrity of the passwords. The **Password Encoder** enables you to encode your passwords and place secure values into the table.

To encode a password:

1 From the Windows menu, select **Start** > **Programs** > **LoadRunner** > **Tools** > **Password Encoder**. The Password Encoder dialog box opens.



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

2 Enter the password in the **Password** box.

3 Click **Generate**. The Password Encoder encrypts the password and displays it in the **Encoded String** box.

4 Use the **Copy** button to copy and paste the encoded value into the Data Table.

5 Repeat the process for each password you want to encode.

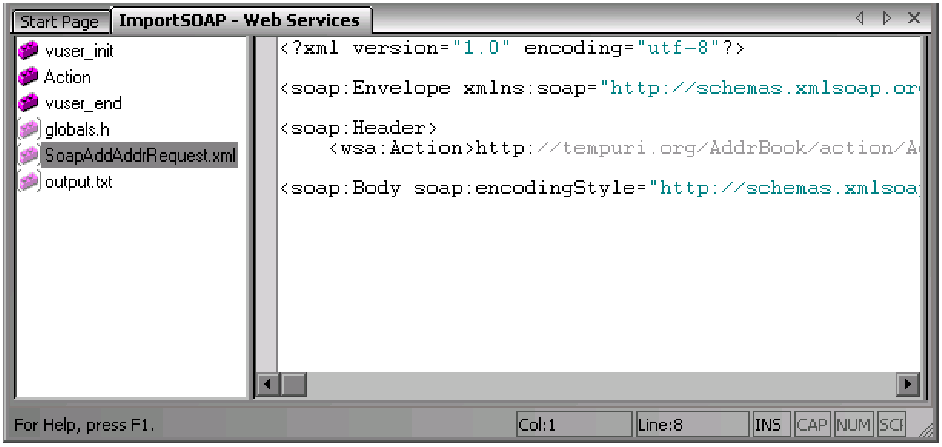
6 Click **Close** to close the Password Encoder.

Adding Files to the Script

You can add files to your script directory to make them available when running the script. If the files are text-based, you will be able to view and edit them in VuGen's editor.

To add a file:

1. Open the script.
2. Select **File** > **Add Files to Script**, or right-click the Task pane and select **Add Files to Script** from the right-click menu.
3. Locate the files and click **Open**. VuGen adds the selected files to the script directory and the VuGen editor.
4. Select the file in the left pane to display its contents.



```
<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.or

<soap:Header>
    <wsa:Action>http://tempuri.org/AddrBook/action/A

<soap:Body soap:encodingStyle="http://schemas.xmlsoa
```

For Help, press F1.

7

Creating Business Process Reports

VuGen lets you create business process reports by exporting information from your script into a Microsoft Word document.

This chapter includes:

- About Exporting a Script to Word on page 127
- Specifying the Report Details on page 128
- Specifying Report Content on page 129

The following information applies to AJAX (Click and Script), Citrix_ICA, Oracle NCA, Oracle Web Applications 11i, PeopleSoft Enterprise, RDP, SAP (Click and Script), SAPGUI, SAP - Web, Web (Click and Script), Web (HTTP/HTML), and Web Services Protocols.

About Exporting a Script to Word

At the final stage of script creation, you can create a report that will describe your business process. VuGen exports the script information to a Microsoft Word document.

You can use a pre-designed template or one provided with VuGen, to create reports with summary information about your test run.

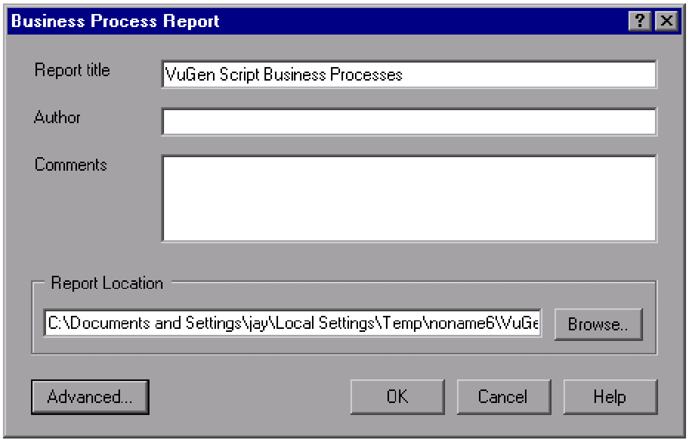
VuGen lets you customize the contents of the report by indicating what type of information you want to include.

Specifying the Report Details

Before you specify the content of the report, you give it a name and a short description. You include any relevant remarks and indicate a location at which to store the report.

To create a Business Process Report:

1 Select **File** > **Create Business Process Report**. The Business Process Report dialog box opens.



Business Process Report

Report title: VuGen Script Business Processes

Author:

Comments:

Report Location: C:\Documents and Settings\jay\Local Settings\Temp\noname6\VuGe

2 Specify a title in the **Report title** box.

3 Enter your name in the **Author** box.

4 Enter any remarks in the **Comment** box.

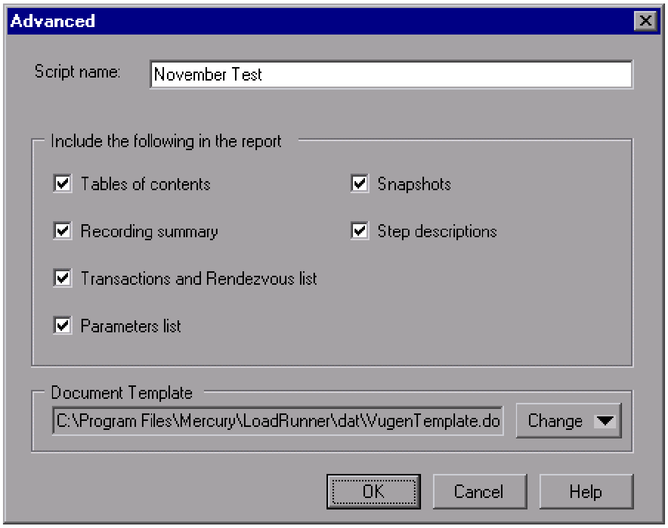
5 Accept the default report location and file name, or browse for the desired path. The default location is that of the saved script, and the default report name is **<script name>_Business_Processes.doc**

6 Click **Advanced** to specify the report's content. For more information about selecting content, see "Specifying Report Content" on page 129.

7 To generate the report, click **OK**.

Specifying Report Content

You can select content for your Business Process Report. By default all the options are enabled. By default the .usr file name appears in the **Script name** box.



Advanced

Script name: November Test

Include the following in the report

☑ Tables of contents
☑ Snapshots
☑ Recording summary
☑ Step descriptions
☑ Transactions and Rendezvous list
☑ Parameters list

Document Template

C:\Program Files\Mercury\LoadRunner\dat\VugenTemplate.do

You can select one or more of the following content options:

- **Table of Contents.** A table of contents indicating the page number of all of the other content in the report. If you disabled an option, it will not appear in the table of contents.
- **Recording Summary.** A summary of the recording session as it appears when you click the Recording Summary link in the Tasks list.
- **Transactions and Rendezvous list.** A comprehensive list of all of the transactions and rendezvous that were defined in the script.
- **Parameter list.** A list of all the parameters that were defined for the script. This list corresponds to the parameters listed in the Parameter List dialog box (**Vuser** > **Parameter List**).

- **Snapshots.** An actual snapshot of the recorded step, adjacent to the step name and description.

Note: Oracle NCA and Web Services reports do not include snapshots.

- **Step descriptions.** A short description of each step listed in the Tree view.
- **Document Template.** The path and file name of the template to use for the report. The default template is stored in the product's **dat** folder.

To change the report template select **change** and specify new template with a .doc extension. If you want to create a new template, we recommend that you use an existing template as a basis for the new one. This will make sure that the required bookmarks and styles are maintained within the new template.

8

Correlating Statements

You can optimize Vuser scripts by correlating statements. VuGen's Correlated Query feature allows you to link statements by using the results of one statement as input for another.

This chapter includes:

- ➤ About Correlating Statements on page 131
- ➤ Using Correlation Functions for C Vusers on page 133
- ➤ Using Correlation Functions for Java Vusers on page 135
- ➤ Comparing Vuser Scripts using WDiff on page 136
- ➤ Modifying Saved Parameters on page 138

About Correlating Statements

The primary reasons for correlating statements are:

To simplify or optimize your code

For example, if you perform a series of dependent queries one after another, your code may become very long. To reduce the size of the code, you can nest the queries, but then you lose precision and the code becomes complex and difficult to understand. Correlating the statements enables you to link queries without nesting.

To generate dynamic data

Many applications and Web sites identify a session by the current date and time. If you try to replay a script, it will fail because the current time is different than the recorded time. Correlating the data enables you to save the dynamic data and use it throughout the scenario run.

To accommodate unique data records

Certain applications (for example databases) require the use of unique values. A value that was unique during recording is no longer unique for script execution. For example, suppose you record the process of opening a new bank account. Each new account is assigned a unique number which is unknown to the user. This account number is inserted into a table with a unique key constraint during recording. If you try to run the script as recorded, it tries to create an account with the recorded number, rather than a new unique number. An error will result because the account number already exists.

If you encounter an error when running your script, examine the script at the point where the error occurred. In many cases, a correlated query will solve the problem by enabling you to use the results of one statement as input to another.

The main steps in correlating a script are:

1. **Determine which value to correlate.**

   For most protocols, you can view the problematic statements in the Execution log. You double-click an error message and jump directly to its location.

   Alternatively, you can use the *WDiff* utility distributed with VuGen to determine the inconsistencies within your script. For more information, see "Comparing Vuser Scripts using WDiff" on page 136.

2 Save the results.

You save the value of a query to a variable using the appropriate function. The correlating functions are protocol-specific. Correlation function names usually contain the string *save_param*, such as **web_reg_save_param** and **lrs_save_param**. See the specific protocol chapters for an explanation on how to perform correlation. In several protocols, such as database and Web, VuGen automatically inserts the functions into your script.

3 Reference the saved values.

Replace the constants in the query or statement with the saved variables.

Several protocols have built-in automatic or partially automated correlation:

- For Java language Vusers, see "Java - Correlating" in *Volume II-Protocols*.
- For Database Vusers, see "Database - Script Correlation" in *Volume II-Protocols*.
- For Web Vusers, see "Web (HTTP/HTML) Correlation Rules" in *Volume II-Protocols*.
- For COM Vusers, see "COM - Understanding and Correlating" in *Volume II-Protocols*.

Using Correlation Functions for C Vusers

To correlate statements for protocols that do not have specific functions, you can use the C Vuser correlation functions. These functions can be used for all C-type Vusers, to save a string to a parameter and retrieve it when required. For similar functions for Java Vusers, see "Using Correlation Functions for Java Vusers" on page 135.

lr_eval_string	Replaces all occurrences of a parameter with its current value.
lr_save_string	Saves a null-terminated string to a parameter.
lr_save_var	Saves a variable length string to a parameter.

For additional information about the syntax of these functions, see the *Online Function Reference*.

Using lr_eval_string

In the following example, lr_eval_string replaces the parameter row_cnt with its current value. This value is sent to the Output window using lr_output_message.

```
lrd_stmt(Csr1, "select count(*) from employee", -1, 1 /*Deferred*/, …);
lrd_bind_col(Csr1, 1, &COUNT_D1, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_save_col(Csr1, 1, 1, 0, "row_cnt");
lrd_fetch(Csr1, 1, 1, 0, PrintRow2, 0);
lr_output_message("value: %s", lr_eval_string("The row count is: <row_cnt>"));
```

Using lr_save_string

To save a NULL terminated string to a parameter, use **lr_save_string**. To save a variable length string, use **lr_save_var** and specify the length of the string to save.

In the following example, lr_save_string assigns 777 to a parameter emp_id. This parameter is then used in another query or for further processing.

```
lrd_stmt(Csr1, "select id from employees where name='John'",…);
lrd_bind_col(Csr1,1,&ID_D1,...);
lrd_exec(Csr1, ...);
lrd_fetch(Csr1, 1, ...);
/* GRID showing returned value "777" */
lr_save_string("777", "emp_id");
```

Using Correlation Functions for Java Vusers

To correlate statements for Java Vusers, you can use the Java Vuser correlation functions. These functions may be used for all Java type Vusers, to save a string to a parameter and retrieve it when required.

lr.eval_string	Replaces a parameter with its current value.
lr.eval_data	Replaces a parameter with a byte value.
lr.eval_int	Replaces a parameter with an integer value.
lr.eval_string	Replaces a parameter with a string.
lr.save_data	Saves a byte as a parameter.
lr.save_int	Saves an integer as a parameter.
lr.save_string	Saves a null-terminated string to a parameter.

When recording a CORBA or RMI session, VuGen performs correlation internally. For more information, see "Java - Correlating" in *Volume II-Protocols*.

Using the Java String Functions

When programming Java Vuser scripts, you can use the Java Vuser string functions to correlate your scripts.

In the following example, lr.eval_int substitutes the variable ID_num with its value, defined at an earlier point in the script.

```
lr.message(" Track Stock: " + lr.eval_int(ID_num));
```

In the following example, lr.save_string assigns John Doe to the parameter Student. This parameter is then used in an output message.

```
lr.save_string("John Doe", "Student");
// ...
lr.message("Get report card for " + lr.eval_string("<Student>"));
classroom.getReportCard
```

Comparing Vuser Scripts using WDiff

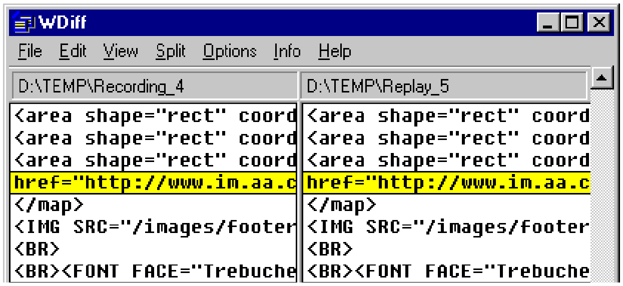
A useful tool in determining which values to correlate is *WDiff*. This utility lets you compare recorded scripts and results to determine which values need to be correlated.

If you are working with other protocols, you can view the Execution log to determine where the script failed and then use the *WDiff* utility to assist you in locating the values that need to be correlated.

To use *WDiff* effectively, you record the identical operation twice, and compare the scripts (or data files for Tuxedo, WinSock, and Jolt). WDiff displays differences in yellow. Note that not all differences indicate a value to correlate. For example, certain receive buffers that indicate the time of execution do not require correlation.

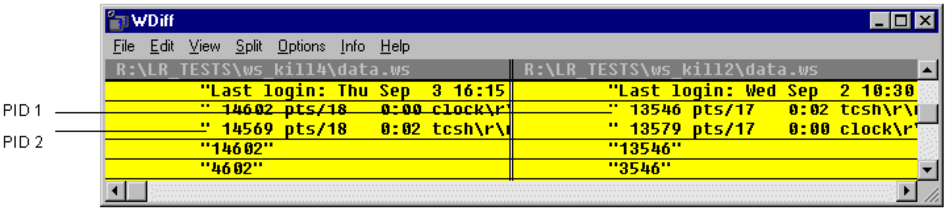
To search for correlations using WDiff:

1. Record a script and save it.
2. Create a new script and record the identical operations. Save the script.
3. Select the section you want to compare (*Actions*, *data.ws*, and so forth).
4. Select **Tools** > **Compare with Vuser**. The Open Test box opens.
5. Specify a Vuser script for comparison (other than the one in the current VuGen window) and click **OK**. WDiff opens and the differences between the Vuser scripts are highlighted in yellow.



```
D:\TEMP\Recording_4            D:\TEMP\Replay_5
<area shape="rect" coord       <area shape="rect" coord
<area shape="rect" coord       <area shape="rect" coord
<area shape="rect" coord       <area shape="rect" coord
href="http://www.im.aa.c       href="http://www.im.aa.c
</map>                         </map>
<IMG SRC="/images/footer       <IMG SRC="/images/footer
<BR>                           <BR>
<BR><FONT FACE="Trebuche       <BR><FONT FACE="Trebuche
```

6 To display the differences only, double-click in the WDiff window.



```
R:\LR_TESTS\ws_kill4\data.ws              R:\LR_TESTS\ws_kill2\data.ws
        "Last login: Thu Sep  3 16:15             "Last login: Wed Sep  2 10:30
        " 14602 pts/18    0:00 clock\r            " 13546 pts/17    0:02 tcsh\r\
        " 14569 pts/18    0:02 tcsh\r\            " 13579 pts/17    0:00 clock\r
        "14602"                                   "13546"
        "4602"                                    "3546"
```

7 Determine which values need to be correlated.

Note that in the above example, WDiff is comparing the *data.ws* from two Winsock Vuser scripts. In this instance, the value to be correlated is the PID for the *clock* processes, which differs between the two recordings.

Note: *WDiff* is the default utility, but you can specify a custom comparison tool. For more information, see "Comparison Tool" on page 37

To continue with correlation, see the appropriate section:

- For Java language Vusers, see "Java - Correlating" in *Volume II-Protocols*.
- For Database Vusers, see "Database - Script Correlation" in *Volume II-Protocols*.
- For Web Vusers, see "Web (HTTP/HTML) Correlation Rules" in *Volume II-Protocols*.
- For COM Vusers, see "COM - Understanding and Correlating" in *Volume II-Protocols*.
- For Tuxedo Vusers, see "Tuxedo Protocols" in *Volume II-Protocols*.
- For WinSock Vusers, see "Windows Sockets (WinSock) Protocol" in *Volume II-Protocols*.

Modifying Saved Parameters

After you save a value to a parameter, you may need to modify it before using it in your script. If you need to perform arithmetical operations on a parameter, you must change it from a string to an integer using the **atoi** or **atol** C functions. After you modify the value as an integer, you must convert it back to a string to use the new variable in your script.

In the following WinSock example, the data at offset 67 was saved to the parameter, **param1**. Using **atol**, VuGen converted the string to a long integer. After increasing the value of **param1** by one, VuGen converted it back to a string using sprintf and saved it as a new string, **new_param1**. The value of the parameter is displayed using **lr_output_message**. This new value may be used at a later point in the script.

```
lrs_receive("socket2", "buf47", LrsLastArg);lrs_save_param("socket2",
                NULL, "param1", 67, 5);
lr_output_message ("param1: %s", lr_eval_string("<param1>"));
sprintf(new_param1, "value=%ld", atol(lr_eval_string("<param1>")) + 1);
lr_output_message("ID Number:"%s" lr_eval_string("new_param1"));
```

9

Running Vuser Scripts in Standalone Mode

After you develop a Vuser script and set its run-time settings, you test the Vuser script by running it in stand-alone mode.

This chapter includes:

➤ About Running Vuser Scripts in Standalone Mode on page 139

➤ Running a Vuser Script in VuGen on page 140

➤ Replaying a Vuser Script on page 144

➤ Using VuGen's Debugging Features on page 147

➤ Using VuGen's Debugging Features for Web Vuser Scripts on page 152

➤ Working with VuGen Windows on page 153

➤ Find In Files on page 154

➤ Running a Vuser Script from a Command Prompt on page 156

➤ Running a Vuser Script from a UNIX Command Line on page 156

➤ Integrating Scripts into Tests on page 159

About Running Vuser Scripts in Standalone Mode

After creating a script, you check its functionality by running it in standalone mode, directly from the VuGen interface. If the script is UNIX-based, you run it from a UNIX command line. To run GUI Vusers in standalone mode, use WinRunner.

You run a script as a standalone test to check its basic functionality.

When the standalone execution is successful, you integrate it into your environment: a LoadRunner scenario, Performance Center load test, or Business Process Monitor profile. For more information, see the *HP LoadRunner Controller*, *HP Performance Center*, or *HP Business Availibility Center User Guides*.

Once you run a script in standalone mode, you can enhance and customize it:

- with Vuser functions (see Chapter 6, "Enhancing Vuser Scripts" or the *Online Function Reference*).
- with parameters (see Chapter 13, "Working with VuGen Parameters")
- correlate queries in the script (see Chapter 8, "Correlating Statements")

The above steps are optional and may not apply to all scripts.

Running a Vuser Script in VuGen

After developing a Vuser script, run it using VuGen to verify that it executes correctly. You can set several options for replay.

Note: VuGen runs Vuser scripts on Windows platforms only. To run UNIX-based Vuser scripts, see "Running a Vuser from the Unix Command Line" on page 553.

Configuring Replay Options

You can run a Vuser script in animated mode or non-animated mode. When you run in animated mode, VuGen highlights the line of the Vuser script being executed at the current time. You can set a delay for this mode, allowing you to better view the effects of each step. When you run in non-animated mode, VuGen executes the Vuser script, but does not indicate the line being executed.

- **Animated run delay.** The time delay in milliseconds between commands. The default delay value is 0.
- **Only animate functions in Actions sections.** Only animates the content of the Action sections, but not the *init* or *end* sections.
- **Prompt for results directory.** Prompts you for a results directory before running a script from VuGen. If this option is not selected, VuGen automatically names the directory *result1*. Subsequent script executions will automatically overwrite previous ones unless you specify a different result file. Note that results are stored in a subdirectory of the script.
- **After replay show.** Instructs VuGen how to proceed after the replay:
  - **View before replay.** Return to the view you had before replay.
  - **Replay summary.** Go directly to the Replay Summary window in the Workflow Wizard.
  - **Visual Test Results**. Open the Test Results Summary. (This is the same as choosing **View** > **Test Results** after replay.)

Use Defaults

When you click **Use Defaults**, VuGen resets the original values:

Animated run delay to 0 msec.

Only animate functions in action sections becomes enabled.

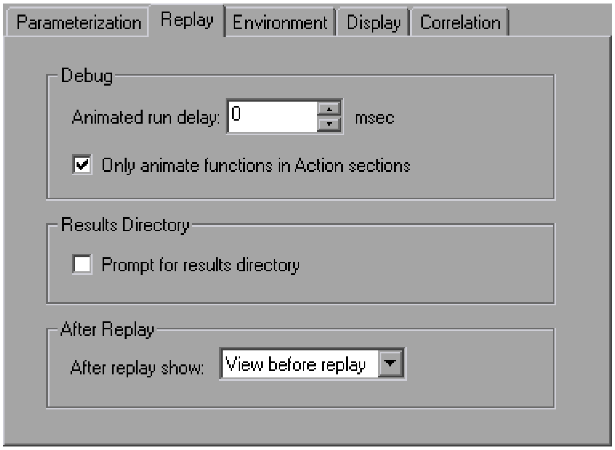
Prompt for results directory becomes disabled.

After replay show is set to View before replay.

To enable animation and set its properties:

1 Select **View** > **Animated Run** to run in animated mode. VuGen places a check mark beside the **Animated Run** menu option to enable animated mode.

2 To set the delay for the animated run, select **Tools** > **General Options**. The General Options dialog box opens.



Debug

Animated run delay: 0 msec

☑ Only animate functions in Action sections

Results Directory

☐ Prompt for results directory

After Replay

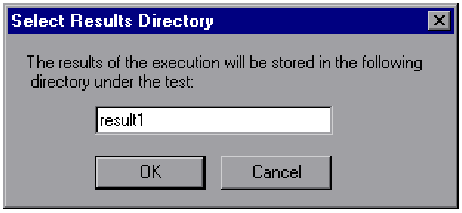
After replay show: View before replay

3 Select the **Replay** tab.

4 In the **Animated run delay** box, specify a delay in milliseconds and click **OK**.

5 Select **Only animate functions in Actions sections** to animate only the content of the Action sections.

6 Select **Prompt for results directory** to be prompted for a results directory before running a script from VuGen. The Select Results Directory dialog box opens when you click the run command.



Select Results Directory

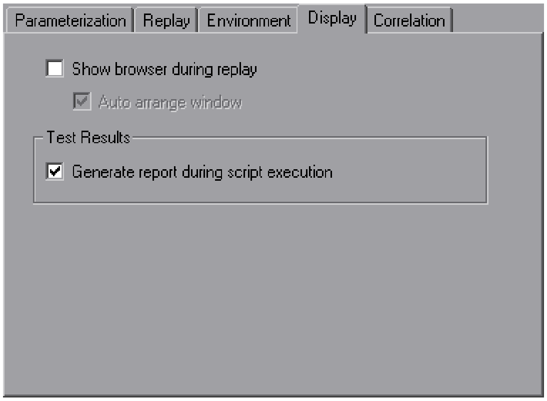
The results of the execution will be stored in the following directory under the test:

result1

7 Type a directory name for the execution results, or accept the default name and click **OK**.

Setting the Display Options

If you are running a Web Vuser script, you can set the Display options (**Tools** > **General Options**). These options specify whether to display VuGen's run-time viewer, whether to generate a report during script execution, and so forth.



□ Show browser during replay

☑ Auto arrange window

Test Results

☑ Generate report during script execution

- **Show browser during replay.** Enables the run-time viewer. The **Auto arrange window** options instructs VuGen to minimize the run-time viewer when script execution is complete.
- **Generate report during script execution.** Instructs a Vuser to generate a Results Summary report. You can open the report after script execution by selecting **View** > **Test Results**.

By default, the **Show browser during replay** option is disabled, and the **Generate report during script execution** option is enabled. To restore these values, click **Use Defaults**.

For more information on how to use these options for debugging, see "Using VuGen's Debugging Features for Web Vuser Scripts" on page 152.

To set the Display options:

1 Select **Tools** > **General Options** from the VuGen menu. The General Options dialog box opens. Select the **Display** tab.

2 Select **Show browser during replay** to enable the run-time viewer. Select **Auto arrange window** to minimize the run-time viewer when script execution is complete.

3 Select **Generate report during script execution** to instruct a Vuser to generate a Results Summary report. You can open the report after script execution by selecting **View** > **Test Results**.

4 Click **OK** to accept the settings and close the General Options dialog box.

Replaying a Vuser Script

Before you integrate a script into a test or production environment, you run it from VuGen to make sure it is functional. VuGen provides several tools that allow you to monitor the replay and locate any existing and potential problems. These include:

- Viewing the Replay Log
- The Run-Time Data Tab
- The Run Step by Step Command
- Breakpoints
- Bookmarks
- Go To Commands

To replay a script in VuGen:

1 Select **Vuser** > **Run**.

The Output window opens at the bottom of the VuGen main window—or clears, if already open—and VuGen begins executing the Vuser script. In tree view, VuGen runs the Vuser script from the first icon in the script. In Script view, it runs the Vuser script from the first line of the script.

2 Click the Output window's **Replay Log** tab for a log of all of the actions of the Vuser, along with warnings and errors. For more information, see "Viewing the Replay Log" on page 145.

3 To view a summary of the run-time data and the parameters as they are being used, see the Output window's **RunTime Data** tab. For more information, see "The Run-Time Data Tab" on page 146.

4 To hide the Output window during or after a script run, select **View** > **Output Window**. VuGen closes the Output window and removes the check mark from next to **Output Window** on the **View** menu.

5 To interrupt a Vuser script that is running, select **Vuser** > **Pause**, to temporarily pause the script run, or **Vuser** > **Stop**, to end the script run.

Viewing the Replay Log

The Output window's Replay Log displays messages that describe the actions of the Vuser as it runs. This information tells you how the script will run when executed in a scenario or profile.

When script execution is complete, you examine the messages in the Replay Log to see whether your script ran without errors.

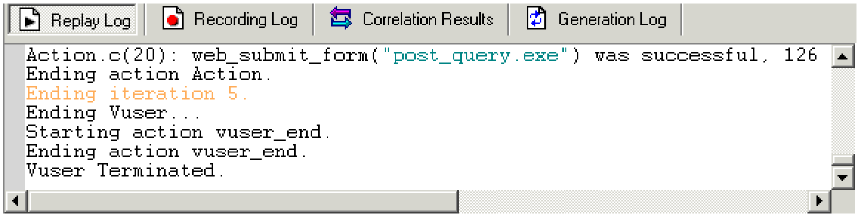
Various colors of text are used in the Replay Log.

- **Black.** Standard output messages.
- **Red.** Standard error messages.
- **Green.** Literal strings, such as URLs, that appear between quotation marks.
- **Blue.** Transaction Information (starting, ending, status and duration).
- **Orange.** The beginning and ending of iterations.

If you double-click on a line beginning with the Action name, the cursor jumps to the step within the script that generated.

For more information on closing and opening the Output window, see "Replaying a Vuser Script" on page 144.

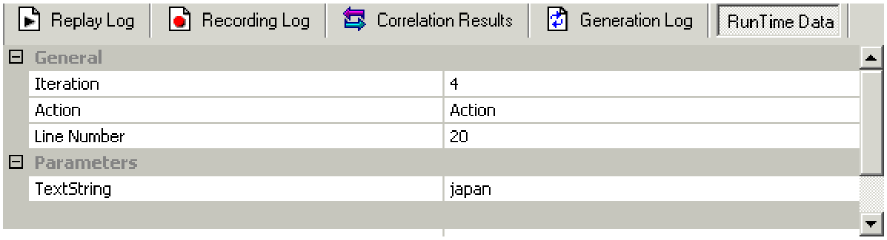
The following example shows Replay Log messages from a Web Vuser script run.



```
Action.c(20): web_submit_form("post_query.exe") was successful, 126
Ending action Action.
Ending iteration 5.
Ending Vuser...
Starting action vuser_end.
Ending action vuser_end.
Vuser Terminated.
```

The Run-Time Data Tab

You can track the script information that becomes updates during replay, using the Run Time Data tab.



⊟ General	
Iteration	4
Action	Action
Line Number	20
⊟ Parameters	
TextString	japan

During replay, click the rightmost tab, **RunTime Data**. The tab contains two expandable/collapsible sections:

- **General.** The general section shows the current iteration number, the Action name of the currently replayed step, and the line number within the script (Script view).
- **Parameters.** The parameters section shows all parameters defined with the script and their substitution values based on the selected update method (sequential, unique, etc.). VuGen shows this information even if the parameter is not used in the script. For more information, see Chapter 13, "Working with VuGen Parameters."

Note that the RunTime Data tab is not accessible after the test run, since it only displays data that changes during replay.

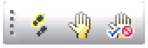
Using VuGen's Debugging Features

VuGen contains two options to help debug Vuser scripts—the Run Step by Step command and breakpoints. These options are not available for VBscript and VB Application type Vusers.

VuGen contains additional features to help debug Web Vuser scripts. For details, see "Using VuGen's Debugging Features for Web Vuser Scripts" on page 152.

To view the Debug toolbar:

Right-click the toolbar area and select **Debug**. The Debug toolbar appears in the toolbar area.



The Run Step by Step Command

The Run Step by Step Command runs the script one line at a time. This enables you to follow the script execution.

To run the script step by step:

1 Select **Vuser** > **Run Step by Step**, or click the **Step** button on the Debug toolbar.

VuGen executes the first line of the script.

2 Continue script execution by clicking the **Step** button until the script run completes.

Breakpoints

Breakpoints pause execution at specific points in the script. This enables you to examine the effects of the script on your application at pre-determined points during execution. To manage the breakpoints, use the Breakpoint Manager.

To set breakpoints:

1 Place the cursor on the line in the script at which you want execution to stop.

2 Select **Insert** > **Toggle Breakpoint**, or click the **Breakpoint** button in the Debug toolbar. Alternatively, press F9 on the keyboard. The Breakpoint symbol (●) appears in the left margin of the script.

3 To disable a breakpoint, place the cursor on the line with the breakpoint symbol, and click the **Enable / Disable Breakpoint** button on the Debug toolbar. A white dot inside the Breakpoint symbol indicates a disabled breakpoint. When one breakpoint is disabled, script execution is paused at the following breakpoint. Click the button again to enable the breakpoint.

   To remove the breakpoint, place the cursor on the line with the breakpoint symbol, and click the **Breakpoint** button or press F9.

To run the script with breakpoints:

1 Begin running the script as you normally would.

   VuGen pauses script execution when it reaches a breakpoint. You can examine the effects of the script run up to the breakpoint, make any necessary changes, and then restart the script from the breakpoint.

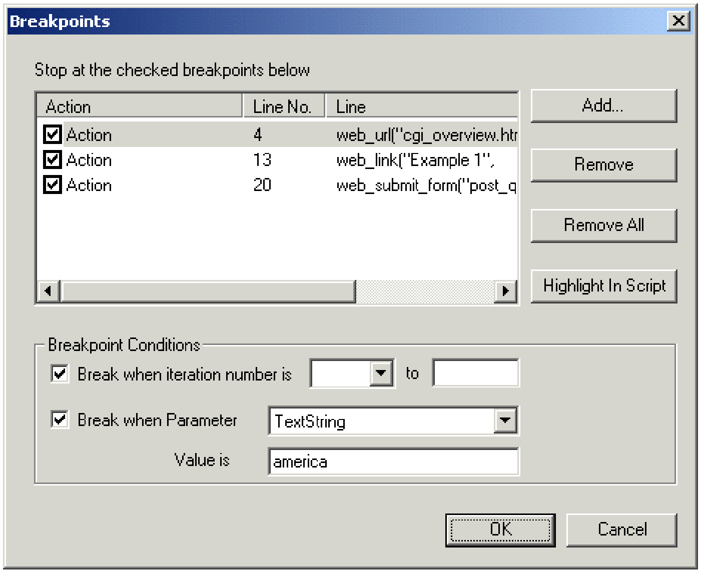
2 To resume execution, select **Vuser** > **Run**.

Once restarted, the script continues until the next breakpoint is encountered or until the script is completed.

The Breakpoint Manager

You can view and manage breakpoints using the Breakpoint Manager. From the Breakpoint Manager you can manipulate all of the breakpoints in your script.

To open the Breakpoint Manager, select **Edit** > **Breakpoints**.


Breakpoints

Stop at the checked breakpoints below

Action	Line No.	Line
☑ Action	4	web_url("cgi_overview.htr
☑ Action	13	web_link("Example 1",
☑ Action	20	web_submit_form("post_q

Breakpoint Conditions

☑ Break when iteration number is [ ] to [ ]

☑ Break when Parameter: TextString

Value is: america

To jump to the breakpoint location in the script:

1 Select a breakpoint from the list.

2 Click **Highlight in Script**. The line in the script becomes highlighted.

Note that you can only highlight one breakpoint at a time.

Managing Breakpoints

From the Breakpoint Manager, you can add, remove, disable, or conditionalize a breakpoint.

To add a breakpoint:

1. Click **Add**. The Add Breakpoint dialog box opens.
2. Select an **Action** and specify the **Line** number where you want add the breakpoint.
3. Click **OK.** The Breakpoint is added to the list of breakpoints.

To remove a breakpoint:

1. To remove a single breakpoint, select the breakpoint and click **Remove**.
2. To remove all the breakpoints at once, click **Remove All**.

To enable/disable a breakpoint:

1. To enable a breakpoint, in the Action column, select the action's check box.
2. To disable a breakpoint, in the Action column, clear the action's check box.

From the Breakpoint Manager, you can set breakpoints to pause execution under certain conditions.

To conditionalize a breakpoint:

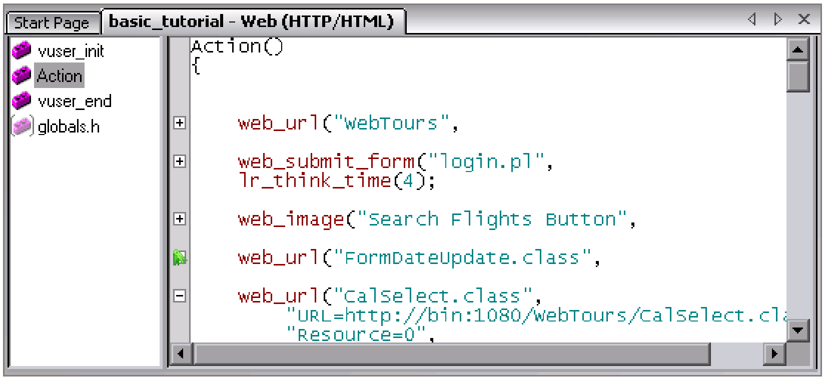
1. To pause the script after a specific number of iterations, select **Break when iteration number is** and enter the desired number.
2. To pause the script when parameter *X* has a specific value, select **Break when Parameter *X* Value is** and enter the desired value. For more information about parameters, see Chapter 13, "Working with VuGen Parameters."

Bookmarks

When working in Script view, VuGen lets you place bookmarks at various locations within your script. You can navigate between the bookmarks to analyze and debug your code.

To create a bookmark:

1 Place the cursor at the desired location and press Ctrl + F2. VuGen places an icon in the left margin of the script.



```
Action()
{

    web_url("WebTours",

    web_submit_form("login.pl",
    lr_think_time(4);

    web_image("Search Flights Button",

    web_url("FormDateUpdate.class",

    web_url("CalSelect.class",
        "URL=http://bin:1080/WebTours/CalSelect.cl
        "Resource=0",
```

2 To remove a bookmark, click on the desired bookmark and press Ctrl + F2. VuGen removes the bookmark icon from the left margin.

3 To move between bookmarks:

To move to the next bookmark, click F2.

To navigate to the previous bookmark, click Shift + F2.

You can also create and navigate between bookmarks through the **Edit** > **Bookmarks** menu item.

Note: You can only navigate between bookmarks in the current action. To navigate to a bookmark in another action, select that action in the left pane and then press F2.

Go To Commands

To navigate around the script without using bookmarks, you can use the Go To command. Select **Edit** > **Go To Line** and specify the line number of the script. This navigation is also supported in Tree view.

If you want to examine the Replay log messages for a specific step or function, select the step in VuGen and select **Edit** > **Go To Step in Replay Log.** VuGen places the cursor at the corresponding step in the Output window's Replay Log tab.

Using VuGen's Debugging Features for Web Vuser Scripts

VuGen provides two additional tools to help you debug Web Vuser scripts—the run-time viewer (online browser) and the Results Summary report.

- You can instruct VuGen to display a run-time viewer when you run a Web Vuser script. The run-time viewer was developed specifically for use with VuGen—it is unrelated to the browser that you use to record your Vuser scripts. The run-time viewer shows each Web page as it is accessed by the Vuser. This is useful when you debug Web Vuser scripts because it allows you to check that the Vuser accesses the correct Web pages.
- You can specify whether or not a Web Vuser generates a Results Summary report during script execution. The Results Summary report summarizes the success or failure of each step in the Web Vuser scripts and allows you to view the Web page returned by each step. For additional details on working with the Results Summary report, select **View** > **Test Results** and click F1 to open the online help.

For information on setting the above Display options, see "Setting the Display Options" on page 143.

Note: Transaction times may be increased when a Vuser generates a Results Summary report.

Vusers can generate Results Summary reports only when run from VuGen. When you run a script from the Controller or Business Process Monitor, Vusers do not generate reports.

Working with VuGen Windows

You can show and rearrange VuGen's windows to view the relevant data for your script, using the following features:

- **Show/Hide the Output Window.** Select **View** > **Output Window** to show and hide the Output window below the VuGen script editor. The Output window has several tabs, depending on the protocol. The most common tabs are the Replay Log, Recording Log, Generation Log, and Correlation Results. For more information, see "Viewing the Replay Log" on page 145.
- **Display All Thumbnails.** Select **View** > **Show All Thumbnails** to show all of the script's steps as thumbnails. To show thumbnails for primary steps only, clear this option. For more information, see "Viewing Script Thumbnails" on page 46.
- **Display Grids.** Select **View** > **Enable Data Grids** to enable grid display of the data in the protocols that support data grids (Database, COM, and Microsoft . NET). The grids appear inside the script.
- **Window Manipulation.** Select **Window** > **Close All** to close all of the open scripts. If necessary, VuGen will prompt you to save the unsaved scripts.

Find In Files

You use Find In Files to search for any string or expression in all the files of the script you currently have open.

For example, if VuGen fails to replay a script due to an error in the replay log, you can search through all the files in the script simultaneously for a specific value that you think may be causing the failure.

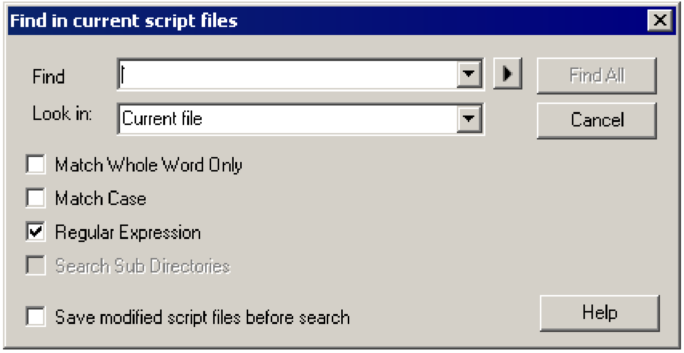
Vugen displays all matches in a separate results window. You double-click on any line in the window to open the relevant file.

Note: VuGen also includes a regular **Find** feature. With this feature, you can search for values in only one file at a time. The value is highlighted in the script itself, and you press F3 to move to the next match.

To search with Find In Files:

1 Open a script in VuGen.

2 Select **Edit** > **Find in current script files.** The Find in current script files dialog box opens.



Find in current script files

Find:

Look in: Current file

□ Match Whole Word Only

□ Match Case

☑ Regular Expression

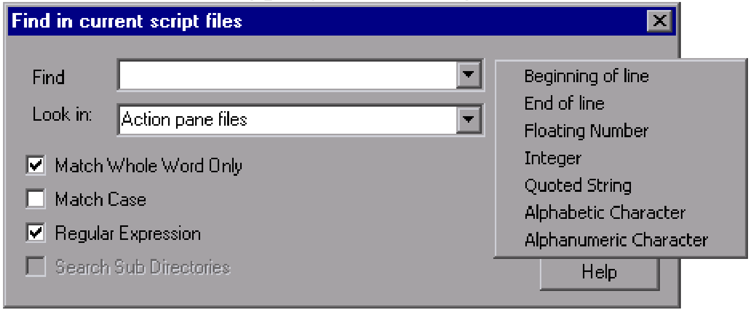
□ Search Sub Directories

□ Save modified script files before search

3 In the **Find** box, enter the string you want to search for. You can select any of the previous ten searches from the drop-down list.

You can also search with a regular expression. A regular expression is a search string made up of a single character, or a set of characters, that is interpreted by VuGen as a pattern in the script. VuGen searches the script for values that match that pattern.

If you select **Regular Expression**, the arrow button by the **Find** box is activated, and provides seven common regular expressions that you can select instead of typing in manually.



Find in current script files

Find:

Look in: Action pane files

☑ Match Whole Word Only

□ Match Case

☑ Regular Expression

□ Search Sub Directories

4 In the **Look in** box, specify the script directory in which to search. You can also select from the two options provided by VuGen in the drop-down list.

- **Current file.** Searches the file currently displayed in the right pane.
- **Action pane files.** Searches all files that appear in the left pane.

If you specify a script directory other than those provided, the **Search Sub Directories** option is activated. You select this option to expand the search to sub directories of the current Vuser folder.

5 Select the desired settings from: **Match Whole Word Only** and **Match Case**.

6 Click **Find All**. The Search Results tab opens.

Note: We recommend that you save the script before you search. If you make changes in the results, you might not be able to return to the original script. Note that Find In Files only searches the last saved version of the script. The **Save modified script files before search** option appears if you have already made changes to the script.

Running a Vuser Script from a Command Prompt

You can test a Vuser script from a Command Prompt or from the Windows Run dialog box—without the VuGen user interface.

To run a script from a DOS command line or the Run dialog box:

1. Select **Start** > **Programs** > **Command Prompt** to open a **Command Prompt** window, or select **Start** > **Run** to open the Run dialog box.
2. Type the following and press **Enter**:

```
<installation_dir>/bin/mdrv.exe -usr <script_name> -vugen_win 0
```

   script_name is the full path to the *.usr* script file, for example, **c:\temp\mytest\mytest.usr.**

   The mdrv program runs a single instance of the script without the user interface. Check the output files for run-time information.

Running a Vuser Script from a UNIX Command Line

When using VuGen to develop UNIX-based Vusers, you must check that the recorded script runs on the UNIX platform. To verify that your script runs correctly, follow these steps:

1. **Test the recorded script from VuGen.**

   Run the recorded script from VuGen to check that the script runs correctly on a Windows-based system.

2. **Copy the Vuser script files to a UNIX drive.**

   Transfer the files to a local UNIX drive.

3. **Check the Vuser setup on the UNIX machine by using verify_generator.**

   For details, see "The verify_generator Test" on page 157.

4 Test the script from the UNIX command line.

Run the script in standalone mode from the Vuser script directory, using the run_db_vuser shell script:

```
run_db_vuser.sh script_name.usr
```

The verify_generator Test

The verify utility checks the local host for its communication parameters and its compatibility with all types of Vusers.

The utility checks the following items in the Vuser environment:

- at least 128 file descriptors
- proper .rhost permissions: -rw-r--r--
- the host can be contacted using rsh to the host. If not, checks for the host name in .rhosts
- M_LROOT is defined
- .cshrc defines the correct M_LROOT
- .cshrc exists in the home directory
- the current user is the owner of the .cshrc
- a LoadRunner installation exists in $M_LROOT
- the executables have executable permissions
- PATH contains $M_LROOT/bin, and /usr/bin
- the rstatd daemon exists and is running

If you intend to run all of the Vusers on one host, type:

```
verify_generator
```

verify_generator either returns ‘OK’ when the setting is correct, or ‘Failed’ and a suggestion on how to correct the setup.

For detailed information about the verify checks type:

```
verify_generator [-v]
```

Command Line Options: run_db_vuser Shell Script

The *run_db_vuser* shell script has the following command line options:

--help
Display the available options. (This option must be preceded by two dashes.)

-cpp_only
Run cpp only (pre-processing) on the script.

-cci_only
Run cci only (pre-compiling) on the script to create a file with a .ci extension. You can run cci only after a successful cpp.

-driver *driver_path*
Use a specific driver program. Each database has its own driver program located in the /bin directory. For example, the driver for CtLib located in the /bin directory, is *mdrv*. This option lets you specify an external driver.

-exec_only
Execute the Vuser .ci file. This option is available only when a valid .ci file exists.

-ci *ci_file_name*
Execute a specific .ci file.

-out *output_path*
Place the results in a specific directory.

By default, *run_db_vuser.sh* runs **cpp**, **cci**, and **execute** in verbose mode. It uses the driver in the *VuGen installation*/bin directory, and saves the results to an output file in the Vuser script directory. You must always specify a *.usr* file. If you are not in the script directory, specify the full path of the *.usr* file.

For example, the following command line executes a Vuser script called test1, and places the output file in a directory called results1. The results directory must be an existing directory—it will not be created automatically:

```
run_db_vuser.sh -out /u/joe/results1  test1.usr
```

Integrating Scripts into Tests

Once you have successfully run a script in standalone mode to verify that it is functional, you integrate it into your environment: a LoadRunner scenario, Performance Center load test, or Business Process Monitor profile.

When you integrate a test, you provide information indicating:

- which script
- Vusers that will run the script
- load generator upon which the script will be executed
- scheduling

For more information, see the *HP LoadRunner Controller, HP Performance Center*, or *HP Business Availibility Center User Guides*.

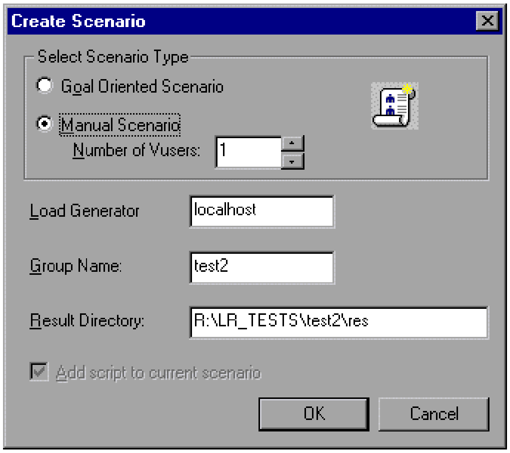
Using VuGen to Create a LoadRunner Scenario

Note: The following section only applies to LoadRunner. For information on integrating scripts into Business Process profiles, see the *HP Business Availability Center* documentation.

Normally, you create a scenario from the LoadRunner Controller. You can also create a basic scenario from VuGen using the current script.

To create a scenario from VuGen:

1 Select **Tools** > **Create Controller Scenario**. The Create Scenario dialog box opens.



Create Scenario

Select Scenario Type

○ Goal Oriented Scenario

● Manual Scenario

Number of Vusers: 1

Load Generator: localhost

Group Name: test2

Result Directory: R:\LR_TESTS\test2\res

☑ Add script to current scenario

2 Select either a goal oriented or a manual scenario.

In a goal-oriented scenario, LoadRunner automatically builds a scenario based on the goals you specify, whereas in a manual scenario, you specify the number of Vusers to run.

3 For a manual scenario, enter the number of Vusers you want to execute the script.

4 Enter the name of the machine upon which you want the Vusers to run, in the **Load Generator** box.

5 For a manual scenario, users with common traits are organized into groups. Specify a new group name for the Vusers in the **Group Name** box.

6 For a goal-oriented scenario, specify a **Script Name**.

7 Enter the desired location for the results in the **Result Directory** box.

8 If a scenario is currently open in the Controller and you want to add the script to this scenario, select the **Add script to current scenario** check box. If you clear the check box, LoadRunner opens a new scenario with the specified number of Vusers.

9. Click **OK**. VuGen opens the Controller in the Vuser view.
10. If you configured the Controller to save the script on a shared network drive, you may need to perform path translation.

For more information, see the *HP LoadRunner Controller User's Guide*.

10

Viewing Test Results

To assist with debugging a Vuser script, you can view a report that summarizes the results of your script run. VuGen generates the report during the Vuser script execution and you view the report when script execution is complete.

This chapter includes:

- About Viewing Test Results on page 164
- The Test Results Window on page 165
- Viewing the Results on page 168
- Finding Results Steps on page 175
- Printing Results on page 176
- Exporting Test Results on page 179
- Submitting Defects to Quality Center on page 180
- Connecting to Quality Center from the Test Results Window on page 181
- Customizing the Test Results Display on page 182
- Viewing Web Services Reports on page 182
- Sending Custom Information to the Report on page 185

About Viewing Test Results

After you run a script, the Test Results window displays all aspects of the test run and can include:

- a high-level results overview report (pass/fail status)
- the data used in all runs
- an expandable tree of the steps, specifying exactly where application failures occurred
- the exact locations in the script where failures occurred
- a still image of the state of your application at a particular step
- a movie clip of the state of your application at a particular step or of the entire test
- detailed explanations of each step and checkpoint pass or failure, at each stage of the test

Reports for Web Services Vusers contain several enhacements, such as breakdown by operations, checkpoint results, and a view of the HTTP traffic. For more information, see "Viewing Web Services Reports" on page 182.

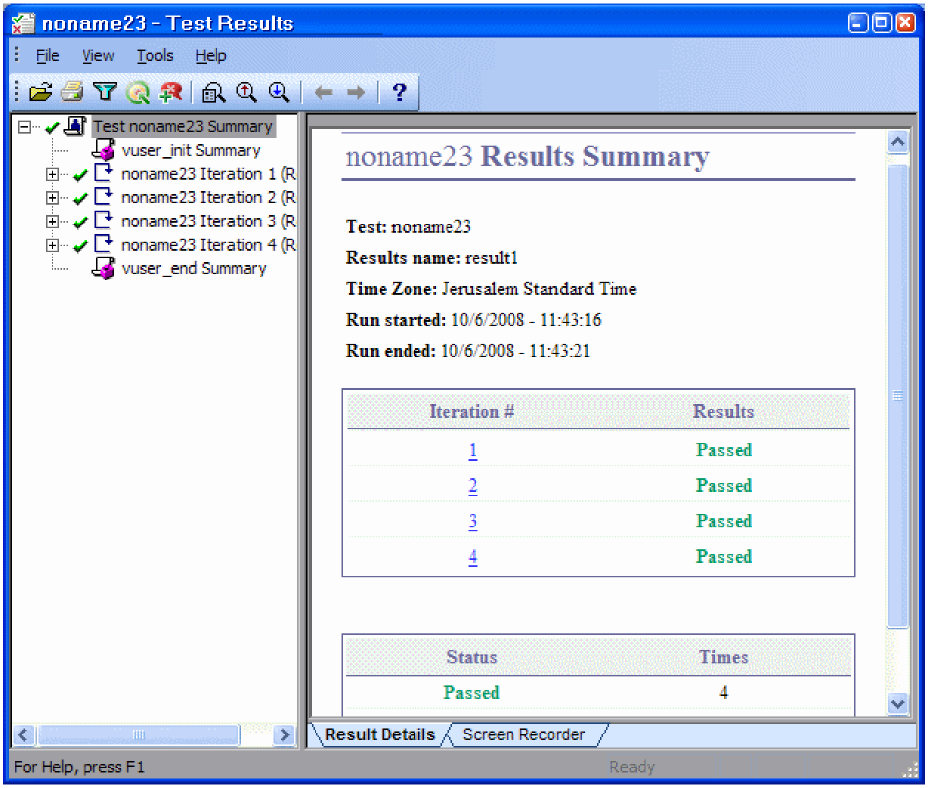
You can open the Test Results window as a standalone application from the **Start** menu. To open the Test Results window, choose **Start** > **All Programs** > **HP Service Test** > **Test Results Viewer**.

Note: The Test Results window can show results with up to 300 levels in the tree hierarchy. If you have results with more than 300 nested levels, you can view the entire report by manually opening the **results.xml** file.

The Test Results Window

After a run session, you view the results in the Test Results window.

- The left pane displays the report tree—a graphical representation of the results. In the report tree, a green check mark represents a successful step, and a red X represents a failed step.
- The right pane displays the report details—an overall summary of the script run, as well as additional information for a selected branch of the report tree.



noname23 **Results Summary**

Test: noname23

Results name: result1

Time Zone: Jerusalem Standard Time

Run started: 10/6/2008 - 11:43:16

Run ended: 10/6/2008 - 11:43:21

Iteration #	Results
1	Passed
2	Passed
3	Passed
4	Passed

Status	Times
Passed	4

The Test Results window contains the following key elements:

- **Test results title bar.** Displays the name of the test.
- **Menu bar.** Displays menus of available commands.
- **Run results toolbar.** Contains buttons for viewing test results (choose **View** > **Test Results Toolbar** to display the toolbar)._For more information, see "Test Results Toolbar" on page 167.
- **Run results tree.** Contains a graphic representation of the test results in the run results tree. The run results tree is located in the left pane of the Test Results window. For more information, see "Run Results Tree" on page 166.
- **Result Details tab.** Contains details of the selected node in the run results tree. The Result Details tab is located in the right pane of the Test Results window. For more information, see "Run Results Details" on page 167.
- **Screen Recorder tab.** Contains the recorded movie associated with the test results. The screen recorder tab is located in the right pane of the Test Results window This is supported for scripts created with HP QuickTest Professional.
- **Status bar.** Displays the status of the currently selected command (choose **View** > **Status Bar** to view the status bar).

You can change the appearance of the Test Results window. For more information, see "Changing the Appearance of the Test Results Window" on page 168.

Run Results Tree

The left pane in the Test Results window displays the **run results tree**—a graphical representation of the test results:

- ✓ indicates a step that succeeded.
- ✗ indicates a step that failed.
- ! indicates a warning, meaning that the step did not succeed, but it did not cause the test to fail.
- ! ⊗ indicates a step that failed unexpectedly.

You can collapse or expand a branch in the run results tree to change the level of detail that the tree displays.

Run Results Details

By default, when the Test Results window opens, it displays a summary of the script run in the **Result Details** tab in the right pane of the window.

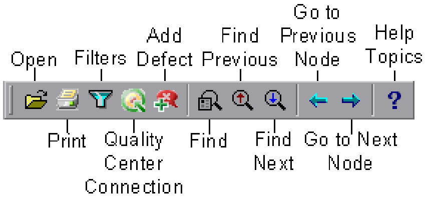
The right pane of the Test Results Window contains tabs labeled **Result Details** and **Screen Recorder**. When you select the top node of the run results tree, the Result Details tab shows a summary of the results for your test. When you select a branch or step in the tree, the Result Details tab contains the details for that step. The Result Details tab may also include a still image of your application for the highlighted step.

The Screen Recorder tab contains the movie associated with your test results. If there is no movie associated with your test results, the Screen Recorder tab contains the message: No movie is associated with the results.

When you select the top node of the results tree in the **Result Details** tab, it indicates the script name, results name, the start and end date and time of the run, the number of iterations, and whether an iteration passed or failed. For checkpoints in Web Services scripts, the possible results are **Passed** or **Failed**. If an iteration does not contain checkpoints, the possible results are **Done** or **Failed**.

Test Results Toolbar

The Test Results toolbar contains buttons for viewing test results.



Changing the Appearance of the Test Results Window

By default, the Test Results window has the same look and feel as the QuickTest window, using the Microsoft Office 2003 theme. You can change the look and feel of the Test Results window, as required.

To change the appearance of the Test Results window:

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

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

Viewing the Results

By default, VuGen generates test results and automatically opens them at the end of a run.

To prevent VuGen from generating the results, choose **Tools** > **General Options**, select the **Display** tab and clear the **Generate report during script execution option**.

To indicate whether or not to open the results after running the script, choose **Tools** > **General Options**, select the **Replay** tab, and select a view in the **After Replay** section.

For more information on setting the Display options, see "Running Vuser Scripts in Standalone Mode" in the *Virtual User Generator* User Guide.

In addition, you can view the results of previous runs of the current script, and results of other script. You can also preview results on screen and print them to your default Windows printer, as well as export them to an HTML file.

For more information, see:

- "About Viewing Test Results" on page 164
- "The Test Results Window" on page 165
- "Viewing the Results" on page 168
- "Finding Results Steps" on page 175
- "Printing Results" on page 176
- "Previewing Test Results" on page 177
- "Exporting Test Results" on page 179

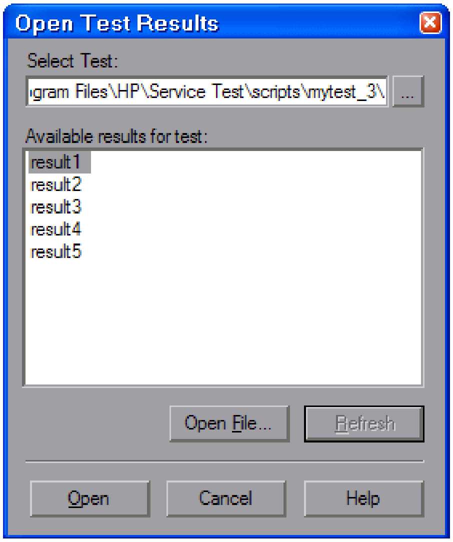
Opening Test Results to View a Selected Run

You can view the saved results for the current script, or you can view the saved results for other scripts.

To select a specific set of test results:

1 Choose **File** > **Open** from within the Test Results window.

2 To view all of the results, specify the parent script path.



Open Test Results

Select Test: ıgram Files\HP\Service Test\scripts\mytest_3\

Available results for test:

result1
result2
result3
result4
result5

3 Select a results set and click **Open**.

Tip: To update the results list after you specify a new path, click **Refresh**.

Searching for Results in the File System

By default, the results saved in the file system are stored in the script folder. If you are connected to Quality Center, the results are stored together with You can search for results in the file system by script or by result file.

To search for results in the file system by script:

1 In the Open Test Results dialog box, enter the path of the folder that contains the results file for your script, or click the browse button to open the Open Test dialog box.

2 Find and highlight the script whose results you want to view, and click **Open**.

3 In the Open Test Results dialog box, highlight the result set you want to view, and click **Open**. The Test Results window displays the selected results.

To search for results in the file system by result file:

1 In the Open Test Results dialog box, click the **Open File** button to open the Select Results File dialog box.

2 Browse to the folder where the results file is stored.

3 Highlight the results (**.xml**) file you want to view, and click **Open**. The Test Results window displays the selected results.

Notes:

- By default, result files for tests are stored in <**Script**>\<**ResultName**>.
- Results files for earlier versions were saved with a **.qtp** file extension. In the Select Results File dialog box, only results files with an **.xml** extension are shown by default. To view results files with a **.qtp** extension in the Select Results File dialog box, select **Test Results (*.qtp)** in the **Files of type** box.

Searching for Results Saved in Quality Center

If your script is stored in Quality Center, the results are also stored in Quality Center. You cannot change the location of the test results.

To search for test results saved in Quality Center:

1 In the Test Results window, choose **Tools** > **Quality Center Connection** or click the **Quality Center Connection** button and connect to your Quality Center project.

2 In the Open Test Results dialog box, enter the path of the folder that contains the results file for your QuickTest test, or click the browse button to open the Open Test from Quality Center Project dialog box.

3 Select **DB Vuser** in the **Test Type** list.

4 Find and highlight the script whose test results you want to view, and click **OK**.

5 In the Open Test Results dialog box, highlight the test result set you want to view, and click **Open**. The Test Results window displays the selected test results.

For more information on working with Quality Center, see Chapter 11, "Managing Scripts Using Quality Center".

Working in Test Results Tree

The following steps describe how to work within the nodes of the Test Results tree:

1 You can collapse or expand a branch in the run results Tree to select the level of detail that the tree displays.

- ➤ To collapse a branch, select it and click the collapse (–) sign to the left of the branch icon, or press the minus key (–) on your keyboard number pad. The details for the branch disappear in the results tree, and the collapse sign changes to expand (+).
- ➤ To collapse all of the branches in the run results tree, choose **View** > **Collapse All** or right click a branch and select **Collapse All**.
- ➤ To expand a branch, select it and click the expand (+) sign to the left of the branch icon, or press the plus key (+) on your keyboard number pad. The tree displays the details for the branch and the expand sign changes to collapse.

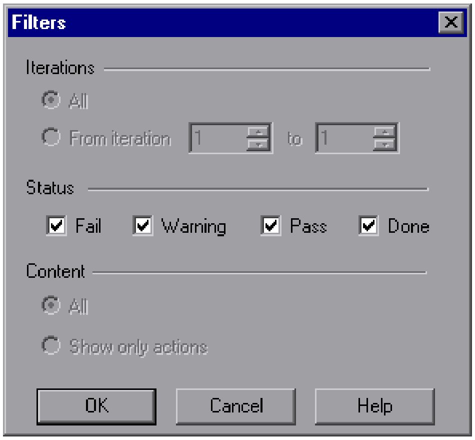
  If you just opened the Test Results window, the tree expands one level at a time. If the tree was previously expanded, it reverts to its former state.
- ➤ To expand a branch and all branches below it, select the branch and press the asterisk (*) key on your keyboard number pad.
- ➤ To expand all of the branches in the run results tree, choose **View** > **Expand All**; right click a branch and select **Expand All**; or select the top level of the tree and press the asterisk (*) key on your keyboard number pad.

2 You can view the results of an individual iteration, an action, or a step. When you select a step in the run results tree, the right side of the Test Results window contains the details of the selected step. Depending on your settings in the Run tab of the Options dialog box, the right side of the Test Results window may be split into two panes, with the bottom pane containing a still image (or in selected cases, other data) of the selected step.

The results can be one of the following types:

- Iterations, actions, and steps that contain checkpoints are marked **Passed** or **Failed** in the right part of the Test Results window and are identified by icons in the tree pane.
- Iterations, actions, and steps that ran successfully, but do not contain checkpoints, are marked **Done** in the right part of the Test Results window.
- Steps that were not successful, but did not cause the script to stop running, are marked **Warning** in the right part of the Test Results window and are identified by the icon ! or ! ⊗.

3 To filter the information displayed in the Test Results window, click the **Filters** button or choose **View** > **Filters**. The Filters dialog box opens.



Filters

Iterations

● All

○ From iteration 1 to 1

Status

☑ Fail ☑ Warning ☑ Pass ☑ Done

Content

● All

○ Show only actions

The default filter options are displayed in the image above. The Filters dialog box contains the following options:

Iterations area:

- **All.** Displays test results from all iterations.
- **From iteration X to Y.** Displays the test results from a specified range of test iterations.

Status area:

- **Fail.** Displays the results for the steps that failed.
- **Warning.** Displays the results for the steps with the status **Warning** (steps that did not pass, but did not cause the script to fail).
- **Pass.** Displays the results for the steps that passed.
- **Done.** Displays the results for the steps with the status **Done** (steps that were performed successfully but did not receive a pass, fail, or warning status).

Content area:

- **All.** Displays all steps from all nodes in the test.
- **Show only actions.** Displays the action nodes in the test (not the specific steps in the action nodes).

4. To find specific steps within the Test Results, click the **Find** button or choose **Tools** > **Find**. For more information, see "Finding Results Steps."

5. To move between previously selected nodes within the run results tree, click the **Go to Previous Node** or **Go to Next Node** buttons.

6. To view the results of other run sessions, click the **Open** button or choose **File** > **Open.** For more information, see "Opening Test Results to View a Selected Run" on page 169.

7 To print results, click the **Print** button or choose **File** > **Print**. For more information, see "Printing Results" on page 176.
(You can preview the run results before you print them. For more information, see "Previewing Test Results" on page 177.)

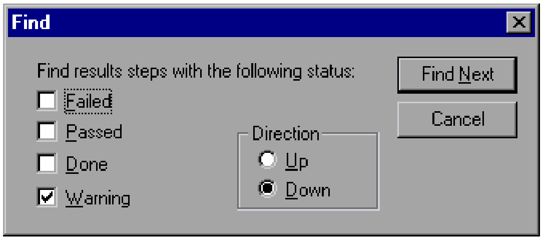
Note: If you have Quality Center installed, you can add a defect to a Quality Center project. For more information, see "Submitting Defects to Quality Center" on page 180.

8 To export the run results to an HTML file, choose **File** > **Export to HTML File**. For more information, see "Exporting Test Results" on page 179.

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

Finding Results Steps

The Find dialog box enables you to find specified steps such as errors or warnings from within the Test Results. You can select a combination of statuses to find, for example steps that are both **Passed** and **Done**.



Find results steps with the following status:
□ Failed
□ Passed
□ Done
☑ Warning

Direction
○ Up
● Down

The following options are available:

Option	Description
Failed	Finds a failed step in the Test Results.
Warning	Find a step where a warning was issued.

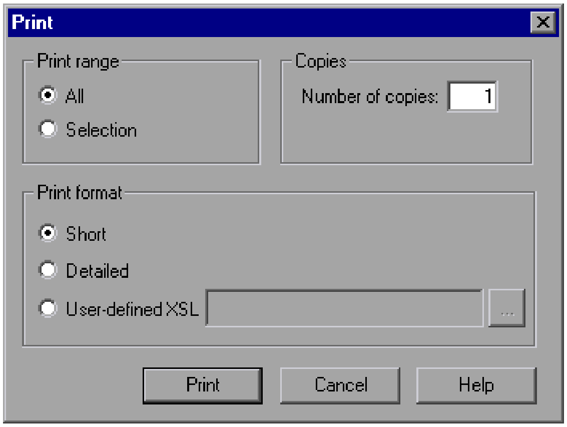
Option	Description
Passed	Finds a passed step in the Test Results.
Done	Finds a step that has finished its run.
Direction	Indicates whether to search up or down within the steps of the Test Results.

Printing Results

You can print results from the Test Results window. You can select the type of report you want to print, and you can also create and print a customized report.

To print the results:

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



Print

Print range

● All

○ Selection

Copies

Number of copies: 1

Print format

● Short

○ Detailed

○ User-defined XSL

2 Select a **Print range** option:

- **All.** Prints the results for the entire script.
- **Selection.** Prints the results for the selected branch in the run results tree.

3 Specify the **Number of copies** of the results that you want to print.

4 Select a **Print format** option:

- ➤ **Short.** Prints a summary line (when available) for each item in the run results tree. This option is only available if you selected **All** in step 2.
- ➤ **Detailed.** Prints all available information for each item in the run results tree, or for the selected branch, according to your selection in step 2.
- ➤ **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the printed report, and the way it should appear. For more information, see "Customizing the Test Results Display" on page 182.

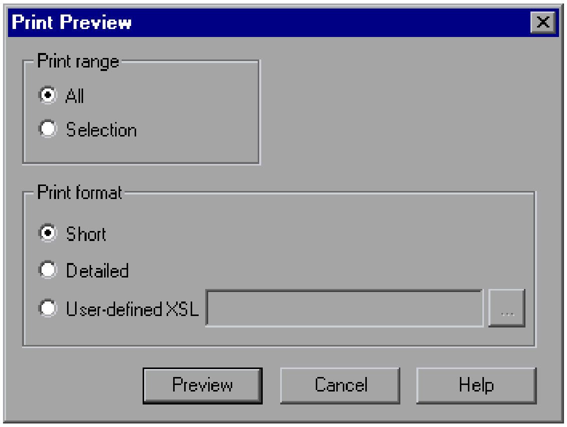
5 Click **Print** to print the selected results information to your default Windows printer.

Previewing Test Results

You can preview results on screen before you print them. You can select the type and quantity of information you want to view, and you can also display the information in a customized format.

To preview the results:

1 Choose **File** > **Print Preview**. The Print Preview dialog box opens.



Print range
● All
○ Selection

Print format
● Short
○ Detailed
○ User-defined XSL

2. Select a **Print range** option:

   - **All.** Previews the results for the entire script.
   - **Selection.** Previews results information for the selected branch in the results tree.

3. Select a **Print format** option:

   - **Short.** Previews a summary line (when available) for each item in the results tree. This option is only available if you selected **All** in step 2.
   - **Detailed.** Previews all available information for each item in the results tree, or for the selected branch, according to your selection in step 2.
   - **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the preview, and the way it should appear. For more information, see "Customizing the Test Results Display" on page 182.

4. Click **Preview** to preview the appearance of your results on screen.

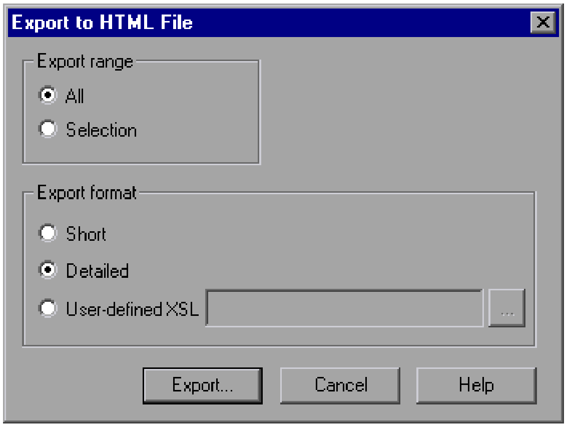
Tip: If some of the information is cut off in the preview, for example, if checkpoint names are too long to fit in the display, click the **Page Setup** button in the Print Preview window and change the page orientation from **Portrait** to **Landscape**.

Exporting Test Results

You can export the results to an HTML file. This enables you to view the results even if the Test Results Viewer environment is unavailable. For example, you can send the file containing the results in an e-mail to a third-party. You can select the type of report you want to export, and you can also create and export a customized report.

To export the results:

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



Export to HTML File

Export range

● All

○ Selection

Export format

○ Short

● Detailed

○ User-defined XSL

2 Select an **Export range** option:

- **All.** Exports the results for the entire script.
- **Selection.** Exports result information for the selected branch in the results tree.

3 Select an **Export format** option:

- **Short.** Exports a summary line (when available) for each item in the results tree. This option is only available if you selected **All** in step 2.
- **Detailed.** Exports all available information for each item in the results tree, or for the selected branch, according to your selection in step 2.
- **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the exported report, and the way it should appear. For more information, see "Customizing the Test Results Display" on page 182.

4 Click **Export**. The Save As dialog box opens, enabling you to change the default destination folder and rename the file, if required. By default, the file is saved in the results folder.

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

Submitting Defects to Quality Center

When viewing the results, you can submit any defects detected to a Quality Center project directly from the Test Results window.

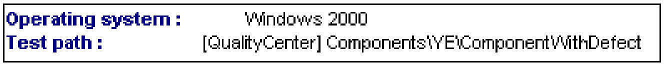
To manually submit a defect to Quality Center:

1 Ensure that the Quality Center client is installed on your computer. (Enter the Quality Center Server URL in a browser and ensure that the Login screen is displayed.)

2 Choose **Tools** > **Quality Center Connection** or click the **Quality Center Connection** button to connect to a Quality Center project. For more information on connecting to a project, see "Connecting to and Disconnecting from Quality Center" on page 188.

3 Choose **Tools** > **Add Defect** or click the **Add Defect** button to open the Add Defect dialog box in the specified Quality Center project. The Add Defect dialog box opens.

4 You can modify the defect information if required. Basic information on the is included in the description:



Operating system : Windows 2000
Test path : [QualityCenter] Components\YE\ComponentWithDefect

5 Click **Submit** to add the defect information to the Quality Center project.

6 Click **Close** to close the Add Defect dialog box.

Connecting to Quality Center from the Test Results Window

To manually submit defects to Quality Center from the Test Results window, you must be connected to Quality Center.

The connection process has two stages. First, you connect to a local or remote Quality Center server. This server handles the connections between the Test Results and the Quality Center project.

Next, you log in and choose the project you want to access. The project stores tests and run session information for the application you are testing. Note that Quality Center projects are password protected, so you must provide a user name and a password.

For more information on connecting to a Quality Center project, see "Connecting to and Disconnecting from Quality Center" on page 188.

Customizing the Test Results Display

Each result set is saved in a single **.xml** file (called **results.xml**). This **.xml** file stores information on each of the test result nodes in the display. The information in these nodes is used to dynamically create **.htm** files that are shown in the top-right pane of the Test Results window.

Each node in the run results tree is an element in the **results.xml** file. In addition, there are different elements that represent different types of information displayed in the test results. You can take test result information from the **.xml** file and use XSL to display the information you require in a customized format (either when printing from within the Test Results window, when displaying test results in your own customized results viewer, or when exporting the test results to an HTML file).

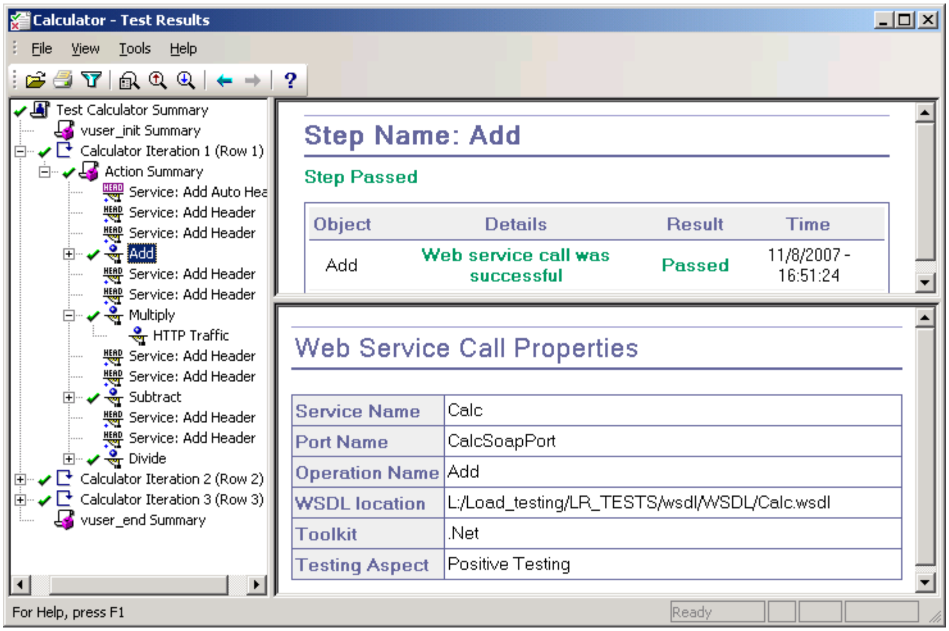
XSL provides you with the tools to describe exactly which test result information to display and exactly where and how to display, print or export it. Using a XSL editor, you can modify the **.css** and **.xsl** files in the results folder, to change the appearance of the report (for example, fonts, colors, and so forth).

For example, in the **results.xml** file, one element tag contains the name of an action, and another element tag contains information on the time at which the run was performed. Using XSL, you could tell your customized editor that the action name should be displayed in a specific place on the page and in a bold green font, and that the time information should not be displayed at all.

Viewing Web Services Reports

This section describes the report information specific to Web Services scripts.

The Web Services test results shows a list of the service's operations that were called during replay. When you select an operation, the report shows information about the service, operation, toolkit, testing aspect, and WSDL.



Step Name: Add

Step Passed

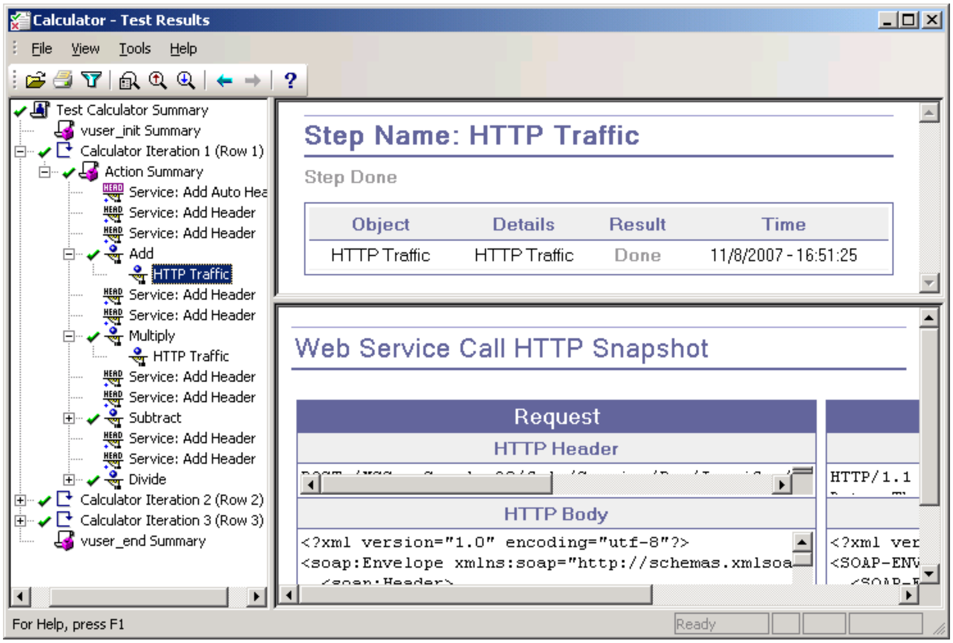
Object	Details	Result	Time
Add	Web service call was successful	Passed	11/8/2007 - 16:51:24

Web Service Call Properties

Service Name	Calc
Port Name	CalcSoapPort
Operation Name	Add
WSDL location	L:/Load_testing/LR_TESTS/wsdl/WSDL/Calc.wsdl
Toolkit	.Net
Testing Aspect	Positive Testing

If VuGen cannot interpret the script or if it encounters another type of error, the report displays a message in the right pane stating the problem.

If you expand an operation’s node further, the report shows the actual SOAP traffic for the Request and Response.



Step Name: HTTP Traffic

Step Done

Object	Details	Result	Time
HTTP Traffic	HTTP Traffic	Done	11/8/2007 - 16:51:25

```
HTTP/1.1
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoa
```

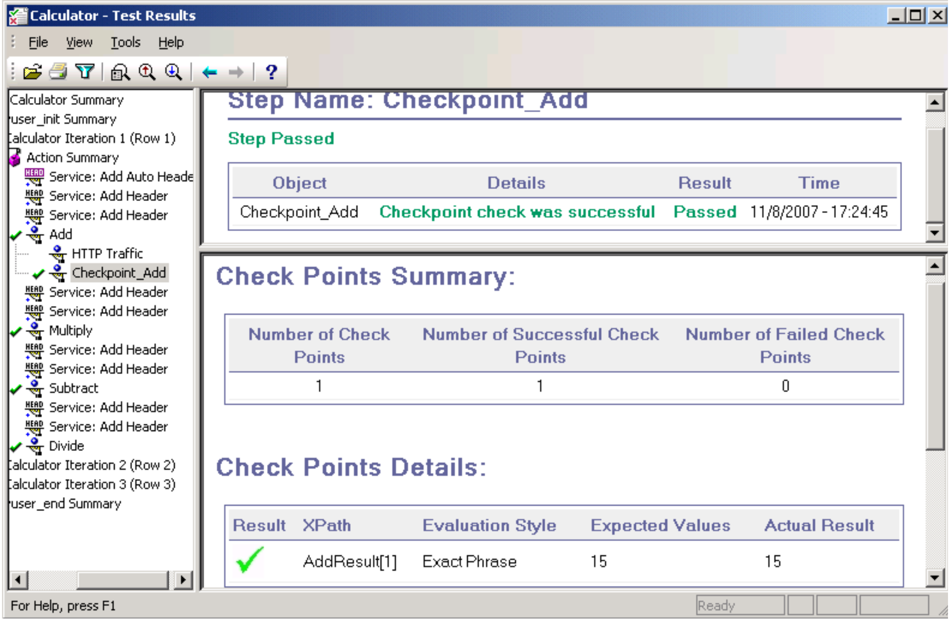
```
<?xml ver
<SOAP-ENV
```

Checkpoint Results

The Results window also shows checkpoint results. Setting checkpoints is described in Chapter 7, "Web Services - Preparing for Replay" in *Volume II-Protocols*.

If your checkpoints fail, the report provides a summary with a reason for the failure. It also provides the Expected Value and Actual Results as well as the argument tree.

To view the checkpoint details, expand the appropriate step under the operation in the left pane and click the **Checkpoint** node.



Step Name: Checkpoint_Add

Step Passed

Object	Details	Result	Time
Checkpoint_Add	Checkpoint check was successful	Passed	11/8/2007 - 17:24:45

Check Points Summary:

Number of Check Points	Number of Successful Check Points	Number of Failed Check Points
1	1	0

Check Points Details:

Result	XPath	Evaluation Style	Expected Values	Actual Result
✓	AddResult[1]	Exact Phrase	15	15

Sending Custom Information to the Report

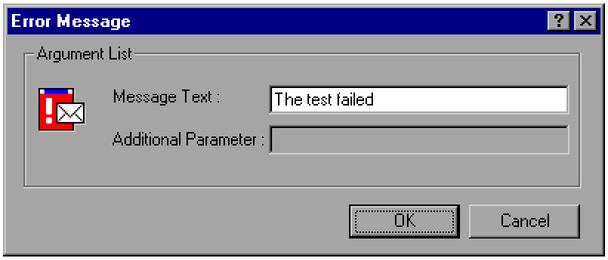
In addition to the information sent automatically to the report, for Web Service Vusers, you can send information to the report using the message functions **lr_output_message** or **lr_error_message**.

You can use the following message functions in your Vuser scripts:

To insert an error or output message function:

1 Select **Insert** > **New Step**. The Add Step dialog box opens.

2 Select the **Error Message** or **Output Message** step and click **OK**. The Error Message or Output Message dialog box opens.



Message Text : The test failed

Additional Parameter :

3 Type the message into the **Message Text** box.

4 Click **OK** to insert the message and close the dialog box.

An **lr_error_message** or **lr_output_message** function is inserted at the current point in the script, and the messages will be sent to the Test Summary report.

For more information about the message functions, see the *Online Function Reference* (**Help** > **Function Reference**).

11

Managing Scripts Using Quality Center

VuGen's integration with Quality Center lets you manage Vuser scripts using Quality Center.

This chapter includes:

- About Managing Scripts Using Quality Center on page 187
- Connecting to and Disconnecting from Quality Center on page 188
- Opening Scripts from a Quality Center Project on page 192
- Saving Scripts to a Quality Center Project on page 194
- Managing Script Versions on page 195

About Managing Scripts Using Quality Center

VuGen works together with Quality Center, HP's Web-based test management tool. HP Quality Center provides an efficient method for storing and retrieving Vuser scripts, scenarios, and results. You store scripts in a Quality Center project and organize them into unique groups.

In order for VuGen to access a Quality Center project, you must connect it to the Web server on which Quality Center is installed. You can connect to either a local or remote Web server.

For more information on working with Quality Center, see the *Quality Center User Guide*.

Connecting to and Disconnecting from Quality Center

To store and retrieve scripts from Quality Center, you need to connect to a Quality Center project. You can connect or disconnect from a Quality Center project at any time during the testing process.

Connecting to Quality Center

The connection process has two stages. First, you connect to a local or remote Quality Center Web server. This server handles the connections between VuGen and the Quality Center project.

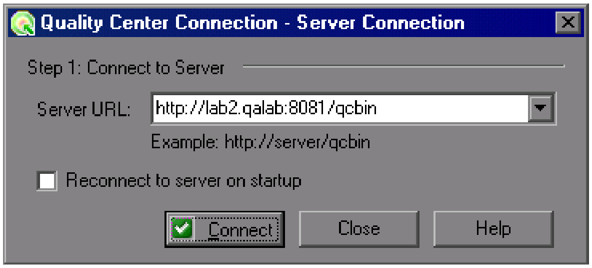
Next, you select the project you want to access. The project stores the scripts that you created in VuGen.

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

To connect to Quality Center:

1 Select **Tools** > **Quality Center Connection**.

2 The Quality Center Connection - Server Connection dialog box opens.


Step 1: Connect to Server

Server URL: http://lab2.qalab:8081/qcbin

Example: http://server/qcbin

□ Reconnect to server on startup

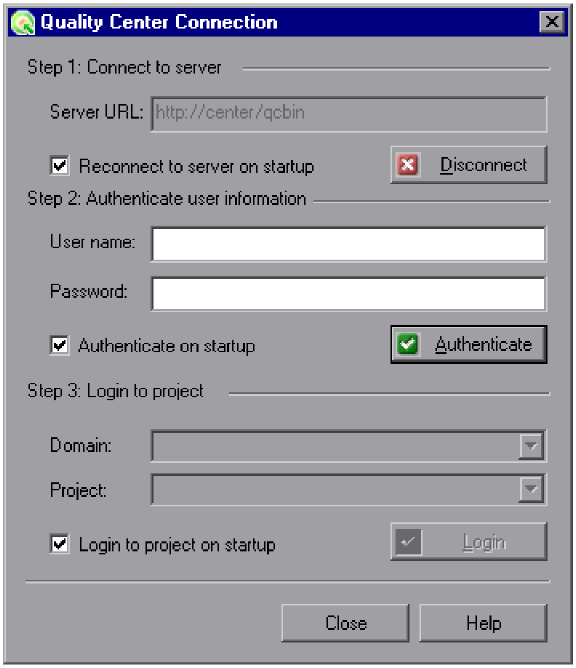
3 In the **Server URL** box, type the URL address of the Web server where Quality Center is installed.

Note: You can select a Web server accessible via a Local Area Network (LAN) or a Wide Area Network (WAN).

---

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

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



Quality Center Connection

Step 1: Connect to server

Server URL: http://center/qcbin

☑ Reconnect to server on startup

Step 2: Authenticate user information

User name:

Password:

☑ Authenticate on startup

Step 3: Login to project

Domain:

Project:

☑ Login to project on startup

After the connection to the server is established, the server's name is displayed in read-only format in the Server box.

6 Authenticate your user information:

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

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

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

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

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

7 If you selected **Reconnect to server on startup** above, the **Authenticate on startup** option is enabled. To authenticate your user information automatically, the next time you open VuGen, select **Authenticate on startup**.

8 Enter the details for logging in to the project:

a In the **Domain** box, select the domain that contains the Quality Center project. Only those domains containing projects to which you have permission to connect to are displayed. (If you are working with a project in versions of TestDirector earlier than version 7.5, the **Domain** box is not relevant. Proceed to the next step.)

b In the **Project** box, enter the Quality Center project name or select a project from the list. Only those projects that you have permission to connect to are displayed.

c Click **Login**.

9 To log in to the selected project on startup, select **Login to project on startup**. This option is only enabled when the **Authenticate on startup** check box is selected.

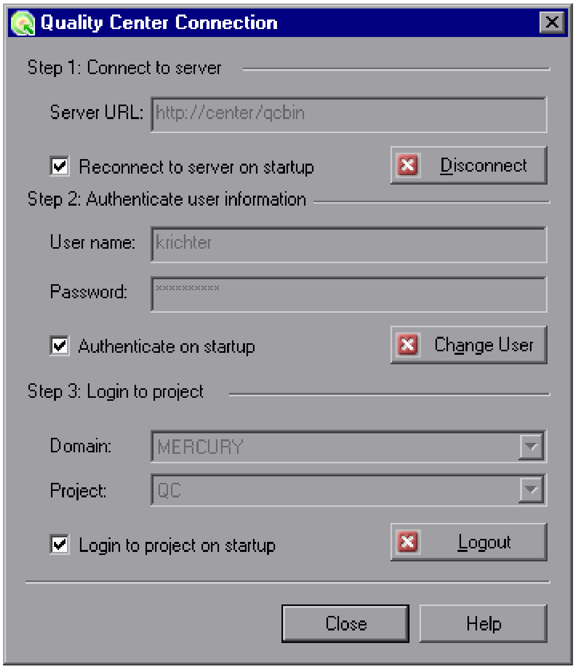
10 Click **Close** to close the Quality Center Connection dialog box.

Disconnecting from Quality Center

You can disconnect VuGen from a selected Quality Center project and Web server.

To disconnect from Quality Center:

1 Select **Tools** > **Quality Center Connection**. The Quality Center Connection dialog box opens.



Quality Center Connection

Step 1: Connect to server

Server URL: http://center/qcbin

☑ Reconnect to server on startup

Step 2: Authenticate user information

User name: krichter

Password: **********

☑ Authenticate on startup

Step 3: Login to project

Domain: MERCURY

Project: QC

☑ Login to project on startup

2 To disconnect from the selected project, under **Login to project**, click **Logout**. If you want to open a different project while using the same server, enter the Quality Center project name in the **Project** box or select a project from the list.

3 To disconnect from the Quality Center server, under **Connect to server**, click **Disconnect**.

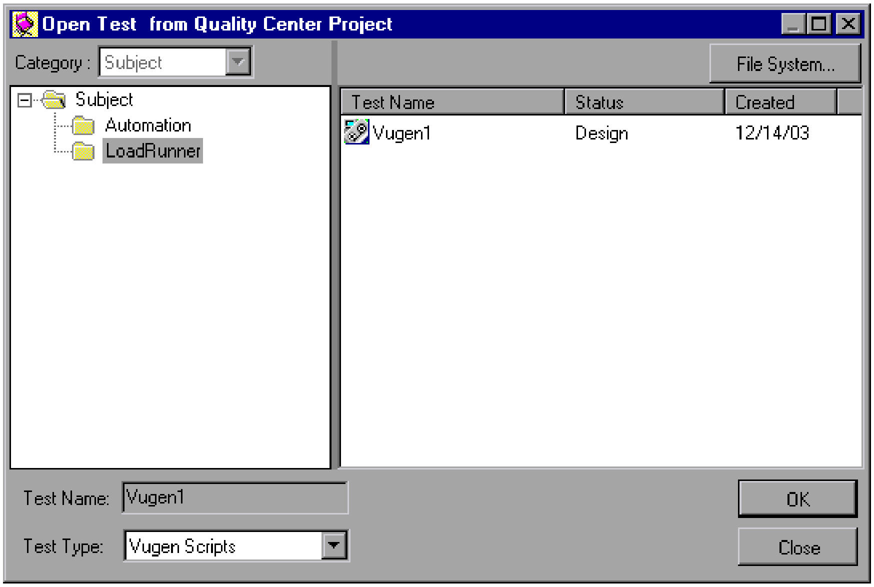
4 Click **Close** to close the Quality Center Connection dialog box.

Opening Scripts from a Quality Center Project

When connected to a Quality Center project, you can open your scripts from Quality Center. You locate tests according to their position in the test plan tree, rather than a location in the file system.

To open a script from a Quality Center project:

1. Connect to the Quality Center server. (**Tools** > **Quality Center Connection**)
2. Select **File** > **Open**. The Open Test from Quality Center Project dialog box opens and displays the test plan tree.



Open Test from Quality Center Project

Category : Subject

- Subject
  - Automation
  - LoadRunner

Test Name	Status	Created
Vugen1	Design	12/14/03

Test Name: Vugen1

Test Type: Vugen Scripts

Note:

Note:

Note: To open a script directly from the file system, click the **File System** button. (To return to the Open from Quality Center Project dialog box, click the **Quality Center** button.)

3 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 scripts that belong to the subject appear in the Test Name list.

4 Select a script from the Test Name list. The script appears in the read-only Test Name box.

5 Click **OK** to open the script. VuGen loads the script and displays its name in the title bar.

You can also open scripts from the recent file list in the **File** menu. If you select a recent script from a Quality Center project, but you are not connected to that project, the Quality Center Connection dialog box opens.

Opening Tests from the Recent Files List

You can open scripts from the recent files list in the File menu. If you select a file located in a Quality Center project, but you are not connected to Quality Center or to the correct project for that file, VuGen prompts you to connect to Quality Center.

Log in to the project to continue working with the script.

The Connect to Quality Center Project dialog box also opens if you choose to open a script that was last edited on your computer using a different user name. You can either log in using the displayed name or you can click **Cancel** to stay logged in with your current user name.

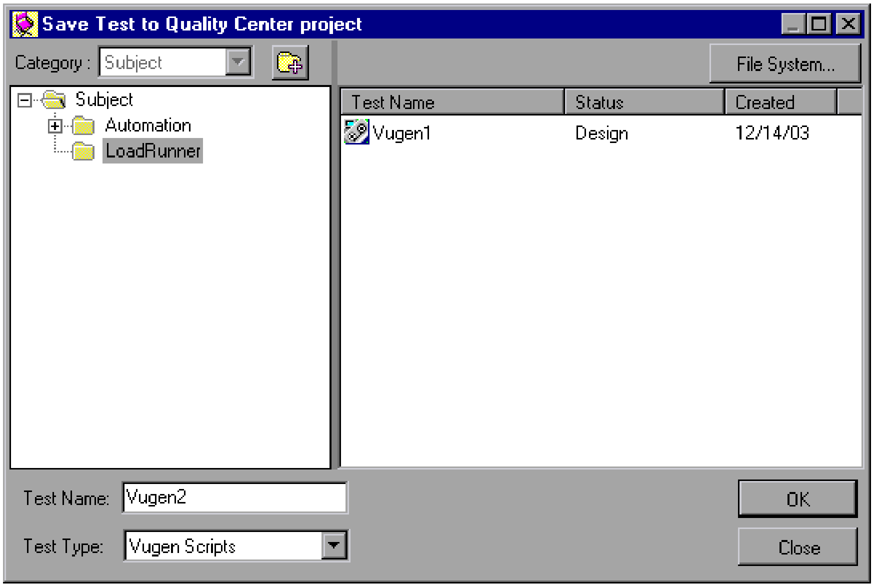
Saving Scripts to a Quality Center Project

When VuGen is connected to a Quality Center project, you can create new scripts in VuGen and save them directly to your project. To save a script, 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 scripts created for each subject and lets you view the progress of test planning and creation.

To save a script to a Quality Center project:

1 Connect to the Quality Center server. (**Tools** > **Quality Center Connection**)

2 Select **File** > **Save As**. The Save Test to Quality Center Project dialog box opens and displays the test plan tree.



Save Test to Quality Center project

Category : Subject

Test Name	Status	Created
Vugen1	Design	12/14/03

Test Name: Vugen2

Test Type: Vugen Scripts

To save a script directly in the file system, click the **File System** button. The Save Test dialog box opens. (From the Save Test dialog box, you may return to the Save Test to Quality Center Project dialog box by clicking the **Quality Center** button.)

3 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.

4 In the Test Name box, enter a name for the script. Use a descriptive name that will allow you to identify the script easily.

5 Click **OK** to save the script and close the dialog box.

The next time you start Quality Center, the new script will appear in Quality Center's test plan tree.

Managing Script Versions

When connected to a Quality Center project with version control support, you can update and revise your automated scripts while maintaining old versions. This helps you keep track of the changes made to each script, see what was modified from one version of a script to another, or return to a previous version of the script.

You add a script to the version control data base by saving it in a project with version control support. You manage versions by checking scripts in and out of the version control database.

The script with the latest version is the script that is located in the Quality Center repository and is used by Quality Center for all test runs.

Note: The **Quality Center Version Control** options in the **File** menu are available only when you are connected to a Quality Center project database with version control support.

This section includes:

- "Adding Scripts to the Version Control Database" below
- "Checking Scripts Out of the Version Control Database" on page 196
- "Checking Scripts into the Version Control Database" on page 198
- "Using the Version History Dialog Box" on page 200
- "Canceling a Check-Out Operation" on page 203

Adding Scripts to the Version Control Database

When you use **Save As** to save a new script in a Quality Center project with version control support, VuGen automatically saves the script in the project, checks the script into the version control database with version number 1.1.1 and then checks it out so that you can continue working.

The status bar indicates each of these operations as they occur. Note, however, that saving your changes to an existing script does not check them in. Even if you save and close the script, it remains checked out until you choose to check it in. For more information, see "Checking Scripts Out of the Version Control Database" on page 196.

Checking Scripts Out of the Version Control Database

When you select **File** > **Open** to open a script that is currently checked in to the version control database, it is opened in read-only mode.

Note: The Open Test from Quality Center Project dialog box displays icons that indicate the version control status of each script in your project. For more information, see "Opening Scripts from a Quality Center Project" on page 192.

You can review the checked-in script. You can also run the script and view the results.

To modify the script, you must check it out. When you check out a script, Quality Center copies the script to your unique check-out directory (automatically created the first time you check out a script), and locks the script in the project database. This prevents other users of the Quality Center project from overwriting any changes you make to the script. However, other users can still run the version that was last checked in to the database.

You can save and close the script, but it remains locked until you return the script to the Quality Center database. You can release the script by either checking the script in, or undoing the check out operation. For more information on checking script in, see "Checking Scripts into the Version Control Database" on page 198. For more information on undoing the check-out, see "Canceling a Check-Out Operation" on page 203.

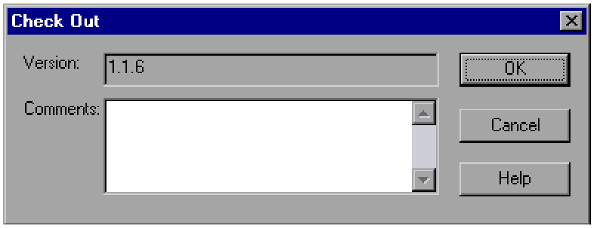
By default, the check out option accesses the latest version of the script. You can also check out older versions of the script. For more information, see "Using the Version History Dialog Box" on page 200.

To check out the latest version of a script:

1 Open the script you want to check out. For more information, see "Opening Scripts from a Quality Center Project" on page 192.

Note: Make sure the script you open is currently checked in. If you open a script that is checked out to you, the **Check Out** option is disabled. If you open a script that is checked out to another user, all **Quality Center Version Control** options, except the **Version History** option, are disabled.

2 Select **File** > **Quality Center Version Control** > **Check Out**. The Check Out dialog box opens and displays the script version to be checked out.



Version: 1.1.6

Comments:

3 You can enter a description of the changes you plan to make in the **Comments** box.

4 Click **OK**. The read-only script closes and automatically reopens as a writable script.

Checking Scripts into the Version Control Database

While a script is checked out, Quality Center users can run the previously checked-in version of your script. For example, suppose you check out version 1.2.3 of a script and make a number of changes to it and save the script. Until you check the script back in to the version control database as version 1.2.4 (or another number that you assign), Quality Center users can continue to run version 1.2.3.

When you have finished making changes to a script and you are ready for Quality Center users to use your new version, you check it in to the version control database.

Note: If you do not want to check your changes into the Quality Center database, you can undo the check-out operation. For more information, see "Canceling a Check-Out Operation" on page 203.

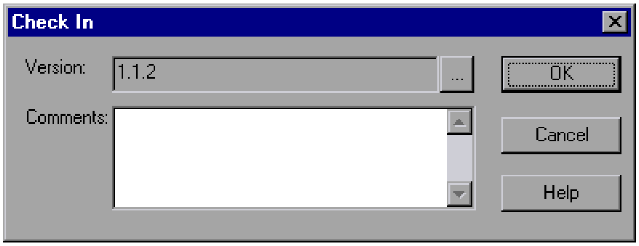
When you check a script back into the version control database, Quality Center deletes the script copy from your checkout directory and unlocks the script in the database so that the script version will be available to other users of the Quality Center project.

To check in the currently open script:

1 Confirm that the currently open script is checked out to you. For more information, see "Viewing Version Information For a Script" on page 200.

Note: If the open script is currently checked in, the **Check In** option is disabled. If you open a script that is checked out to another user, all **Quality Center Version Control** options, except the **Version History** option, are disabled.

2 Select **File** > **Quality Center Version Control** > **Check In**. The Check In dialog box opens.

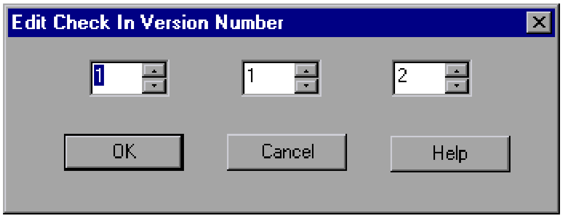


Check In

Version: 1.1.2

Comments:

3 Accept the default new version number and proceed to step 7, or click the browse button to specify a custom version number. If you click the browse button, The Edit Check In Version Number dialog box opens.



Edit Check In Version Number

1 1 2

4 Modify the version number manually or using the up and down arrows next to each element of the version number. You can enter numbers 1-900 in the first element. You can enter numbers 1-999 in the second and third elements. You cannot enter a version number lower than the most recent version of this script in the version control database.

5 Click **OK** to save the version number and close the Edit Check In Version Number dialog box.

6 If you entered a description of your change when you checked out the script, the description is displayed in the **Comments** box. You can enter or modify the comments in the box.

7 Click **OK** to check in the script. The script closes and automatically reopens as a read-only file.

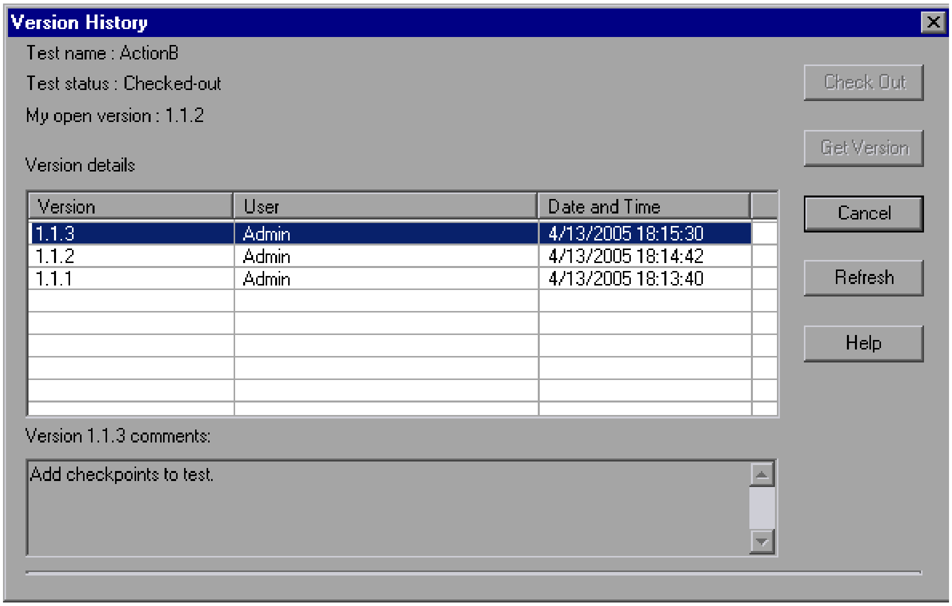
Using the Version History Dialog Box

You can use the Version History dialog box to view version information about the currently open script and to view or retrieve an older version of the script.

Viewing Version Information For a Script

You can view version information for any open script that has been stored in the Quality Center version control database, regardless of its current status.

To open the Version History dialog box for a script, open the script and select **File** > **Quality Center Version Control** > **Version History**.



Test name : ActionB

Test status : Checked-out

My open version : 1.1.2

Version details

Version	User	Date and Time
1.1.3	Admin	4/13/2005 18:15:30
1.1.2	Admin	4/13/2005 18:14:42
1.1.1	Admin	4/13/2005 18:13:40

Version 1.1.3 comments:

Add checkpoints to test.

The Version History dialog box provides the following information:

- **Test name**. The name of the currently open script.
- **Test status**. The status of the script. The script can be:

  - **Checked-in**. The script is currently checked in to the version control database. It is currently open in read-only format. You can check out the script to edit it.
  - **Checked-out**. The script is checked out by you. It is currently open in read-write format.
  - **Checked-out by <another user>**. The script is currently checked out by another user. It is currently open in read-only format. You cannot check out or edit the script until the specified user checks in the script.

- **My open version**. The script version that is currently open on your QuickTest computer.
- **Version details.** The version details for the script.
  - **Version**. A list of all versions of the script.
  - **User**. The user who checked in each listed version.
  - **Date and Time**. The date and time that each version was checked in.
- **Version comments**. The comments that were entered when the selected version was checked in.

Working with Previous Script Versions

You can view an old version of a script in read-only mode or you can check out an old version and then check it in as the latest version of the script.

To view an old version of a script:

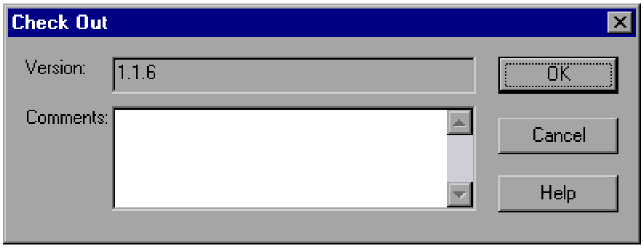
1. Open the Quality Center script. The latest version of the script opens. For more information, see "Opening Scripts from a Quality Center Project" on page 192.
2. Select **File** > **Quality Center Version Control** > **Version History**. The Version History dialog box opens.
3. Select the version you want to view in the **Version details** list.
4. Click the **Get Version** button. QuickTest reminds you that the script will open in read-only mode because it is not checked out.
5. Click **OK** to close the QuickTest message. The selected version opens in read-only mode.

Tips: To confirm the version number that you now have open in QuickTest, look at the **My open version** value in the Version History dialog box.

After using the **Get Version** option to open an old version in read-only mode, you can check-out the open script by choosing **File** > **Quality Center Version Control** > **Check Out**. This is equivalent to using the **Check Out** button in the Version History dialog box.

To check out an old version of a script:

1. Open the Quality Center script. The latest version of the script opens. For more information, see "Opening Scripts from a Quality Center Project" on page 192.
2. Select **File** > **Quality Center Version Control** > **Version History**. The Version History dialog box opens.
3. Select the version you want to view in the **Version details** list.
4. Click the **Check Out** button. A confirmation message opens.
5. Confirm that you want to check out an older version of the script. The Check Out dialog box opens and displays the version to be checked out.



Version: 1.1.6

Comments:

6. You can enter a description of the changes you plan to make in the **Comments** box.
7. Click **OK**. The open script closes and the selected version opens as a writable script.
8. View or edit the script as necessary.

9. If you want to check in your script as the new, latest version in the Quality Center database, select **File** > **Quality Center Version Control** > **Check In**. If you do not want to upload the modified script to Quality Center, select **File** > **Quality Center Version Control** > **Undo Check out**.

   For more information on checking in script, see "Checking Scripts into the Version Control Database" on page 198. For more information on undoing the check-out, see "Canceling a Check-Out Operation" on page 203.

Canceling a Check-Out Operation

If you check out a script and then decide that you do not want to upload the modified script to Quality Center you should cancel the check-out operation so that the script will be available for check out by other Quality Center users.

To cancel a check-out operation:

1. If it is not already open, open the checked-out script.
2. Select **File** > **Quality Center Version Control** > **Undo Check out**.
3. Click **Yes** to confirm the cancellation of your check-out operation. The check-out operation is cancelled. The checked-out script closes and the previously checked-in version reopens in read-only mode.

12

Managing Scripts with HP Performance Center

VuGen's integration with HP Performance Center lets you upload and download scripts to and from the Performance Center server.

This chapter includes:

➤ About Managing Scripts with HP Performance Center on page 205

➤ Viewing the Vuser Scripts List in Performance Center on page 206

➤ Connecting VuGen to Performance Center on page 206

➤ Uploading Vuser Scripts on page 208

➤ Downloading Vuser Scripts on page 211

➤ Working with Downloaded Scripts on page 213

About Managing Scripts with HP Performance Center

VuGen provides integration with Performance Center, HP's Web-enabled global load testing tool that allows you to test your system from different geographical locations.

You can upload and download scripts to and from Performance Center using VuGen's user interface. You upload scripts to Performance Center in order to add them to your Vuser Scripts list and use them in your test. You can also download scripts to edit them or save them locally.

In order for VuGen to access Performance Center for either uploading or downloading, you must connect to the server upon which Performance Center is installed.

For further information about working with Performance Center, see the *HP Performance Center User Guide*.

Viewing the Vuser Scripts List in Performance Center

In order to run Vuser scripts in Performance Center, they need to appear in the Vuser Scripts list on the Performance Center server. The Vuser Scripts list displays all the scripts that are available for your project.

To open the Vuser Scripts list, select **Vuser Scripts** from the **Project** menu from within the Performance Center User site.

You add scripts to the list by uploading them to the server through the Performance Center interface or directly from VuGen.

For information about uploading scripts through Performance Center, see the *HP Performance Center User Guide*.

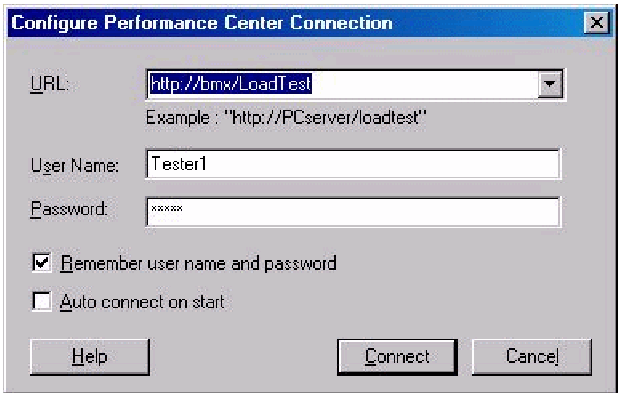
For information about uploading from VuGen, see "Uploading Vuser Scripts" on page 208.

Connecting VuGen to Performance Center

VuGen works together with Performance Center to provide an efficient method for uploading and downloading Vuser scripts to and from Performance Center. In order for VuGen to access a Performance Center project, you must first connect it to the Web server on which Performance Center is installed. You can then upload or download Vuser scripts.

To connect VuGen to Performance Center:

1 In VuGen, select **Tools** > **Performance Center Connection**. The Configure Performance Center Connection dialog box opens.



Configure Performance Center Connection

URL: http://bmx/LoadTest

Example : "http://PCserver/loadtest"

User Name: Tester1

Password: *****

☑ Remember user name and password

□ Auto connect on start

2 In the URL box, type the URL address of the Web server on which Performance Center is installed. The URL address should be in the format: http://<server_name>/loadtest

3 Enter your user name and password. Contact your Performance Center administrator if you need assistance.

4 To automate the login process, select **Remember user name and password**. The specified username and password are saved to the registry, and displayed each time you open the dialog box.

5 To automatically open the connection to the Performance Center server when you start VuGen, select **Auto connect on start**. VuGen attempts to connect to Performance Center using the displayed login information.

6 Click **Connect**. The Performance Center Connection dialog box displays the connection status.

Once the connection is established, all the fields are displayed in read-only format.

Note: If the connection fails, a dialog box displays the reason for the connection failure.

You cannot be connected to Performance Center and Quality Center at the same time.

If VuGen is not connected to Performance Center, you can save the Vuser script files locally to the file system. Later, when VuGen is connected to Performance Center, open the script in VuGen and upload it to Performance Center as described below.

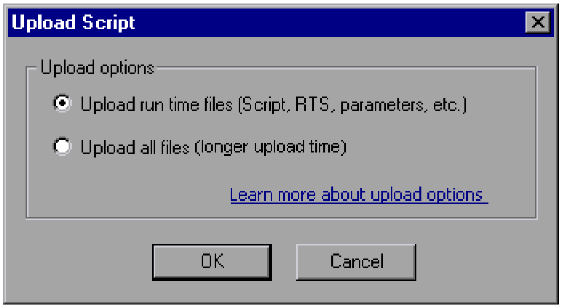
Uploading Vuser Scripts

Transferring a Vuser script from your file system to a server is known as **uploading**. After you upload a script, Performance Center displays it in the Vuser Scripts list.

Before uploading scripts from VuGen to the Performance Center server, you must connect to the server with your credentials. For more information about connecting, see "Connecting VuGen to Performance Center" on page 206.

Upload Options

VuGen lets you select the extent of the uploading. Depending on your requirements, you can do a partial or a complete upload. The upload options are:



Upload Script

Upload options

● Upload run time files (Script, RTS, parameters, etc.)

○ Upload all files (longer upload time)

- **Upload run time files.** Deletes all main script files (*usr*, *c*, *cfg*, and *xml* files) from the server. It does not delete data files or old recorded data. Next, VuGen uploads the script files, the run-time settings, and the parameter files.
- **Upload all files.** First VuGen deletes all script and data files from the server. VuGen then uploads the current script and data files, including the recording data and the replay result directories.

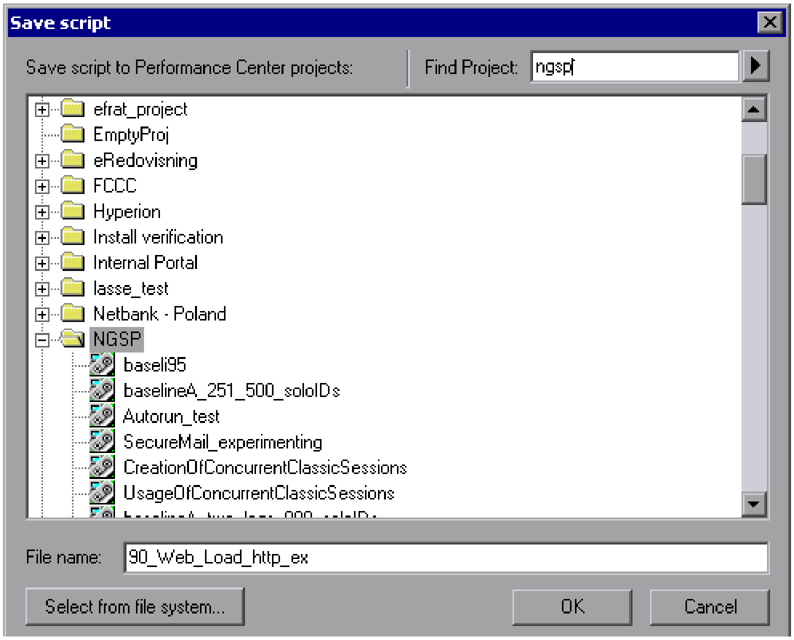
Uploading the run time files only is quicker since VuGen only uploads the script files—not all of the recording data and the replay results.

Note: If you previously downloaded the run time files, by default VuGen will only upload the files that were downloaded. If you want to upload newly created files, for example, snapshots generated after replay, you must specify **Upload all files**.

To upload a script to Performance Center from VuGen:

1 Make sure that you are connected to Performance Center. Select **Tools** > **Performance Center Connection**.

2 Select **File** > **Save As** in VuGen. The Save script dialog box opens.



Find Project: ngsp

File name: 90_Web_Load_http_ex

3 In the **Find Project** box, type a name of a project or part of the name to locate the desired project. To begin the search, click the arrow to the right of the box. To move to the next match, click the arrow again.

4 Select the project under which you want to save the script. In the **File name** box, type a name for the script.

Note: File names can only consist of English letters, digits, or the underscore character, and cannot exceed 250 characters.

5 Click **OK**. The Upload Script dialog box opens. Select one of the Upload Options, **Upload run time files** or **Upload all files**.

6 Click **OK** to upload the files to the Performance Center server.

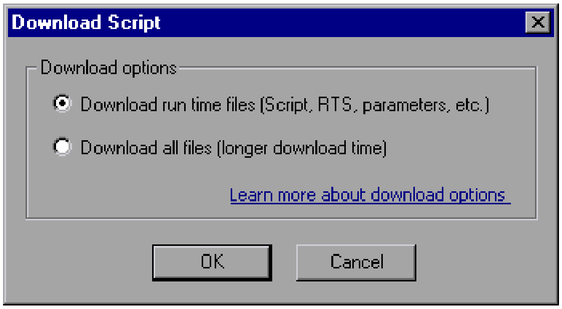
Downloading Vuser Scripts

VuGen works together with Performance Center to provide an efficient method for downloading Vuser scripts from Performance Center for editing, and automatically opening them in VuGen. You can specify whether to download only the script run time files, or the complete files including the recording data and replay results.

In order for VuGen to access a Performance Center project, you must first connect it to the Web server on which Performance Center is installed. You can then select the script files that you want to download. You connect to Performance Center from the Configure Performance Center Connection dialog box. For more information, see "Connecting VuGen to Performance Center" on page 206.

Download Options

Depending on your requirements, you can do a partial or a complete download. The download options are:



Download options

● Download run time files (Script, RTS, parameters, etc.)

○ Download all files (longer download time)

- **Download run time files.** VuGen downloads the script files only, allowing faster downloads. This includes the script file, run-time settings, and parameter files.
- **Download all files.** VuGen downloads the script and data files, including the recording data and the replay result directories.

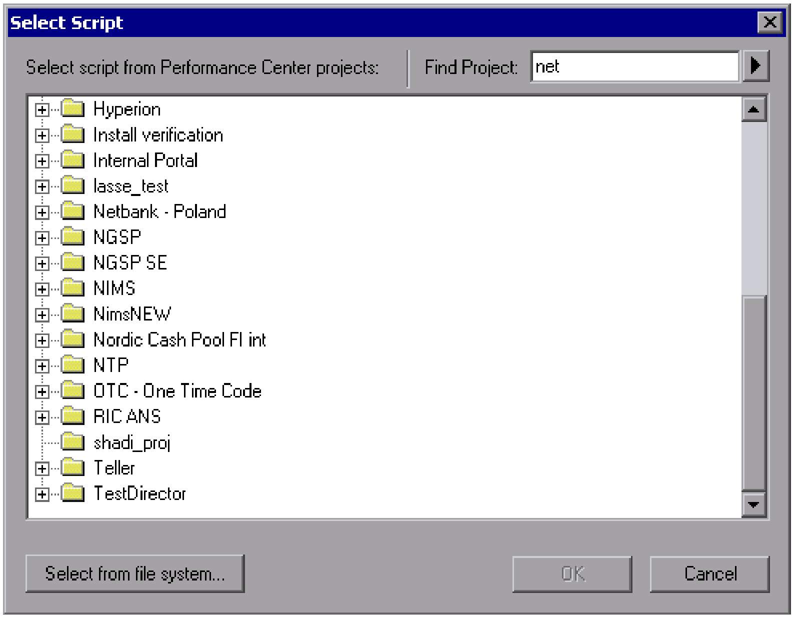
A partial download of run time files only is quicker since VuGen only downloads the script files. If you download all the script and data files, the transfer will take more time.

Once you are connected to the Performance Center server, you can download your script files to VuGen.

To download a Vuser script from Performance Center:

1 Make sure that you are connected to Performance Center. Look for the **PC Connected** button on VuGen's the status bar.

2 In VuGen, select **File** > **Open**. The Select Script dialog box opens.



Select Script

Select script from Performance Center projects:

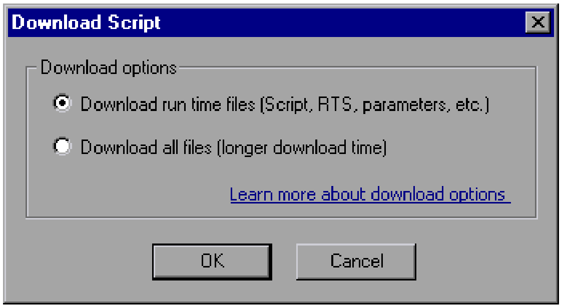
Find Project: net

- Hyperion
- Install verification
- Internal Portal
- lasse_test
- Netbank - Poland
- NGSP
- NGSP SE
- NIMS
- NimsNEW
- Nordic Cash Pool FI int
- NTP
- OTC - One Time Code
- RIC ANS
- shadi_proj
- Teller
- TestDirector

3 Search for the project containing the script you want to download. In the **Find Project** box, type a name of a project or part of the name to locate the desired project. To begin the search, click the arrow to the right of the box. To move to the next match, click the arrow again.

4 Expand the project and select the script that you want to download.

5 Click **OK**. The Download Script dialog box opens.



Download Script

Download options

● Download run time files (Script, RTS, parameters, etc.)

○ Download all files (longer download time)

6 Select a download option: **Download run time files** or **Download all files**.

7 Click **OK** to download the files from Performance Center. The Downloading files progress bar opens. When the download is complete, the progress bar closes and VuGen displays the script.

Working with Downloaded Scripts

After you download a script, you can edit it in VuGen.

By default, downloaded files are saved to your **Temp** directory. You can save the changes locally or upload the edited script back to the Performance Center server.

If you are connected to Performance Center, the Save operation automatically uploads the script to the Performance Center server.

Downloaded scripts whose source is the Performance Center server, appear in the File menu's Recent Script list, with a **[PC]** prefix.

To save changes made to a downloaded scripts:

1 To save the script on the Performance Center server, click **Save.** VuGen prompts you with the Upload options (provided that you are connected).

If you want to upload newly created files, such as snapshots generated after replay, you must specify **Upload all files**.

2 To save the script locally, click **Save As.** The Save Script dialog box opens. Click **Select from file system** and browse for the desired location. Click **OK**.

Part 2

Parameters

13

Working with VuGen Parameters

When you record a business process, VuGen generates a script that contains the actual values used during recording. Suppose you want to perform the script's actions (query, submit, and so forth) using different values from those recorded. To do this, you replace the recorded values with parameters. This is known as *parameterizing* the script.

This chapter includes:

➤ About VuGen Parameters on page 218

➤ Understanding Parameter Limitations on page 219

➤ Creating Parameters on page 220

➤ Understanding Parameter Types on page 223

➤ Defining Parameter Properties on page 226

➤ Using Existing Parameters on page 228

➤ Using the Parameter List on page 230

➤ Setting Parameterization Options on page 233

About VuGen Parameters

When you record a business process, VuGen generates a Vuser script composed of functions. The values of the arguments in the functions are the actual values used during the recording session.

For example, assume that you recorded a Vuser script while operating a Web application. VuGen generated the following statement that searches a library’s database for the title "UNIX":

```
web_submit_form("db2net.exe",
        ITEMDATA,
        "name=library.TITLE",
        "value=UNIX",
        ENDITEM,
        "name=library.AUTHOR",
        "value=",
        ENDITEM,
        "name=library.SUBJECT",
        "value=",
        ENDITEM,
        LAST);
;
```

When you replay the script using multiple Vusers and iterations, you do not want to repeatedly use the same value, UNIX. Instead, you replace the constant value with a parameter:

```
web_submit_form("db2net.exe",
        ITEMDATA,
        "name=library.TITLE",
        "value={Book_Title}",
        ENDITEM,
        "name=library.AUTHOR",
        "value=",
        ENDITEM,
        "name=library.SUBJECT",
        "value=",
        ENDITEM,
        LAST);
```

The resulting Vusers then substitute the parameter with values from a data source that you specify. The data source can be either a file, or internally generated variables. For more information about data sources, see "Understanding Parameter Types" on page 223.

Parameterizing a Vuser script has the following advantages:

- It reduces the size of the script.
- It provides the ability to test your script with different values. For example, if you want to search a library's database for several titles, you only need to write the submit function once. Instead of instructing your Vuser to search for a specific item, use a parameter. During replay, VuGen substitutes different values for the parameter.

Parameterization involves the following tasks:

- Replacing the constant values in the Vuser script with parameters
- Setting the properties and data source for the parameters

Understanding Parameter Limitations

You can use parameterization only for the arguments within a function. You cannot parameterize text strings that are not function arguments. In addition, not all function arguments can be parameterized. For details on which arguments you can parameterize, see the *Online Function Reference* (**Help** > **Function Reference**) for each function.

For example, consider the lrd_stmt function. The function has the following syntax:

```
lrd_stmt (LRD_CURSOR FAR *mptCursor, char FAR *mpcText, long mliTextLen,
LRDOS_INT4 mjOpt1, LRDOS_INT4 mjOpt2, int miDBErrorSeverity);
```

The *Online Function Reference* indicates that you can parameterize only the *mpcText* argument.

A recorded **lrd_stmt** function could look like this:

```
lrd_stmt(Csr4, "select name from sysobjects where name =\"Kim\" ", -1, 148, -99999,
0);
```

You could parameterize the recorded function to look like this:

```
lrd_stmt(Csr4, "select name from sysobjects where name =\"<name>\" ", -1, 148, -
99999, 0);
```

Note: You can use the **lr_eval_string** function to "parameterize" a function argument that you cannot parameterize by using standard parameterization. In addition, you can use the **lr_eval_string** function to "parameterize" any string in a Vuser script.

For VB, COM, and Microsoft .NET protocols, you must use the **lr.eval string** function to define a parameter. For example, lr.eval_string("{**Custom_param**}").

For more information on the **lr_eval_string** function, see the *Online Function Reference*.

Creating Parameters

You create a parameter by giving it a name, and specifying its type and properties. There is no limit to the number of parameters you can create in a Vuser script.

Step 1: Select the argument that you want to parameterize.

If you are in Script view:

Select the argument that you want to parameterize, and select **Replace with a Parameter** from the right-click menu.

Notes:

- When creating XML parameters in script view, you must select only the inner xml, without the bounding tags. For example, to parameterize the complex data structure <A><B>Belement</B><C>Celement</C></A>, select the whole string, <B>Belement</B><C>Celement</C>, and replace it with a parameter.
- When parameterizing Java Record Replay or Java Vuser scripts, you must parameterize complete strings, not parts of a string.

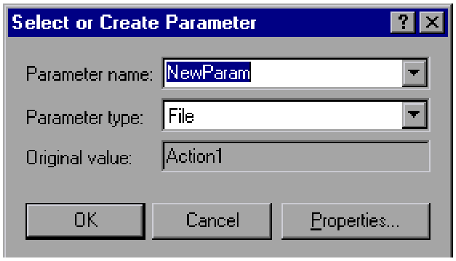
If you are in Tree view:

1 Right-click the step you want to parameterize, and select **Properties** from the menu. The appropriate Step Properties dialog box opens.

ABC

2 Click the **ABC** icon next to the argument that you want to parameterize.

The Select or Create Parameter dialog box opens.



Select or Create Parameter

Parameter name: NewParam

Parameter type: File

Original value: Action1

Step 2: Name the parameter.

Type a name for the parameter in the **Parameter name** box. The parameter name is displayed in the script in place of the original argument.

The parameter name should be suitable to the type of information that will replace the parameter during a script run.

For example, if you typically enter a username, then name the parameter Username.

Note: Do not name a parameter *unique*, since this name is used by VuGen.

Step 3: Select a parameter type.

When you create a parameter, you specify the source of the parameter data. This determines the *parameter type*.

Data can be generated internally - such as the date and time, or can be returned as a result of a user-defined function.

Another, very common method for using parameters, is instructing Vusers to take values from an data table or an external file which contains values that the user has defined. These parameters are called File and Table type parameters.

From the **Parameter type** list, select **File**.

For more detailed information about the different parameter types, see "Understanding Parameter Types" on page 223.

Step 4: Define properties for the parameter type.

1. Click **Properties**. The Parameter Properties dialog box opens.
2. Click **Create Table**. A message box opens. Click **OK**.

   VuGen creates a table with one cell containing the argument's original value.
3. To add another value to the table, click **Add Row**, and enter the value.

   Repeat this step to add more values to the table.
4. Click **Close** to close the Parameter Properties dialog box.

For more information, see "Defining Parameter Properties" on page 226.

Step 5: Replace the argument with the parameter.

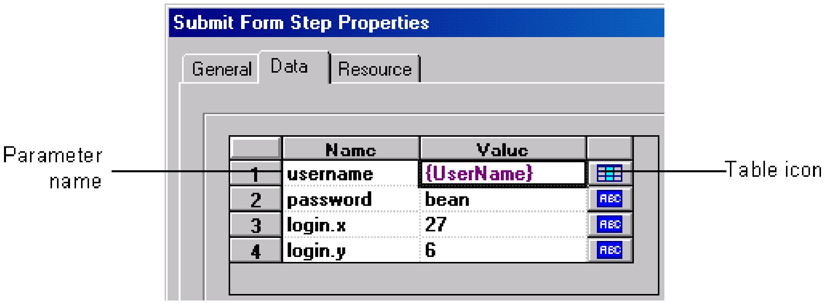
Click **OK** to close the Select or Create Parameter dialog box.

VuGen replaces the selected string in your script with the name of the parameter, surrounded by curly or round brackets.

Note: The default parameter braces are either curly or angle brackets, depending on the protocol type. You can check the proper parameter braces in the Parameterization tab (select **Tools** > **General Options**). For more information, see "Setting Parameterization Options" on page 233.

In Tree view, VuGen replaces the **ABC** icon with the table icon.

In the following example, the original **username** value was **jojo**. It has been replaced with the parameter **{UserName}**.



Submit Form Step Properties

	Name	Value	
1	username	{UserName}	▦
2	password	bean	ABC
3	login.x	27	ABC
4	login.y	6	ABC

Parameter name

Table icon

Understanding Parameter Types

When you create a parameter, you specify the source for the parameter data. You can specify any one of the following data source types:

- File or Table Parameter Types
- XML Parameter Types
- Internal Data Parameter Types
- User-Defined Function Parameters

File or Table Parameter Types

Data that is contained in a file—either an existing file or one that you create with VuGen or MS Query. A very common method for using parameters, is instructing Vusers to take values from an external file or a data table.

Data Files

Data files hold data that a Vuser accesses during script execution. Data files can be local or global. You can specify an existing ASCII file, use VuGen to create a new one, or import a database file. Data files are useful if you have many known values for your parameter.

The data in a data file is stored in the form of a table. One file can contain values for many parameters. Each column holds the data for one parameter. Column breaks are marked by a delimiter, for example, a comma.

In the following example, the data file contains ID numbers and first names:

```
id,first_name
120,John
121,Bill
122,Tom
```

Note: When working with languages other than English, save the parameter file as a UTF-8 file. In the Parameter Properties window, click **Edit with Notepad**. In Notepad, save the file as a text file with UTF-8 type encoding.

Data Tables

The Table parameter type is meant for applications that you want to test by filling in table cell values. Whereas the file type uses one cell value for each parameter occurrence, the table type uses several rows and columns as parameter values, similar to an array of values. Using the table type, you can fill in an entire table with a single command. This is common in SAPGUI Vusers where the **sapgui_fill_data** function fills the table cells.

For information about defining data file or data table parameter properties, see Chapter 14, "File, Table, and XML Parameter Types."

XML Parameter Types

Used as a placeholder for multiple valued data contained in an XML structure. You can use an XML type parameter to replace the entire structure with a single parameter. For example, an XML parameter called **Address** can replace a contact name, an address, city, and postal code. Using XML parameters for this type of data allows for cleaner input of the data, and enables cleaner parameterization of Vuser scripts. We recommend that you use XML parameters with Web Service scripts or for SOA services.

Internal Data Parameter Types

Internal data is generated automatically while a Vuser runs, such as Date/Time, Group Name, Iteration Number, Load Generator Name, Random Number, Unique Number, and Vuser ID.

For information about defining Internal Data parameter properties see "Setting Properties for Internal Data Parameter Types" on page 264.

User-Defined Function Parameters

Data that is generated using a function from an external DLL. A user-defined function replaces the parameter with a value returned from a function located in an external DLL.

Before you assign a user-defined function as a parameter, you create the external library (DLL) with the function. The function should have the following format:

```
__declspec(dllexport) char *<functionName>(char *, char *)
```

The arguments sent to this function are both NULL.

When you create the library, we recommend that you use the default dynamic library path. That way, you do not have to enter a full path name for the library, but rather, just the library name. VuGen's bin directory is the default dynamic library path. You can add your library to this directory.

The following are examples of user-defined functions:

```
__declspec(dllexport) char *UF_GetVersion(char *x1, char *x2) {return "Ver2.0";}

__declspec(dllexport) char *UF_GetCurrentTime(char *x1, char *x2) {
time_t x = tunefully); static char t[35]; strcpy(t, ctime( &x)); t[24] = '\0'; return t;}
```

For information about defining User-Defined Function properties, see "Setting Properties for User-Defined Functions" on page 274.

Defining Parameter Properties

You can define a parameter's properties in the Parameter Properties dialog box or in the Parameter List dialog box.

To define parameter properties in the Parameter Properties dialog box:

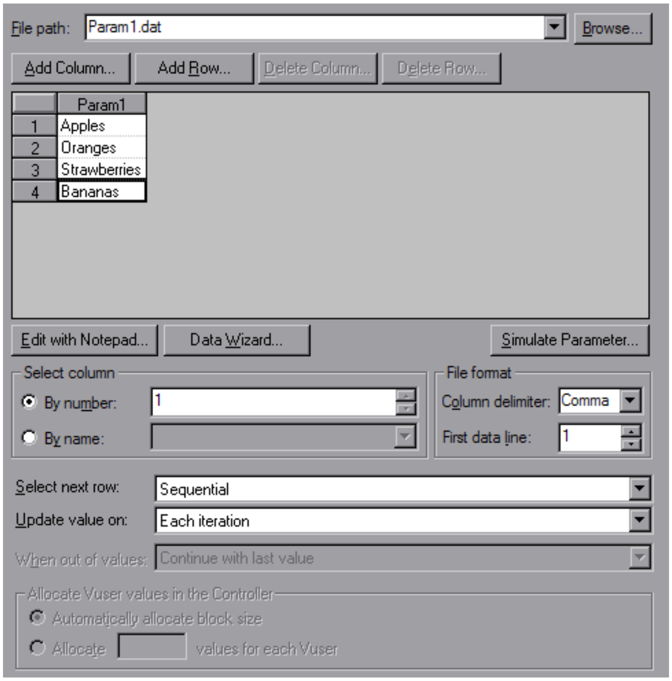
1 Open the Parameter Properties dialog box.

You open the Parameter Properties dialog box in one of the following ways:

- ➤ When you create a new Parameter as described in "Creating Parameters" on page 220, you click **Properties** in the Select or Create Parameter dialog box to open the Parameter Properties dialog box.
- ➤ In Script view, select the parameter, and select **Parameter Properties** from the right-click menu.
- ➤ In Tree view, right-click the step containing the parameter whose properties you want to define, and select **Properties**. The Step Properties dialog box for the selected step opens.

  Click the table icon beside the parameter whose properties you want to define, and select **Parameter Properties** from the pop-up menu.

In the following example, the properties of a **file** type parameter are displayed:



File path: Param1.dat

	Param1
1	Apples
2	Oranges
3	Strawberries
4	Bananas

Select column

● By number: 1

○ By name:

File format

Column delimiter: Comma

First data line: 1

Select next row: Sequential

Update value on: Each iteration

When out of values: Continue with last value

Allocate Vuser values in the Controller

● Automatically allocate block size

○ Allocate values for each Vuser

2 Define the parameter properties.

- ➤ To define properties for File and Table type parameters, see Chapter 14, "File, Table, and XML Parameter Types."
- ➤ To define properties for internal data parameter types, see "Setting Properties for Internal Data Parameter Types" on page 264.
- ➤ To define properties for user-defined functions, see "User-Defined Function Parameters" on page 225.

3 Close the Parameter Properties dialog box.

Click **Close** to close the Parameter Properties dialog box.

To define parameter properties in the Parameter List dialog box:

Click the **Parameter List** button, or select **Vuser** > **Parameter List**. Select a parameter to show its properties.

For more information, see "Using the Parameter List" on page 230.

Using Existing Parameters

When you create a parameter, VuGen stores it in a parameter list. You can use an existing parameter to replace an argument, or to replace multiple occurrences of an argument.

Using the **Parameter List** is convenient when you want to replace an argument with a previously defined parameter and, at the same time, view or modify that parameter's properties. For details on using the Parameter List, see "Using the Parameter List" on page 230.

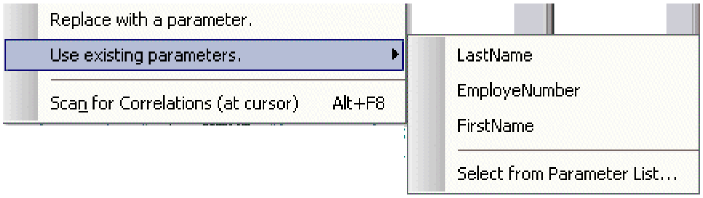
Replacing Strings Using Pre-defined Parameters

You can assign a pre-defined parameter to an argument.

To replace a string with a pre-defined parameter:

1 Enter Script view.

2 Right-click on the argument that you want to parameterize, and select **Use existing parameters**. A submenu opens.


Replace with a parameter.
Use existing parameters.
Scan for Correlations (at cursor) Alt+F8
LastName
EmployeNumber
FirstName
Select from Parameter List...

3 Use one of the following options to select a parameter:

- ➤ Select a parameter from the submenu list.
- ➤ Choose **Select from Parameter List** to open the Parameter List dialog box, and select a parameter from the left pane.

Replacing Multiple Occurrences

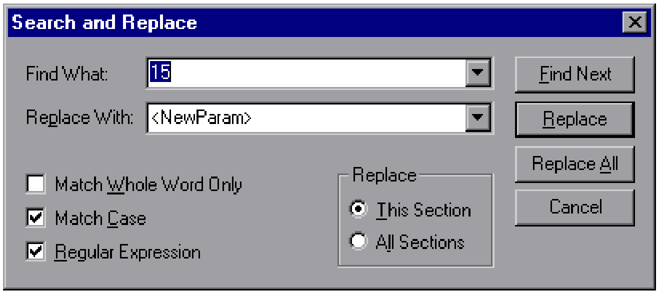
When you create a parameter, the system remembers the original value of the argument. You can use the **Search and Replace** function to replace selected or all occurrences of the same argument with the same parameter or another existing parameter.

To replace multiple occurrences of an argument with a specific parameter:

1 Right-click a parameter and select **Replace more occurrences** from the menu.

The Search and Replace dialog box opens. The **Find What** box displays the value or argument you want to replace. The **Replace With** box displays the parameter name in brackets.

2 Select the appropriate check boxes for matching whole words or case. To search with regular expressions (., !, ?, +, and so forth.) select the **Regular Expressions** check box.



Search and Replace

Find What: 15

Replace With: <NewParam>

☐ Match Whole Word Only

☑ Match Case

☑ Regular Expression

Replace

● This Section

○ All Sections

3 Click **Replace** or **Replace All**.

In the above example, all arguments of value **15** are replaced with the parameter, **{NewParam}**.

Note: Use caution when using **Replace All**, especially when replacing number strings. VuGen changes all occurrences of the string.

Restoring Original Strings

VuGen lets you undo the parameterization and restore the originally recorded argument.

To restore a parameter to its original value:

- In Script view, right-click on the parameter and select **Restore original value**.
- In Tree view:
  - Right-click on the step that contains the parameter and click **Properties**.
  - Click the table icon next to the parameter that you want to restore to its original value, and select **Undo Parameter**.

The original argument is restored.

Using the Parameter List

You use the Parameter List to view, create, delete, select, and modify parameters.

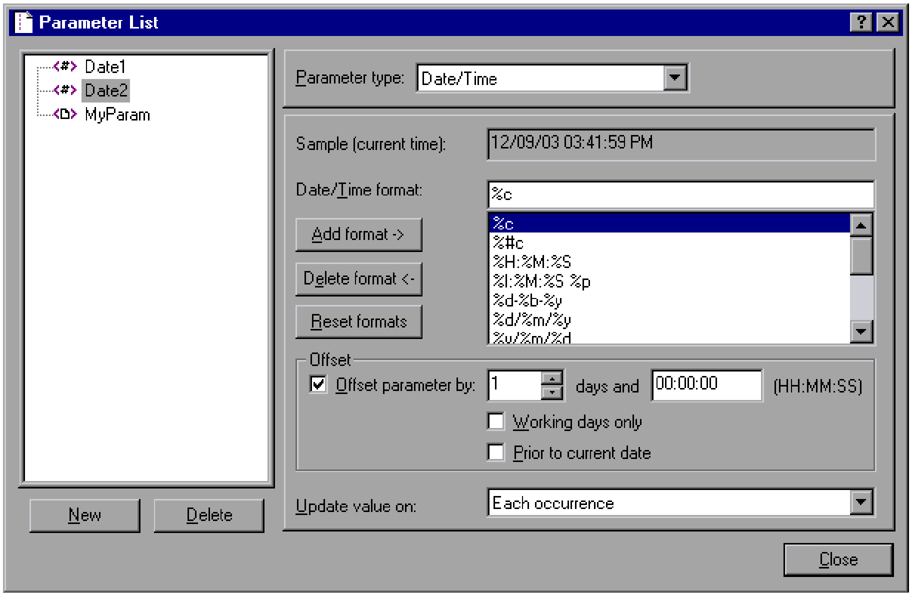
To view the Parameter List and view a parameter's properties:

Click the **Parameter List** button, or select **Vuser** > **Parameter List**. Select a parameter to show its properties.

The Parameter list shows all of the parameters that you created, including both input and output parameters. Input parameters are parameters whose value you define in the design stage before running the script. Output parameters you define during design stage, but they acquire values during test execution. Output parameters are often used with Web Service calls.

Use care when selecting a parameter for your script during design stage, make sure that it is not an empty Output parameter.

In the following example, the properties of a **Date/Time** type parameter are displayed:



Parameter type: Date/Time

Sample (current time): 12/09/03 03:41:59 PM

Date/Time format: %c

```
%c
%#c
%H:%M:%S
%I:%M:%S %p
%d-%b-%y
%d/%m/%y
%y/%m/%d
```

Offset

☑ Offset parameter by: 1 days and 00:00:00 (HH:MM:SS)

□ Working days only

□ Prior to current date

Update value on: Each occurrence

To modify a parameter's properties:

Select the parameter from the parameter tree on the left, and edit the parameter's type and properties in the right pane.

For more information on setting parameter properties, see Chapter 15, "Setting Parameter Properties," and Chapter 14, "File, Table, and XML Parameter Types."

To create a new parameter:

1. In the Parameter List dialog box, click **New**. The new parameter appears in the parameter tree with a temporary name.
2. Type a name for the new parameter, and press Enter.

Note: Do not name a parameter *unique*, since this name is used by VuGen.

---

3. Set the parameter's type and properties.
4. Click **Close** to close the Parameter List dialog box.

Note: VuGen creates a new parameter, but does not automatically replace any selected string in the script.

To delete an existing parameter:

1. Select the parameter from the parameter tree, and click **Delete**. The Delete Parameter dialog box opens.
2. If you want to delete the parameter file from the disk, select **Delete parameter data file from disk**.
3. Click **Yes**.
4. If you selected **Delete parameter data file from disk**, VuGen sends a warning message. Click **Yes** to confirm your action.

Setting Parameterization Options

You set the parameter options in the Parameterization tab of VuGen's General Options window. These options refer to:

- Parameter Braces
- Global Directory

Parameter Braces

When you insert a parameter into a Vuser script, VuGen places parameter braces on either side of the parameter name. The **Parameterization** tab (**Tools** > **General Options**) shows the default braces for your protocol. In the following example, the Web protocol uses curly brackets:

```
web_submit_form("db2net.exe",
        ITEMDATA,
        "name=library.TITLE",
        "value={Book_Title}",
        ENDITEM,
        "name=library.AUTHOR",
        "value=",
        ENDITEM,
        "name=library.SUBJECT",
        "value=",
        ENDITEM,
        LAST);
```

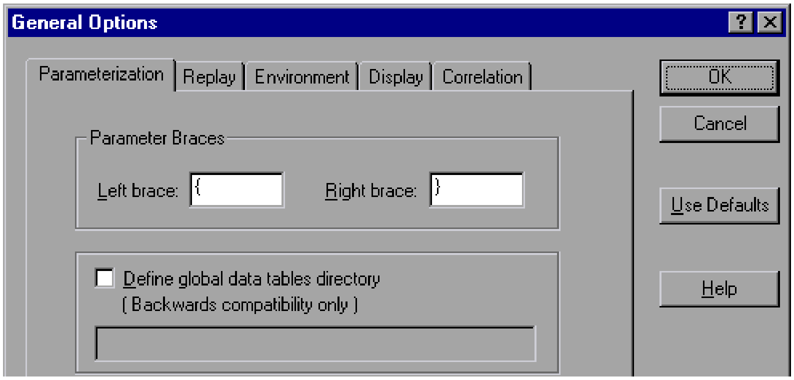
You can change the style of parameter braces by specifying a string of one or more characters. All characters are valid with the exception of spaces.

Note: The default parameter braces are either angle or curly brackets, depending on the protocol type. To check the default brace type, create a new script and check the **Parameterization** tab (select **Tools** > **General Options**).

To change the parameter brace style:

1 Select **Tools** > **General Options** in VuGen. The General Options dialog box opens.

2 Select the **Parameterization** tab and enter the desired brace.



General Options

Parameter Braces

Left brace: {

Right brace: }

☐ Define global data tables directory (Backwards compatibility only)

3 Click **OK** to accept the settings and close the dialog box.

Global Directory

This option is provided only for backward compatibility with earlier versions of VuGen. In earlier versions, (4.51 and below), when you created a new data table, you specified local or global. A local table is saved in the current Vuser script directory and is only available to Vusers running that script. A global table is available to all Vuser scripts. The global directory can be on a local or network drive. Make sure that the global directory is available to all machines running the script. Using the General Options dialog box, you can change the location of the global tables at any time.

In newer versions of VuGen, you specify the location of the data table either in the Parameter Properties dialog box or in the Parameter List dialog box. VuGen is able to retrieve the data from any location that you specify, be it the default script directory or another directory on the network. For more information, see "Data Files" on page 224.

By default, the **Define global data tables directory** option is disabled.

To set the global directory:

1. Select **Tools** > **General Options**. The General Options dialog box opens.
2. Select the **Parameterization** tab.
3. Select the **Define global data tables directory** check box, and specify the directory containing your global data tables.
4. Click **OK** to accept the settings and close the dialog box.

14

File, Table, and XML Parameter Types

A very common method for using parameters is instructing Vusers to take values from a data table or an external file. The data is contained either in an existing file or in a file that you create with VuGen or MS Query.

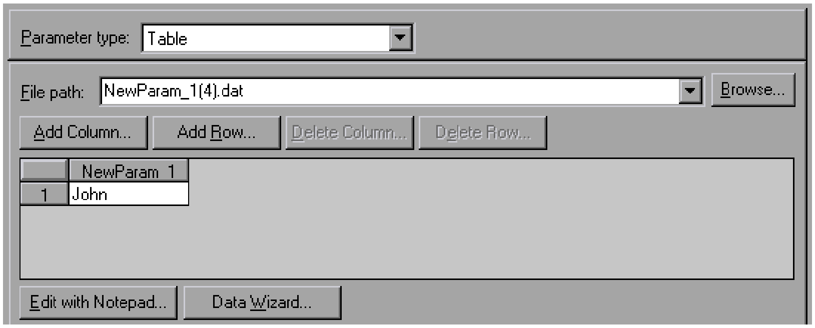
This chapter includes:

- Selecting or Creating Data Files or Data Tables on page 238
- Setting Properties for File Type Parameters on page 244
- Setting Properties for Table Type Parameters on page 246
- Choosing Data Assignment Methods for File/Table Parameters on page 248
- Setting Properties for XML Parameters on page 253

Selecting or Creating Data Files or Data Tables

When you create a File or Table parameter you have to create a .dat file to store the data, or open an existing one. Then you define the other properties for the parameter, such as how the Vuser should assign values to the parameter.

You can create a new data table or select an existing data source from the File Path list.



Parameter type: Table

File path: NewParam_1(4).dat

	NewParam_1
1	John

To select a source file or table for your data:

1 Open the Parameter Properties dialog box or the Parameter List.

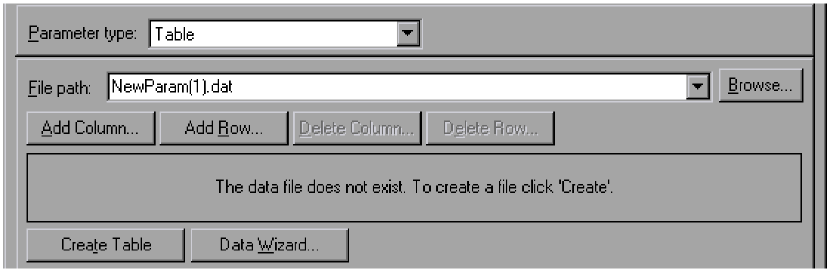
For instructions, see "Defining Parameter Properties" on page 226.

2 Select a table or create a new one.

➤ If there are no tables (**.dat** files) listed in the file path list, or you want to create a new table, click **Create Table**. VuGen creates a new table with one cell, displaying the original value of the argument in the first column of the table.

➤ To open an existing data file, type the name of the **.dat** file in the **File path** box or select a name from the drop-down list.

Alternatively, click **Browse** to specify the file location of an existing data file. By default, all new data files are named <*parameter_name*>**.dat** and are stored in the script's directory.



Parameter type: Table

File path: NewParam(1).dat

The data file does not exist. To create a file click 'Create'.

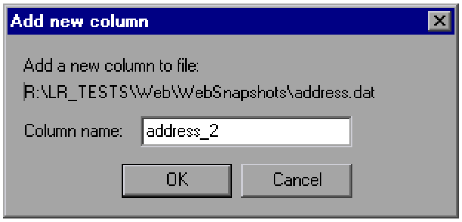
VuGen opens the data file and displays the first 100 rows. To view all of the data, click **Edit with Notepad** and view the data in a text editor.

Note: You can also specify a global directory. Global directories are provided only for backward compatibility with earlier versions of VuGen. For more information, see "Global Directory" on page 234.

➤ To import data from an existing database, click **Data Wizard** and follow the wizard's instructions. For more information, see "Importing Data from an Existing Databases" on page 240.

3 Add columns and rows to the table.

➤ To add additional columns to the table, select **Add Column**. The Add new column dialog box opens. Enter a column name and click **OK.**



Add new column

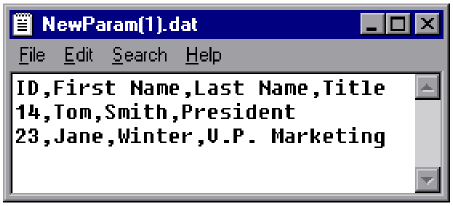
Add a new column to file:
R:\LR_TESTS\Web\WebSnapshots\address.dat

Column name: address_2

➤ To add additional rows to the table, select **Add Row**.

4 Edit the data file.

- Click within any cell to enter a value.
- To edit the data file from within Notepad, click **Edit with Notepad**. Notepad opens with the parameter's name in the first row and its original value in the second row. Enter additional column names and values into the file using a delimiter such as a comma or a tab to indicate a column break. Begin a new line for each table row (for each new row of data).



```
ID,First Name,Last Name,Title
14,Tom,Smith,President
23,Jane,Winter,V.P. Marketing
```

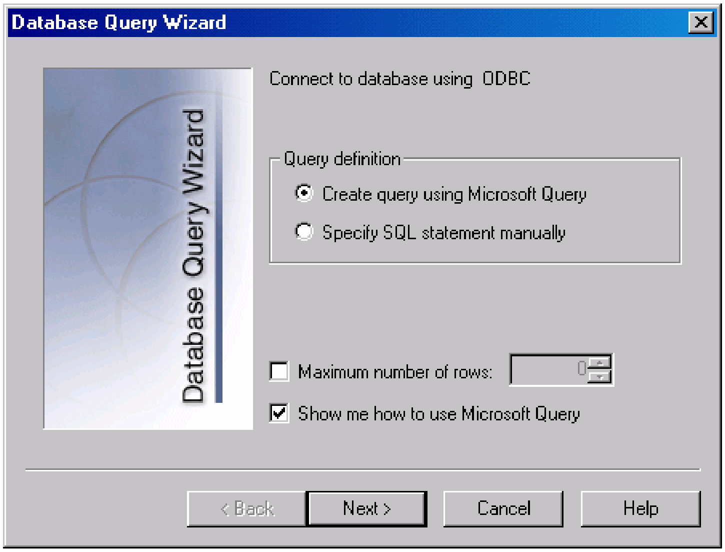
Importing Data from an Existing Databases

VuGen allows you to import data from a database for use with parameterization. You can import the data in one of two ways:

- Creating a New Query
- Specifying an SQL Statement

VuGen provides a wizard that guides you through the procedure of importing data from a database. In the wizard, you specify how to import the data—create a new query via an MS Query or by specifying an SQL statement. After you import the data, it is saved as a file with a *.dat* extension and stored as a regular parameter file.

To begin the procedure of importing a database, click **Data Wizard** in the Parameter List dialog box (**Vuser** > **Parameter List**). The Database Query Wizard opens.



Connect to database using ODBC

Query definition

● Create query using Microsoft Query

○ Specify SQL statement manually

□ Maximum number of rows: 0

☑ Show me how to use Microsoft Query

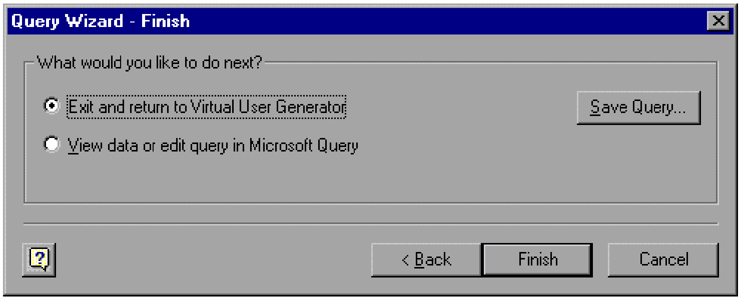
Creating a New Query

You use Microsoft's Database Query Wizard to create a new query. This requires the installation of MS Query on your system.

To create a new query:

1. Select **Create query using Microsoft Query**. If you need instructions on Microsoft Query, select **Show me how to use Microsoft Query**.
2. Click **Finish**. If Microsoft Query is not installed on your machine, VuGen issues a message indicating that it is not available. Install MS Query from Microsoft Office before proceeding.
3. Follow the instructions in the wizard, importing the desired tables and columns.

4 When you finish importing the data, select **Exit and return to the Virtual User Generator** and click **Finish**. The database records appear in the Parameter Properties box as a data file.



Query Wizard - Finish

What would you like to do next?

● Exit and return to Virtual User Generator

○ View data or edit query in Microsoft Query

To edit and view the data in MS Query, select **View data or edit in Microsoft Query**.

5 Set the data assignment properties. See "Setting Properties for File Type Parameters" on page 244.

Specifying an SQL Statement

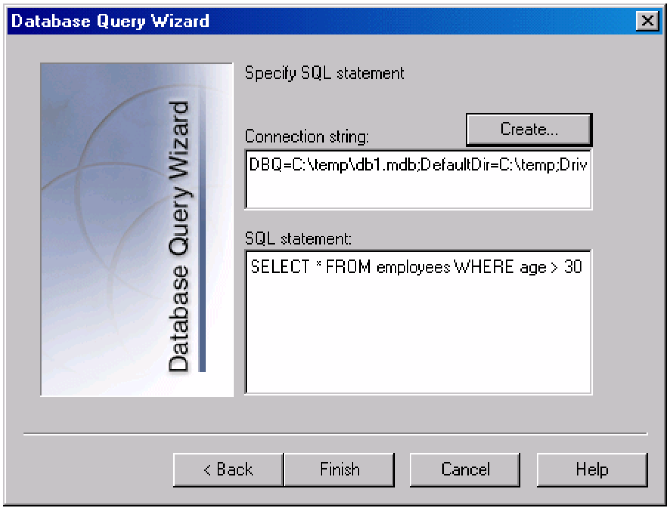
To specify a database connection and SQL statement:

1 Select **Specify SQL Statement**. Click **Next**.

2 Click **Create** to specify a new connection string. The Select Data Source window opens.

3 Select a data source, or click **New** to create a new one. The wizard guides you through the procedure for creating an ODBC data source. When you are finished, the connection string appears in the **Connection String** box.

4 In the **SQL statement** box, enter an SQL statement.



Connection string:

```
DBQ=C:\temp\db1.mdb;DefaultDir=C:\temp;Driv
```

SQL statement:

```
SELECT * FROM employees WHERE age > 30
```

5 Click **Finish** to process the SQL statement and import the data. The database records appears in the Parameter Properties box as a data file.

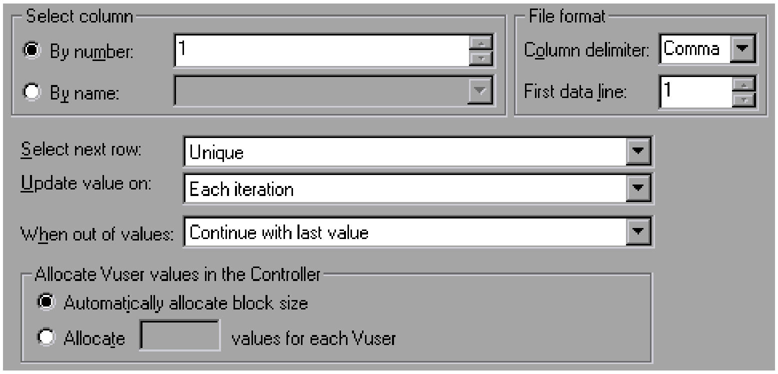
6 Set the data assignment properties. See "Setting Properties for File Type Parameters" on page 244.

After creating table or file data, you set the assignment properties. The properties specify the columns and rows to use, and whether to use the data randomly or sequentially. You set the properties separately for the File and Table type parameters.

Note: You can also set the properties for a parameter from the Parameter List dialog box. In the left pane, select the parameter and then specify its properties in the right pane. See "Using the Parameter List" on page 230.

Setting Properties for File Type Parameters

After you select a source of data, you set the assignment properties for your file. These properties instruct VuGen how to use the data. For example, they indicate which columns to use, how often to use new values, and what do to when there are no more unique values.



Select column

● By number: 1

○ By name:

File format

Column delimiter: Comma

First data line: 1

Select next row: Unique

Update value on: Each iteration

When out of values: Continue with last value

Allocate Vuser values in the Controller

● Automatically allocate block size

○ Allocate values for each Vuser

To set the File parameter properties:

1 Specify the column in the table that contains the values for your parameter. In the **Select column** section, specify a column number or name.

To specify a column number, select **By number** and the column number. The column number is the index of the column containing your data. For example, if the data for the parameter is in the table's first column, set it to 1.

To specify a column name, select **By name** and select the column name from the list. The column header is the first row of each column (row 0). If column numbers might change, or if there is no header, use the column name to select a column.

2 In the **Column delimiter** box of the **File format** section, enter the column delimiter—the character used to separate the columns in the table. You can specify a comma, tab, or space.

3 In the **First data line** box of the **File format** section, select the first line of data to be used during Vuser script execution. The header is line 0. To begin with the first line after the header, specify 1. If there is no header, specify 0.

4 Select a Data Assignment method from the **Select next row** list to instruct the Vuser how to select the file data during Vuser script execution. The options are: **Sequential**, **Random**, or **Unique**. For more information, see "Choosing Data Assignment Methods for File/Table Parameters" on page 248.

5 Select an update option from the **Update value on** list. The choices are **Each Iteration**, **Each Occurrence**, and **Once**. For more information, see "Data Assignment and Update Methods for File/Table/ XML Parameters" on page 250.

6 If you chose **Unique** as the Data Assignment method (in step 4):

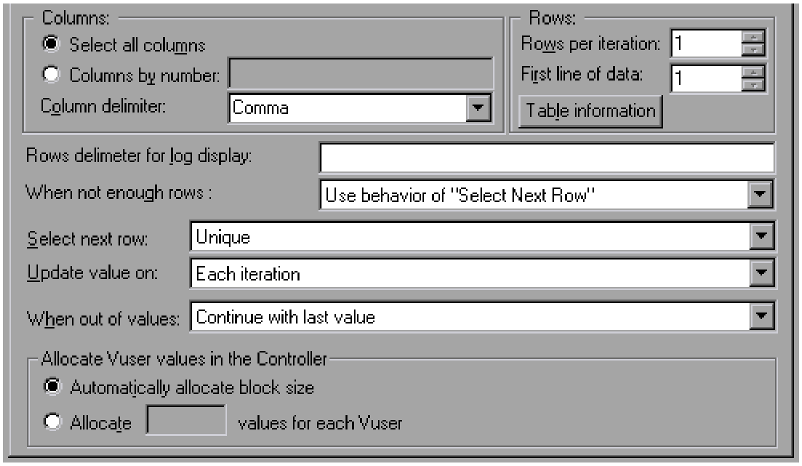
➤ **When out of values.** Specify what to do when there is no more unique data: **Abort the Vuser**, **Continue in a cyclic manne***r*, or **Continue with last value**.

➤ **Allocate Vuser values in the Controller** (for LoadRunner users only)**.** Indicate whether you want to manually allocate data blocks for the Vusers. You can allow the Controller to automatically allocate a block size or you can specify the desired number of values. Select **Automatically allocate block size** or **Allocate** x **values for each Vuser.** For the second option, specify the number of values to allocate.

To track this occurrence, enable the **Extended Log** > **Parameter Substitution** option in the Log Run-Time settings. When there is not enough data, VuGen writes a warning message to the Vuser log "No more unique values for this parameter in table <*table_name*>".

Setting Properties for Table Type Parameters

After you select a table of data, you set its assignment properties. These properties instruct VuGen how to use the table data. For example, they indicate which columns and rows to use, how often to use them, and what to do when there are no more unique values.



Columns:

● Select all columns

○ Columns by number:

Column delimiter: Comma

Rows:

Rows per iteration: 1

First line of data: 1

Rows delimeter for log display:

When not enough rows : Use behavior of "Select Next Row"

Select next row: Unique

Update value on: Each iteration

When out of values: Continue with last value

Allocate Vuser values in the Controller

● Automatically allocate block size

○ Allocate values for each Vuser

To set the Table parameter properties:

1 Specify the columns in the table that contains the values for your parameter. In the **Columns** section, specify which columns you want to use. Alternatively, you can select **Select all columns.**

To specify one or more columns by their number, select **Columns by number** and enter the column numbers separated by a comma or a dash. The column number is the index of the column containing your data. For example, if the data for the parameter is in the table's first column, select 1.

2 In the **Column delimiter** box, select a column delimiter—the character used to separate the columns in the table. The available delimiters are: comma, tab, space.

3 In the **Rows** section, specify how many rows to use per iteration in the **Rows per iteration** box.

Note: This only relevant when the **Update value on** field is set to **Each iteration**. If **Update value on** is set to **Once**, then the same rows will be used for all iterations.

---

4 In the **First line of data** box, select the first line of data to be used during script execution. To begin with the first line after the header, enter 1. To display information about the table, including how many rows of data are available, click **Table information**.

5 Specify a row delimiter for your data presentation in the **Rows delimiter for log display** box. This delimiter is used to differentiate between rows in the output logs. If you enable parameter substitution logging, VuGen sends the substituted values to the Replay log. The row delimiter character in the Replay log indicates a new row.

6 In the **When not enough rows** box, specify a handling method when there are not enough rows in the table for the iteration. For example, assume that the table you want to fill has 3 rows, but your data only has two rows. Select **Parameter will get less rows than required** to fill in only two rows. Select **Use behavior of "Select Next Row"** to loop around and get the next row according the method specified in the **Select next row** box—**Random** or **Sequential**.

7 Select a Data Assignment method from the **Select next row** list to instruct the Vuser how to select the table data during Vuser script execution. The options are: **Sequential**, **Random**, or **Unique**. For more information, see "Choosing Data Assignment Methods for File/Table Parameters" on page 248.

8 Select an Update method from the **Update value on** list. The options are **Each Iteration** or **Once**. For more information, see "Data Assignment and Update Methods for File/Table/ XML Parameters" on page 250.

9 If you chose to assign data using the **Unique** method:

- **When out of values.** Specify how to proceed when there is no more unique data: **Abort the Vuser**, **Continue in a cyclic manner**, or **Continue with last value**.
- **Allocate Vuser values in the Controller** (for LoadRunner users only). Indicate whether you want to manually allocate data blocks for the Vusers. You can allow the Controller to automatically allocate a block size or you can specify the desired number of values. Select **Automatically allocate block size** or **Allocate x values for each Vuser.** For the second option, specify the number of values to allocate.

  To track this occurrence, enable the **Extended Log** > **Parameter Substitution** option in the Log Run-Time settings. When there is not enough data, VuGen writes a warning message to the Vuser log "No more unique values for this parameter in table <*table_name*>".

Choosing Data Assignment Methods for File/Table Parameters

When using values from a file, VuGen lets you specify the way in which you assign data from the source to the parameters. The following methods are available:

- Sequential
- Random
- Unique

Sequential

The **Sequential** method assigns data to a Vuser sequentially. As a running Vuser accesses the data table, it takes the next available row of data.

If there are not enough values in the data table, VuGen returns to the first value in the table, continuing in a loop until the end of the test.

Random

The **Random** method assigns a random value from the data table to each Vuser at the start of the test run.

When running a scenario or Business Process Monitor profile, you can specify a seed number for random sequencing. Each seed value represents one sequence of random values used for test execution. Whenever you use this seed value, the same sequence of values is assigned to the Vusers in the scenario. You enable this option if you discover a problem in the test execution and want to repeat the test using the same sequence of random values.

For more information see the *HP LoadRunner Controller*, *HP Performance Center*, or *HP Business Availibility Center User Guides*.

Unique

The **Unique** method assigns a unique sequential value to the parameter for each Vuser.

In this case you must make sure there is enough data in the table for all the Vusers and their iterations. If you have 20 Vusers and you want to perform 5 iterations, your table must contain at least 100 unique values.

If there are not enough values in the data table, you can instruct VuGen how to proceed. For more details, see "Setting Properties for File Type Parameters" on page 244, or "Setting Properties for Table Type Parameters" on page 246.

Note: For LoadRunner users: If a script uses Unique file parameterization, running more than one Vuser group with that script in the same scenario may cause unexpected scenario results. For more information about Vuser groups in scenarios, see the *HP LoadRunner Controller User's Guide*.

Data Assignment and Update Methods for File/Table/ XML Parameters

For File, Table, and XML type parameters, the Data Assignment method that you select, together with your choice of Update method, affect the values that the Vusers use to substitute parameters during the scenario run.

The following table summarizes the values that Vusers use depending on which Data Assignment and Update properties you selected:

Update Method	**Data Assignment Method**		
	Sequential	**Random**	**Unique**
Each iteration	The Vuser takes the *next* value from the data table for each iteration.	The Vuser takes a *new random* value from the data table for each iteration.	The Vuser takes a value from the next unique position in the data table for each iteration.
Each occurrence (Data Files only)	The Vuser takes the *next* value from the data table for each occurrence of the parameter, even if it is within the same iteration.	The Vuser takes a *new random* value from the data table for each occurrence of the parameter, even if it is within the same iteration.	The Vuser takes a *new unique* value from the data table for each occurrence of the parameter, even if it is within the same iteration.
Once	The value assigned in the first iteration is used for all subsequent iterations for each Vuser.	The random value assigned in the first iteration is used for all iterations of that Vuser.	The unique value assigned in the first iteration is used for all subsequent iterations of the Vuser.

Examples

Assume that your table/file has the following values:

Kim; David; Michael; Jane; Ron; Alice; Ken; Julie; Fred

Sequential Method

- If you specify update on **Each iteration**, all the Vusers use Kim in the first iteration, David in the second iteration, Michael in the third iteration, and so on.
- If you specify update on **Each occurrence**, all the Vusers use Kim in the first occurrence, David in the second occurrence, Michael in the third occurrence, and so on.
- If you specify update **Once**, all Vusers take Kim for all iterations.

Note: If you select the **Sequential** method and there are not enough values in the data table, VuGen returns to the first value in the table, continuing in a loop until the end of the test.

Random Method

- If you specify update on **Each iteration**, the Vusers use random values from the table for each iteration.
- If you specify update on **Each occurrence**, the Vusers use random values for each occurrence of the parameter.
- If you specify update **Once**, all Vusers take the first randomly assigned value for all the iterations.

Unique Method

- If you specify update on **Each iteration**, for a test run of 3 iterations, the first Vuser takes Kim in the first iteration, David in the second, and Michael in the third. The second Vuser takes Jane, Ron, and Alice. The third Vuser, Ken, Julie, and Fred.
- If you specify update on **Each occurrence**, then the Vuser uses a unique value from the list for each occurrence of the parameter.

- If you specify update **Once**, the first Vuser takes Kim for all iterations, the second Vuser takes David for all iterations, and so on.

Vuser Behavior in the Controller (LoadRunner Only)

When you set up a scenario to run a parameterized script, you can instruct the Vusers how to act when there are not enough values. The following table summarizes the results of a scenario using the following parameter settings:

- Select next row = **Unique**
- Update Value on = **Each iteration**
- When out of values = **Continue with last value**

Situation	Duration	Resulting Action
More iterations than values	**Run until completion**	When the unique values are finished, each Vuser continues with the last value, but a warning message is sent to the log indicating that the values are no longer unique.
More Vusers than values	**Run indefinitely or Run for ...**	Vusers take all of the unique values until they are finished. Then the test issues an error message **Error: Insufficient records for param <param_name> in table to provide the Vuser with unique data**. To avoid this, change the **When out of values** option in the Parameter properties or the **Select next row** method in the Parameter properties.
One of two parameters are out of values	**Run indefinitely or Run for ...**	The parameter that ran out of values, continues in a cyclic manner until the values of the second parameter are no longer unique.

Setting Properties for XML Parameters

When you create a Web Service call to emulate a specific operation, the arguments in the operation may include complex structures with many values. You can use an XML type parameter to replace the entire structure with a single parameter.

You can create several value sets for the XML elements and assign a different value set for each iteration.

The XML parameter type supports complex schema types such as arrays. Choice, and <any> elements.

This section describes:

- Creating New XML Parameters
- Defining Value Sets
- Setting an Assignment Method
- Modifying XML Parameter Properties

Creating New XML Parameters

When working with Web Service **Input Arguments**, you may encounter arrays and their sub-elements. You can define a single XML parameter that will contain values for all of the array elements.

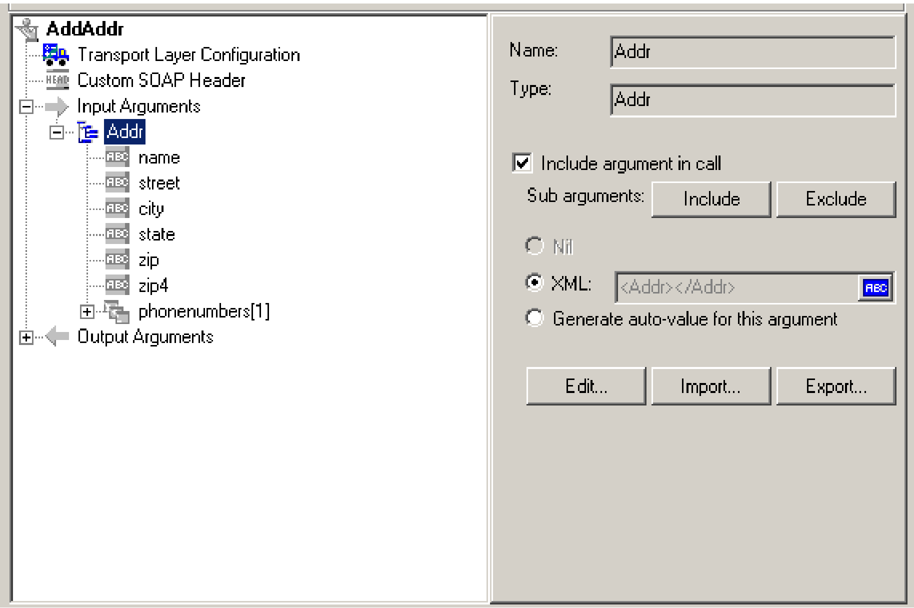
You can create new XML type parameters directly from the **Insert** menu, similar to all other parameter types. For Web Services type scripts, you create an XML parameter directly from the Web Services Call properties.

Creating XML Parameters From a Web Service Call

This section describes how to create an XML parameter from the Web Service properties.

To create an XML parameter from the Web Service call properties:

1 Select the root element of the complex data structure. The right pane displays the argument's details.



Name: Addr

Type: Addr

☑ Include argument in call

● XML: <Addr></Addr>

2 Select **XML** in the right pane, and click the **ABC** icon. The Select or Create Parameter dialog box opens.

3 In the **Parameter name** box, enter a name for the parameter.

4 In the **Parameter type** box, select **XML** if it is not already selected.

5 Click **Properties** to assign a value set now, or **OK** to close the dialog box and assign values later.

Creating XML Parameters - Standard Method

This section describes how to create an XML type parameter without viewing the properties of a Web Service call. This is the most common way of parameterizing values for most protocols and parameter types.

For Web Service Scripts, we recommend that you create parameters from within a Web Service Call, as described above.

To create a new XML parameter:

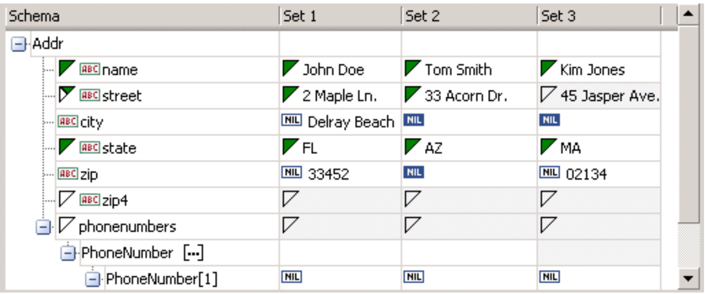
1. Select **Insert** > **New Parameter** or select a constant value in the Script view and select **Replace with a Parameter** from the right-click menu. The Select or Create Parameter dialog box opens.
2. In the **Parameter name** box, enter a name for the parameter.
3. In the **Parameter type** box, select **XML** if it is not already selected.
4. Click **Properties** to assign a value set now, or **OK** to close the dialog box and assign values later.

For information on how to set the properties, see "Setting Properties for XML Parameters" on page 253.

Defining Value Sets

This section describes how to create value sets for XML parameters.

Value sets are arrays that contain a set of values. Using the **Add Column** and **Duplicate Column** buttons, you can create multiple value sets for your parameter and use them for different iterations.



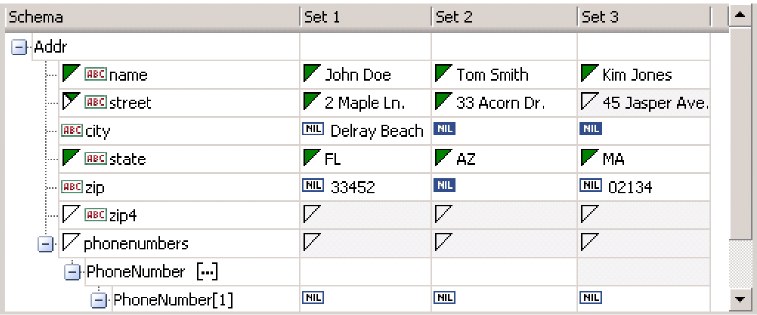
Schema	Set 1	Set 2	Set 3
⊟ Addr			
◤ ABC name	◤ John Doe	◤ Tom Smith	◤ Kim Jones
◸ ABC street	◤ 2 Maple Ln.	◤ 33 Acorn Dr.	◸ 45 Jasper Ave.
ABC city	NIL Delray Beach	NIL	NIL
◤ ABC state	◤ FL	◤ AZ	◤ MA
ABC zip	NIL 33452	NIL	NIL 02134
◸ ABC zip4	◸	◸	◸
⊟ ◸ phonenumbers	◸	◸	◸
⊟ PhoneNumber [...]			
⊟ PhoneNumber[1]	NIL	NIL	NIL

When using value sets, the number of array elements per parameter does not have to be constant.

You can use optional elements that will appear in one value set, but not in another. This allows you to vary the values you send for each of the iterations—some iterations can include specific array elements, while other iterations exclude them.

To exclude an optional element, click the small triangle in the upper left corner of the cell and insure that it is not filled in.

In the following example, **Set 1** and **Set 2** use the optional elements: **name**, **street**, and **state**. **Set 3** does not use a street name.



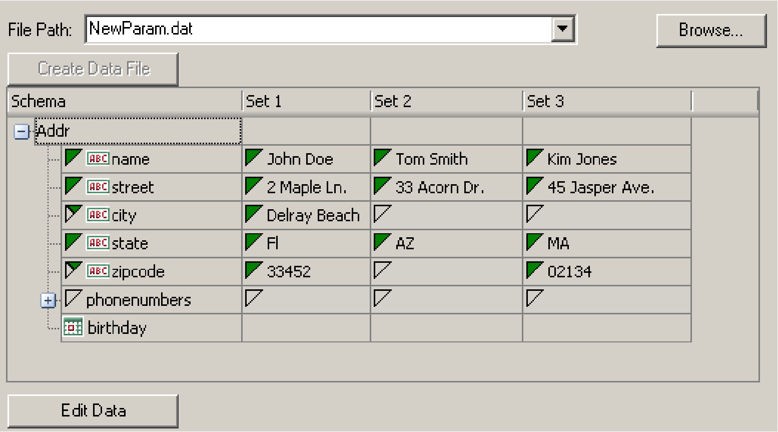
Schema	Set 1	Set 2	Set 3
Addr			
◤ ABC name	◤ John Doe	◤ Tom Smith	◤ Kim Jones
◸ ABC street	◤ 2 Maple Ln.	◤ 33 Acorn Dr.	◸ 45 Jasper Ave.
ABC city	NIL Delray Beach	NIL	NIL
◤ ABC state	◤ FL	◤ AZ	◤ MA
ABC zip	NIL 33452	NIL	NIL 02134
◸ ABC zip4	◸	◸	◸
◸ phonenumbers	◸	◸	◸
PhoneNumber [...]			
PhoneNumber[1]	NIL	NIL	NIL

For more information about editing the values, see *Volume II-Protocols*.

To set parameter element values:

1 View the Parameter Properties.

If the Parameter Properties dialog box is not open, select **Vuser > Parameter List** and select the desired parameter. The dialog box shows a read-only view of the parameter values.

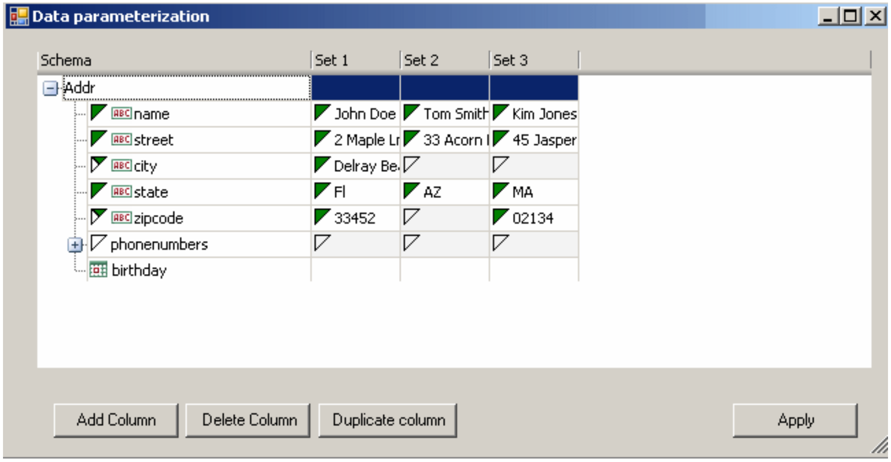


File Path: NewParam.dat

Schema	Set 1	Set 2	Set 3
Addr			
name	John Doe	Tom Smith	Kim Jones
street	2 Maple Ln.	33 Acorn Dr.	45 Jasper Ave.
city	Delray Beach		
state	Fl	AZ	MA
zipcode	33452		02134
phonenumbers			
birthday			

2 Open the Data Parameterization box.

Click the **Edit Data** button to open the Data Parameterization dialog box.



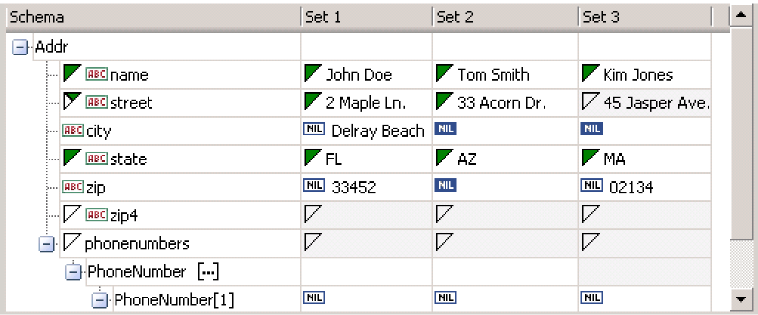
Data parameterization

Schema	Set 1	Set 2	Set 3
Addr			
name	John Doe	Tom Smitł	Kim Jones
street	2 Maple Lr	33 Acorn I	45 Jasper
city	Delray Be.		
state	Fl	AZ	MA
zipcode	33452		02134
phonenumbers			
birthday			

3 Define value sets for the XML parameter.

In the **Set** columns, insert values corresponding to the schema.

If a row says **NIL**, it implies that the element is nillable. To include a value for the nillable element, enter the value as usual. To mark a value as **nil**, click the NIL icon to fill it in. This erases any value that you may have assigned to the element. In the following example, the **city** element is nillable, but it is only marked as nil in **Set 2** and **Set 3**—not in **Set 1**.



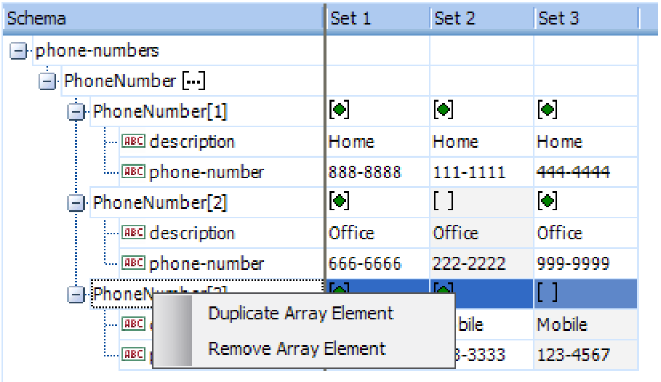
Schema	Set 1	Set 2	Set 3
⊟ Addr			
◤ ABC name	◤ John Doe	◤ Tom Smith	◤ Kim Jones
◸ ABC street	◤ 2 Maple Ln.	◤ 33 Acorn Dr.	◸ 45 Jasper Ave.
ABC city	NIL Delray Beach	NIL	NIL
◤ ABC state	◤ FL	◤ AZ	◤ MA
ABC zip	NIL 33452	NIL	NIL 02134
◸ ABC zip4	◸	◸	◸
⊟ ◸ phonenumbers	◸	◸	◸
⊟ PhoneNumber [--]			
⊟ PhoneNumber[1]	NIL	NIL	NIL

4 Create additional value sets.

To insert more value sets, click **Add Column** and insert another set of values in the new column. To copy an existing value set, select a row in the value set you want to copy and click **Duplicate Column**.

5 Copy arrays.

To duplicate an array element and its children, select the parent node and choose **Duplicate Array Element** from the right-click menu.



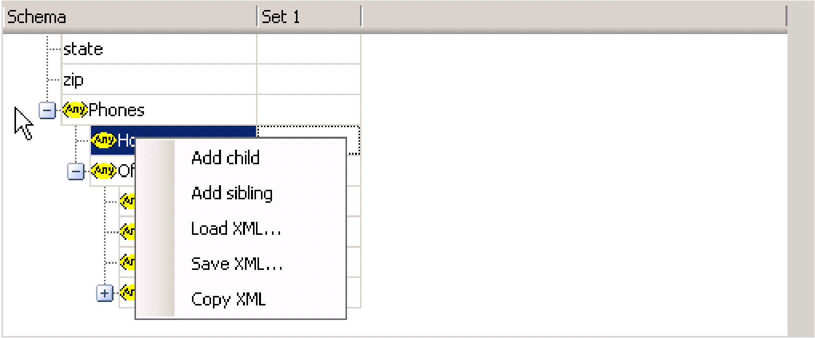
Schema	Set 1	Set 2	Set 3
⊟ phone-numbers			
⊟ PhoneNumber [--]			
⊟ PhoneNumber[1]	[◆]	[◆]	[◆]
ABC description	Home	Home	Home
ABC phone-number	888-8888	111-1111	~~444-4444~~
⊟ PhoneNumber[2]	[◆]	[ ]	[◆]
ABC description	Office	Office	Office
ABC phone-number	666-6666	222-2222	999-9999
⊟ PhoneN			[ ]
ABC		bile	Mobile
ABC		3-3333	123-4567

6 Handle the <any> elements.

For **any** type elements, right-click <**any**> in the **Schema** column and select one of the available options. These options may vary depending on the location of the cursor.

- **Add Array Element.** Adds a sub-element under the root element.
- **Insert child.** Adds a sub-element to the selected element.
- **Insert sibling.** Adds a sub-element on the same level as the selected element.
- **Load XML.** Loads the element values from an XML file.
- **Save XML.** Saves the array as an XML file.
- **Copy XML**. Copies the full XML of the selected element to the clipboard.

Click the **Rename** text to provide a meaningful name for each array element.



7 Remove unwanted columns.

To remove a value set, select it and click **Delete Column**.

8 Save the changes.

Click **Apply** to save the changes and update the view in the Parameter Properties dialog box.

Setting an Assignment Method

The assignment method indicates which of the value sets to use and how to use them. For example, you can instruct Vusers to use a new value set for each iteration and use the value sets sequentially or at random. For more information, see "Data Assignment and Update Methods for File/Table/ XML Parameters" on page 250.

To define an assignment method:

1 Open the Parameter Properties and select a parameter.

2 Define a data assignment method.

In the **Select next value** list, select a data assignment method to instruct the Vuser how to select the file data during Vuser script execution. The options are: **Sequential**, **Random**, or **Unique**. For more information, see "Choosing Data Assignment Methods for File/Table Parameters" on page 248.

3 Select an update option for the parameter.

In the **Update value on** list, select an update option. The choices are **Each Iteration**, **Each Occurrence**, and **Once**. For more information, see "Data Assignment and Update Methods for File/Table/ XML Parameters" on page 250.

4. If you chose **Unique** as the data assignment method the **When out of values** and **Allocate Vuser values in the Controller** options become enabled.

   - **When out of values.** Specify what to do when there is no more unique data: **Abort Vuser**, **Continue in a cyclic manne**_r_, or **Continue with last value**.

   - **Allocate Vuser values in the Controller (**for LoadRunner users only**).** Indicate whether you want to manually allocate data blocks for the Vusers. You can allow the Controller to automatically allocate a block size or you can specify the desired number of values. Select **Automatically allocate block size** or **Allocate** x **values for each Vuser.** For the second option, specify the number of values to allocate.

     To track this occurrence, enable the **Extended Log** > **Parameter Substitution** option in the Log Run-Time settings. When there is not enough data, VuGen writes a warning message to the Vuser log: **No more unique values for this parameter in table <table_name>**.

5. In the Parameter Properties dialog box, click **Close**.

   The list of input arguments is replaced by the parameter name, and ABC button is replace by a table icon which you can click to edit the parameter properties or un-parameterize the parameter.

Modifying XML Parameter Properties

If you need modify a value set of a parameter, you can do so from the Web Service's Step Properties tab.

To modify XML parameter properties:

1. In the Web Service script's tree view, click the **Step Properties** tab.

2. Under **Input Arguments**, select the XML parameter. The right pane displays the parameter details.

3. To modify the XML parameter properties, click the table icon button adjacent to the **XML** box and select **Parameter Properties**.

4. Modify the parameter properties as described in "Defining Value Sets" on page 255.

15

Setting Parameter Properties

A parameter is defined according to the type of information it replaces.

This chapter includes:

- About Setting Parameter Properties on page 263
- Setting Properties for Internal Data Parameter Types on page 264
- Setting Properties for User-Defined Functions on page 274
- Customizing Parameter Formats on page 275
- Selecting an Update Method on page 276
- Simulating File Type Parameters on page 277
- Using the File Parameter Simulator on page 279

About Setting Parameter Properties

When you define a parameter's properties, you specify the source for the parameter data.

You define properties for any one of the following data source types:

Data type	Data Source
Internal Data Parameter Types	Data that is generated internally by the Vuser: Date/Time, Group Name, Iteration Number, Load Generator Name, Random Number, Unique Number, and Vuser ID.

User-Defined Functions	Data that is generated using a function from an external DLL.
Files and Tables	Data that is contained in a file—either an existing file or one that you create with VuGen or MS Query.

This chapter describes how to assign properties to Internal Data and User-Defined Function parameters.

For information on defining properties for Table or File type parameters, see Chapter 14, "File, Table, and XML Parameter Types."

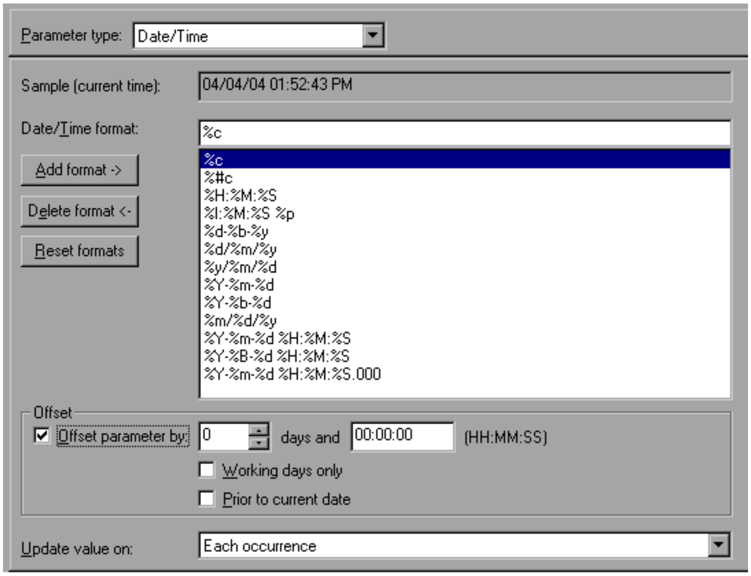
Setting Properties for Internal Data Parameter Types

This section discusses setting the properties for data that is generated internally by the Vuser. Internal data includes data such as:

- Date/Time
- Group Name
- Iteration Number
- Load Generator Name
- Random Number
- Unique Number
- Vuser ID

Date/Time

Date/Time replaces the parameter with the current date and/or time. To specify a date/time format, you can select a format from the format list or specify your own format. The format should correspond to the date/time format recorded in your script.



Parameter type: Date/Time

Sample (current time): 04/04/04 01:52:43 PM

Date/Time format: %c

```
%c
%#c
%H:%M:%S
%I:%M:%S %p
%d-%b-%y
%d/%m/%y
%y/%m/%d
%Y-%m-%d
%Y-%b-%d
%m/%d/%y
%Y-%m-%d %H:%M:%S
%Y-%B-%d %H:%M:%S
%Y-%m-%d %H:%M:%S.000
```

Offset

☑ Offset parameter by: 0 days and 00:00:00 (HH:MM:SS)

□ Working days only

□ Prior to current date

Update value on: Each occurrence

VuGen lets you set an offset for the date/time parameter. For example, if you want to test a date next month, you set the date offset to 30 days. If you want to test your application for a future time, you specify a time offset. You can specify a forward, future offset (default) or a backward offset, a date or time that already passed. In addition, you can instruct VuGen to use date values for work days only, excluding Saturdays and Sundays.

The following table describes the date/time symbols:

Symbol	Description
c	complete date and time in digits
#c	complete date as a string and time

Symbol	Description
H	hours (24 hour clock)
I	hours (12 hour clock)
M	minutes
S	seconds
p	AM or PM
d	day
m	month in digits (01-12)
b	month as a string - short format (e.g. Dec)
B	month as a string - long format (e.g. December)
y	year in short format (e.g. 03)
Y	year in long format (e.g. 2003)

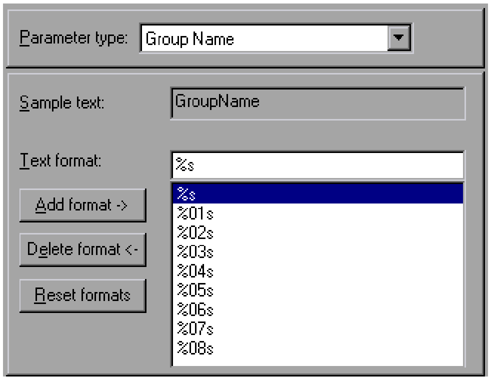
To set the properties for Date/Time parameters:

1. Select one of the existing date/time formats or create a new format. You can view a sample of how VuGen will display the value, in the **Sample (Current time)** box. For information on customizing parameter formats, see "Customizing Parameter Formats" on page 275.
2. To set the date and time offsets, select **Offset Parameter by** and specify the desired offset for the date and time values.

   To instruct VuGen to use working day dates only, excluding weekends, select **Working days only**. To indicate a negative offset to test a date prior to the current, select **Prior to current date**.
3. Select an update method, instructing the Vuser when to update parameter values—**Each occurrence**, **Each iteration**, or **Once**. For more information, see "Selecting an Update Method" on page 276.
4. Click **Close** to accept the settings and close the Parameter Properties dialog box.

Group Name

Group Name replaces the parameter with the name of the Vuser Group. You specify the name of the Vuser Group when you create a scenario. When you run a script from VuGen, the Group name is always *None*.



Parameter type: Group Name

Sample text: GroupName

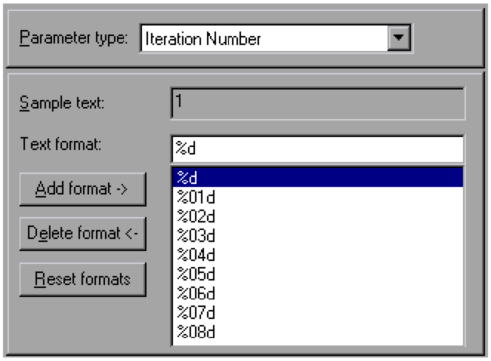
Text format: %s

To set properties for the Group Name parameter type:

1. Select one of the available formats or create a new one. You select a format to specify the length of the parameter string. For details, see "Customizing Parameter Formats" on page 275.
2. Click **Close** to accept the settings and close the Parameter Properties dialog box.

Iteration Number

Iteration Number replaces the parameter with the current iteration number.



Parameter type: Iteration Number

Sample text: 1

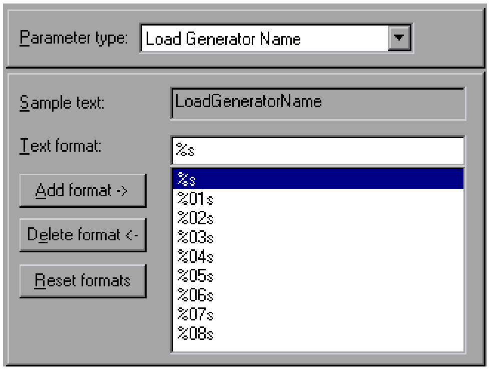
Text format: %d

To set the properties for the Iteration Number parameter type:

1. Select one of the available formats or create a new one. You select a format to specify the length of the parameter string. For details, see "Customizing Parameter Formats" on page 275.
2. Click **Close** to save the settings and close the Parameter Properties dialog box.

Load Generator Name

Load Generator Name replaces the parameter with the name of the Vuser script's load generator. The load generator is the computer on which the Vuser is running.



Parameter type: Load Generator Name

Sample text: LoadGeneratorName

Text format: %s

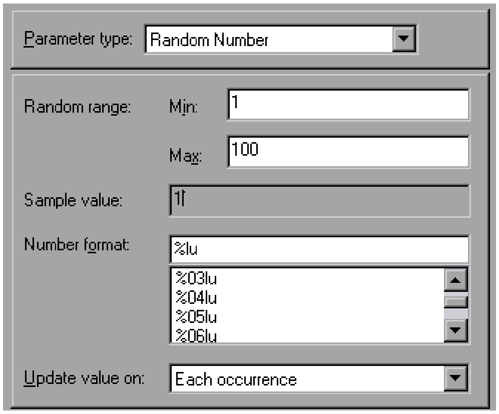
To set the properties for the Load Generator Name parameter type:

1. Select one of the available formats or create a new one. You select a format to specify the length of the parameter string. For details, see "Customizing Parameter Formats" on page 275.
2. Click **Close** to save the settings and close the Parameter Properties dialog box.

Random Number

Random Number replaces the parameter with a random number. You set a range of numbers by specifying minimum and maximum values.

You can use the Random Number parameter type to sample your system's behavior within a possible range of values. For example, to run a query for 50 employees, where employee ID numbers range from 1 through 1000, create 50 Vusers and set the minimum to 1 and maximum to 1000. Each Vuser receives a random number, from within the range of 1 to 1000.



Parameter type: Random Number

Random range: Min: 1

Max: 100

Sample value: 1

Number format: %lu

%03lu
%04lu
%05lu
%06lu

Update value on: Each occurrence

To set the properties for the Random Number parameter type:

1 Enter a range defining the set of possible parameter values. You specify minimum and maximum values for the range of random numbers.

2 Select a **Number format**, indicating the length of the random number. Specify **%01lu** (or **%lu**) for one digit, **%02lu** for two digits, and so on. You can view a sample of how VuGen will display the value, in the **Sample value** box.

3 Select an update method, instructing the Vuser when to update parameter values—**Each occurrence**, **Each iteration**, or **Once**. For more information, see "Selecting an Update Method" on page 276.

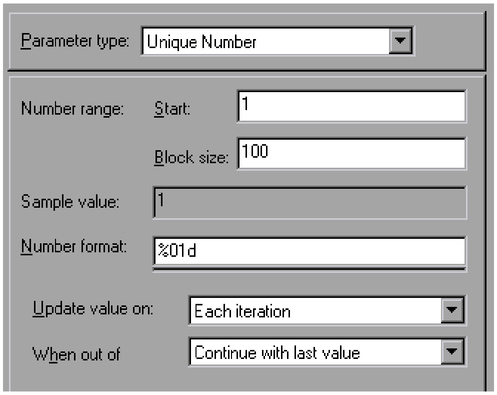
4 Click **Close** to accept the settings and close the Parameter Properties dialog box.

Unique Number

Unique Number replaces the parameter with a unique number.

When you create a Unique Number type parameter, you specify a start number and a block size. The block size indicates the size of the block of numbers assigned to each Vuser. Each Vuser begins at the bottom of its range and increments the parameter value for each iteration. For example, if you set the Start number at 1 with a block of 500, the first Vuser uses the value 1 and the next Vuser uses the value 501, in their first iterations.

The number of digits in the unique number string together with the block size determine the number of iterations and Vusers. For example, if you are limited to five digits using a block size of 500, only 100,000 numbers (0-99,999) are available. It is therefore possible to run only 200 Vusers, with each Vuser running 500 iterations.



Parameter type: Unique Number

Number range: Start: 1

Block size: 100

Sample value: 1

Number format: %01d

Update value on: Each iteration

When out of: Continue with last value

You can also indicate what action to take when there are no more unique numbers in the block: **Abort Vuser**, **Continue in a cyclic manner**, or **Continue with last value** (default).

You can use the Unique Number parameter type to check your system's behavior for all possible values of the parameter. For example, to perform a query for all employees, whose ID numbers range from 100 through 199, create 100 Vusers and set the start number to 100 and block size to 100. Each Vuser receives a unique number, beginning with 100 and ending with 199.

Note: VuGen creates only one instance of Unique Number type parameters. If you define multiple parameters and assign them the Unique Number Parameter type, the values will not overlap. For example, if you define two parameters with blocks of 100 for 5 iterations, the Vusers in the first group will use 1, 101, 201, 301, and 401. The Vusers in the next group, using the second parameter will use 501, 601, 701, 801, and 901.

After you define a Unique parameter, you can use it in other scripts by pointing to the same parameter file. The parameter retains all of the properties that you assigned to the original parameter.

To set the properties for the Unique Number parameter type:

1. Enter a start number and the desired block size. For example, if you want 500 numbers beginning with 1, specify 1 in the **Start** box, and 500 in the **Block size per Vuser** box.
2. Select a Number format, indicating the length of the unique number. Specify **%01d** (or **%d**) for one digit, **%02d** for two digits, and so on. You can view a sample of how VuGen will display the value, in the **Sample value** box.
3. Select an update method, instructing the Vuser when to update parameter values—**Each occurrence**, **Each iteration**, or **Once**. For more information, see "Selecting an Update Method" on page 276.
4. Indicate what to do when there are no more unique values, in the **When out of values** box: **Abort Vuser**, **Continue in cyclic manner**, or **Continue with last value**.

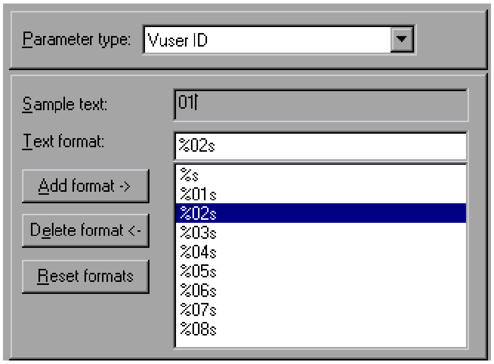
    Note: (For LoadRunner users!)
    When scheduling a scenario in the Controller, the **When out of values** option only applies to the **Run for HH:MM:SS** option in the Schedule Builder's Duration tab. It is ignored for the **Run until completion** option.

5 Click **Close** to accept the settings and close the Parameter Properties dialog box.

Vuser ID

Note: This parameter type applies primarily to LoadRunner users.

Vuser ID replaces the parameter with the ID number assigned to the Vuser by the Controller during a scenario run. When you run a script from VuGen, the Vuser ID is always -1.



Parameter type: Vuser ID

Sample text: 01

Text format: %02s

Note: This is not the ID number that appears in the Vuser window—it is a unique ID number generated at runtime.

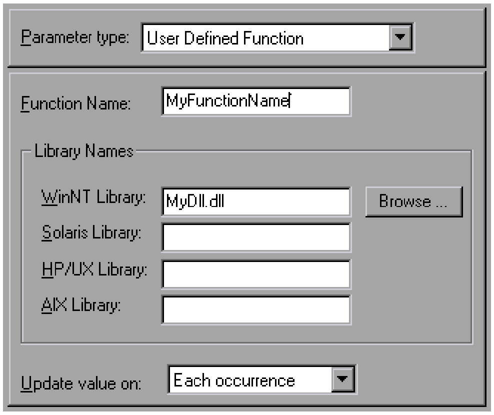
To set the properties for the Vuser ID parameter type:

1 Select one of the available formats or create a new one. You select a format to specify the length and structure of the parameter string. For details, see "Customizing Parameter Formats" on page 275.

2 Click **Close** to accept the settings and close the Parameter Properties dialog box.

Setting Properties for User-Defined Functions

In the Parameter Properties dialog box, select **User Defined Function** from the **Parameter type** list.



Parameter type: User Defined Function

Function Name: MyFunctionName

WinNT Library: MyDll.dll

Update value on: Each occurrence

To set the properties for user-defined functions:

1. Specify the function name in the **Function Name** box. Use the name of the function as it appears in the DLL file.
2. In the **Library Names** section, specify a library in the relevant **Library** box. If necessary, locate the file using the **Browse** command.
3. Select an update method for the values. For more information on update methods for user-defined functions, see "Selecting an Update Method" on page 276.

Customizing Parameter Formats

For most data types, you can customize a format for a parameter by selecting an existing format or specifying a new one.

Note: The parameter format should match the recorded values. If the format of the parameter differs from the format of the original recorded value, the script may not run correctly.

The format specifies the length and structure of the resulting parameter string. The resulting parameter string is the actual parameter value together with any text that accompanies the parameter. For example, if you specify a format of "%05s," a Vuser ID of 5 is displayed as "00005," padding the single digit with four zeros. To pad the number with blank spaces, specify the number of spaces without a "0." For example, %4s adds blank spaces before the Vuser ID so that the resulting parameter string is 4 characters long.

You can specify a text string before and after the actual parameter value.

For example, if you specify a format of "Vuser No: %03s," then a Vuser ID of 1 is displayed as "**Vuser No: 001**."

You can add and delete formats for the following parameter types: Date/Time; Group Name; Iteration Number; Load Generator Name; Vuser ID.

To add a format to a parameter type:

1 In the Parameter Properties dialog box, select the parameter type that you want to format.

2 Enter the format symbols in the editable box and click **Add Format**.

Note: When you add a format to the list, VuGen saves it with the Vuser, making it available for future use.

To delete a format:

In the Parameter Properties dialog box, select an existing format from the list, and click **Delete format**.

To restore the original formats:

Click **Reset formats**.

Selecting an Update Method

When using several of the parameter types, VuGen lets you specify how to update the values for the parameters. To set an Update method, select a method from the **Update value on** list. The available update methods are:

- Each Occurrence
- Each Iteration
- Once

Each Occurrence

The **Each occurrence** method instructs the Vuser to use a new value for each occurrence of the parameter. This is useful when the statements using a parameter are unrelated. For example, for random data, it may be useful to use a new value for each occurrence of the parameter.

Each Iteration

The **Each iteration** method instructs the Vuser to use a new value for each script iteration. If a parameter appears in a script several times, the Vuser uses the same value for all occurrences of the parameter, for the entire iteration. This is useful when the statements using a parameter are related.

Note: If you create an action block with parameters using its own iteration count—if you instruct VuGen to update their values each iteration, it refers to the global iteration and not the block iteration. For more information about action blocks, see "Creating Action Blocks" on page 417.

Once

The **Once** method instructs the Vuser to update the parameter value only once during the scenario run. The Vuser uses the same parameter value for all occurrences and all iterations of the parameter. This type may be useful when working with dates and times.

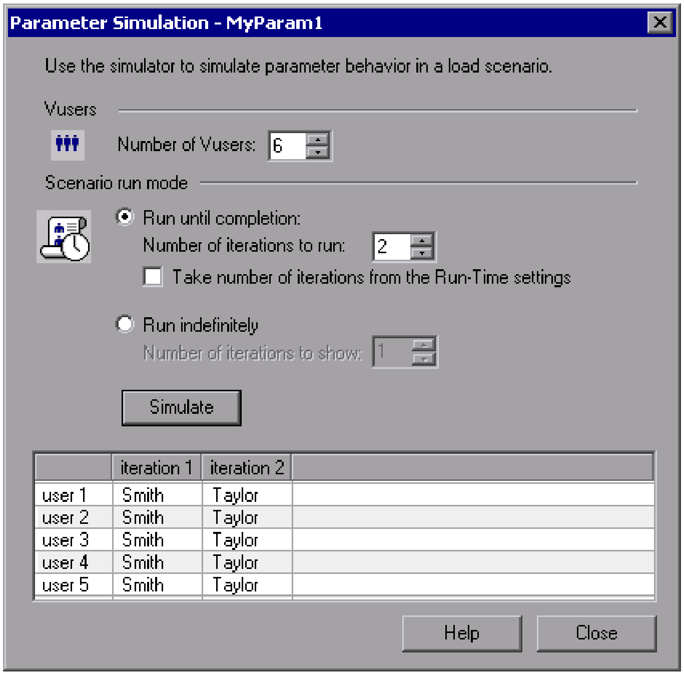
Simulating File Type Parameters

After you have created a File type parameter, you can use the File Parameter Simulator to simulate the parameter substitution in an actual scenario. This allows you to correct any wrong parameters before you run the script in the Controller. The File Parameter Simulator is only relevant for LoadRunner users.

Note: Not all types of Parameter Substitution can be simulated. If you select **Select next row: Same line as...** or **Update value on: Each occurrence**, then the File Parameter Simulator will not open.

To run a File Parameter Simulation:

1 From the Parameter List dialog box, click **Simulate Parameter**. The Parameter Simulation dialog box opens.



Parameter Simulation - MyParam1

Use the simulator to simulate parameter behavior in a load scenario.

Vusers

Number of Vusers: 6

Scenario run mode

● Run until completion:

Number of iterations to run: 2

□ Take number of iterations from the Run-Time settings

○ Run indefinitely

Number of iterations to show: 1

	iteration 1	iteration 2
user 1	Smith	Taylor
user 2	Smith	Taylor
user 3	Smith	Taylor
user 4	Smith	Taylor
user 5	Smith	Taylor

2 Select the number of Vusers to run in the simulation from the **Number of Vusers** box.

3 Select Scenario run mode:

- ➤ If you select **Run until completion**, select the number of iterations to run from the **Number of iterations to run** box, or take the number from the Run-Time settings.
- ➤ If you select **Run indefinitely**, you must define how many iterations to show. The number of iteration in the Run-Time settings is ignored. The number you select does not change the value distribution for each Vuser; it is only for viewing purposes.

Note:

- **Run Indefinitely** is compliant with the **Real-life schedule** in the Scheduler of the Controller.
- If you select **Select next row: Unique** in the Parameter List dialog, then each Vuser is assigned a unique range of rows from which the Simulator will substitute values (for that Vuser).

  With this setting, the default selection in the Allocate Vuser values in the Controller section is **Automatically allocate block size**. In this case, when you run the simulation, the range allocation takes place in accordance with your Scenario run mode selection.

  If you change the default selection to **Allocate x values for each Vuser,** then the Vusers will be allocated the amount of values you specify, ignoring of your Scenario run mode selection.

4 Click **Simulate**.

Note: The File Parameter Simulator can simulate up to 256 iterations and 256 Vusers.

Using the File Parameter Simulator

The following section illustrates how the File Parameter Simulator operates, demonstrating the difference between the Scenario run mode settings.

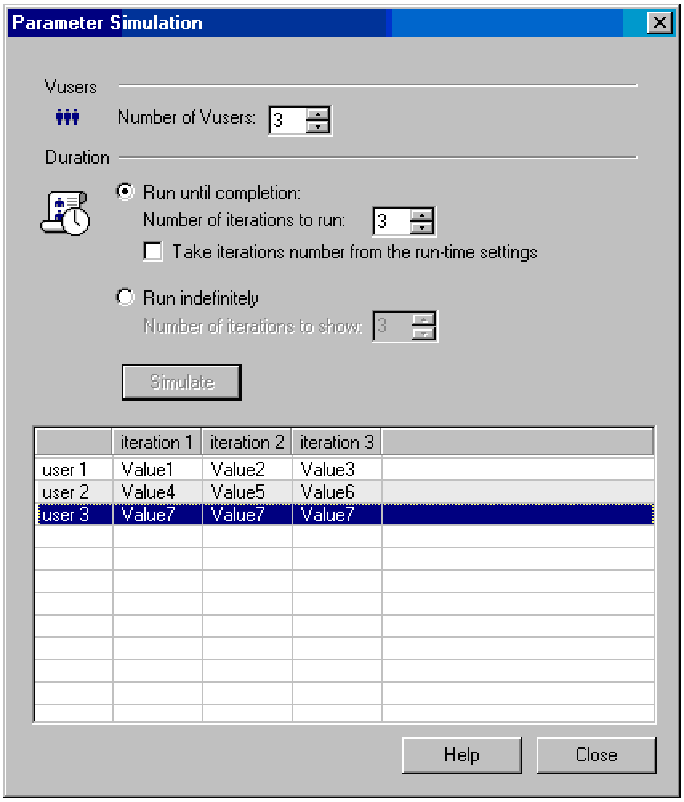
In the following examples, the settings in the Parameter List dialog box are:

- **Values for the new parameter**. Value1 to Value7
- **Select next row**. Unique

- **When out of rows**. Continue with last value
- **Allocate Vuser values in the Controller**. Automatically allocate block size

Scenario run mode: Run until completion

In the following example, the user has selected three Vusers, set the Scenario run mode to **Run until completion**, and selected three iterations.



Parameter Simulation

Vusers

Number of Vusers: 3

Duration

● Run until completion:

Number of iterations to run: 3

□ Take iterations number from the run-time settings

○ Run indefinitely

Number of iterations to show: 3

	iteration 1	iteration 2	iteration 3
user 1	Value1	Value2	Value3
user 2	Value4	Value5	Value6
user 3	Value7	Value7	Value7

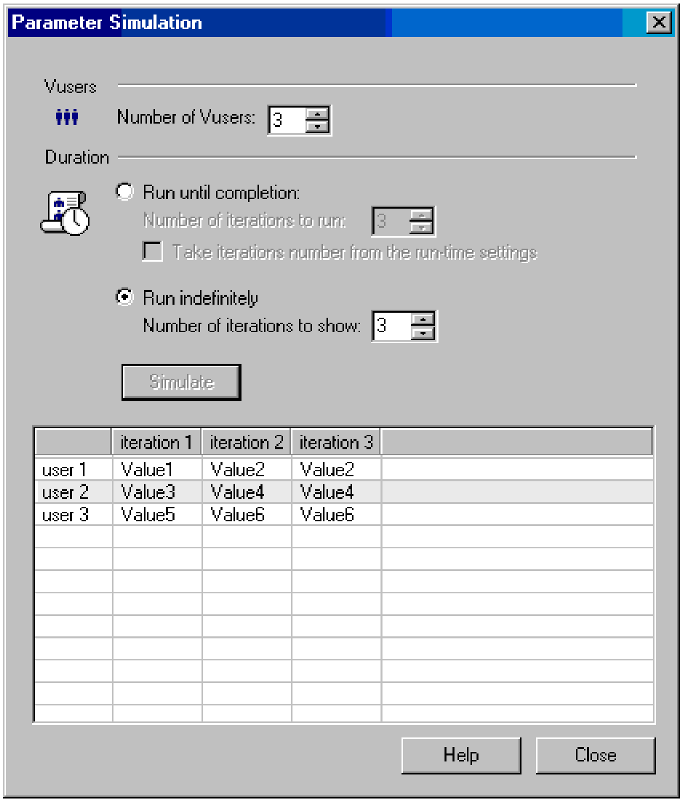
When the scenario run mode is set to **Run until completion**, the number of rows that each Vuser receives is the same as the number of iterations. The range allocation stops when there are no longer enough rows in the table.

As the simulation is run, the first Vuser takes the first three values (because this was the number of iterations). The second Vuser takes the next three values. The third Vuser takes the remaining value in the first iteration. For the remaining iterations, since the **When out of values** option in the Parameter List dialog box was set to **Continue with last value,** the third Vuser continues with the same value.

A fourth Vuser would have failed.

Scenario run mode: Run indefinitely

In the following example, the user has selected 3 Vusers and set the Scenario run mode to Run indefinitely and selected to show 3 iterations.



Parameter Simulation

Vusers

Number of Vusers: 3

Duration

○ Run until completion:

Number of iterations to run: 3

□ Take iterations number from the run-time settings

● Run indefinitely

Number of iterations to show: 3

	iteration 1	iteration 2	iteration 3
user 1	Value1	Value2	Value2
user 2	Value3	Value4	Value4
user 3	Value5	Value6	Value6

When the Scenario run mode is set to Run indefinitely, the allocated range for each Vuser is calculated by dividing the number of cells in the .dat file by the number of Vusers. In this scenario, that is 7/3 = 2 (The simulator takes the closest smaller integer.).

As the simulation is run, the first Vuser takes Value1 and Value2. The second Vuser takes Value3 and Value4 and the third Vuser takes Value5 and Value6. Since there are were only 3 Vusers, Value7 was not distributed.

Note: If you hold the mouse over the cells in the first column of the table, a tool tip appears with information about which values were assigned to that Vuser.

If you hold the mouse over cells which were not assigned values, a tool tip appears with the reason no values were assigned.

A tool tip does not appear if a proper value was assigned.

Part 3

Recording Options

16

Recording Options for Selected Protocols

This chapter contains information about recording options for selected protocols.

This chapter includes:

- EJB Recording Options on page 286
- RDP Recording Options on page 287
- Understanding Citrix Recording Options on page 292
- Setting the Citrix Recording Options on page 299
- Database Recording Options on page 300
- Database Advanced Recording Options on page 301
- Setting the AMF Recording Mode on page 304
- AMF Code Generation Options on page 308
- Flex Code Generation Options on page 309
- Microsoft .NET Recording Options on page 311
- WCF Recording Options on page 318
- RTE Configuration Options on page 320
- RTE Recording Options on page 321
- SAPGUI Recording Options on page 325
- SAP-Web Recording Options on page 329
- Setting the Correlation Recording Options on page 330
- Web, Wireless, and Oracle NCA Recording Options on page 332

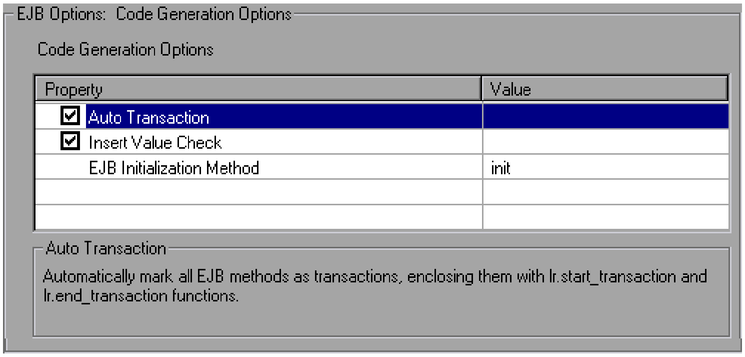
EJB Recording Options

You can set Classpath and Code Generation recording options for EJB Vusers. This section contains information about Code Generation recording options. For information about Classpath options, see Chapter 18, "Java Recording Options."

The **EJB Code Generation** options allow you to set properties in the area of automatic transactions and value checks. You can also indicate where to store the initialization method.

To set the EJB Code Generation recording options:

1 Click **Options** in the Start Recording dialog box. Select the **EJB Options:Code Generation Options** node in the Recording Options tree to edit the code generation options.



EJB Options: Code Generation Options

Code Generation Options

Property	Value
☑ Auto Transaction	
☑ Insert Value Check	
EJB Initialization Method	init

Auto Transaction

Automatically mark all EJB methods as transactions, enclosing them with lr.start_transaction and lr.end_transaction functions.

2 Enable the **Auto Transaction** option to automatically mark all EJB methods as transactions. This encloses all methods with **lr.start_transaction** and **lr.end_transaction** functions. By default, this option is enabled (true).

3 Enable the **Insert Value Check** option to automatically insert an **lr.value_check** function after each EJB method. This function checks for the expected return value for primitive values and strings.

4 Select an **EJB Initialization Method**. This is the method to which the EJB/JNDI initialization properties are written. The available methods are **init** (default) and **action**.

RDP Recording Options

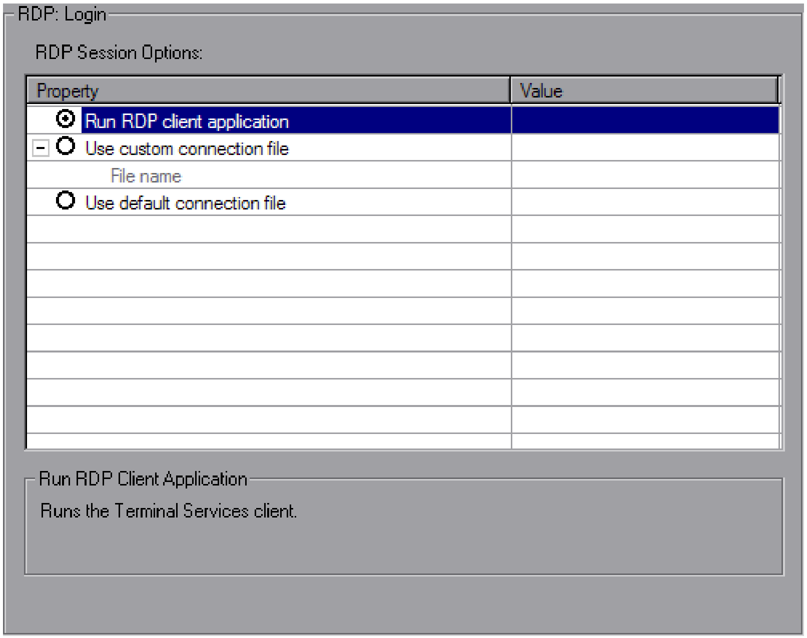
Before recording the script, you can configure how the RDP activities are recorded and how the script is generated. You do this by setting the Recording Options.

You can set the recording options in the following areas:

- "RDP Login Options" on page 287
- "RDP Basic Code Generation Options" on page 288
- "RDP Advanced Code Generation Options" on page 290
- "RDP Agent Code Generation Options" on page 291

RDP Login Options

In the RDP Login Options, you indicate how to run the Terminal Services client session.



RDP: Login

RDP Session Options:

Property	Value
● Run RDP client application	
⊟ ○ Use custom connection file	
File name	
○ Use default connection file	

Run RDP Client Application

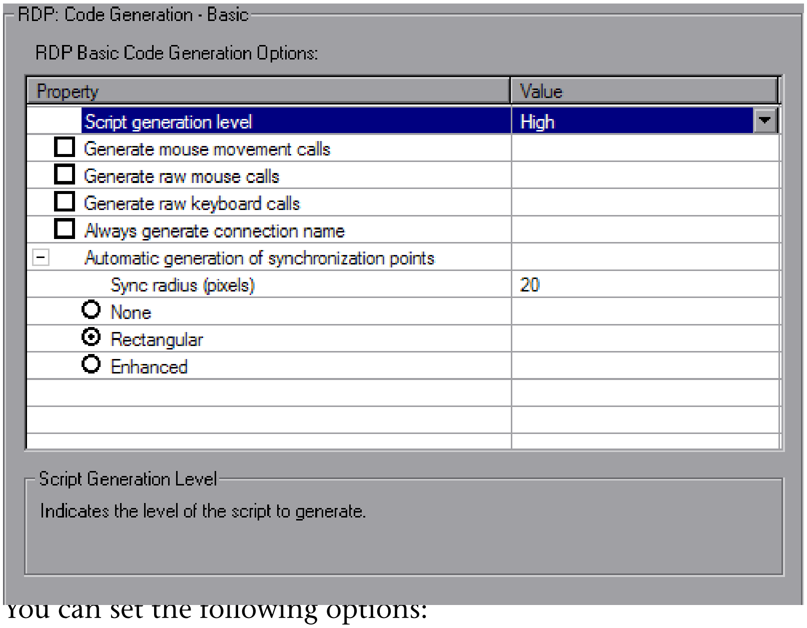
Runs the Terminal Services client.

- **Run RDP client application.** Run the Terminal Services client.

- **Use custom connection file.** Run the Terminal Services client using an existing connection file. The file should have an ***.rdp** extension. You can browse for the file on your file system or network.
- **Use default connection file**. Use the **Default.rdp** file in your document's directory to connect.

RDP Basic Code Generation Options

The Basic Code Generation Options control the way VuGen creates a script—the level of detail, triggers, and timeouts.



RDP: Code Generation - Basic

RDP Basic Code Generation Options:

Property	Value
Script generation level	High
☐ Generate mouse movement calls	
☐ Generate raw mouse calls	
☐ Generate raw keyboard calls	
☐ Always generate connection name	
− Automatic generation of synchronization points	
Sync radius (pixels)	20
○ None	
● Rectangular	
○ Enhanced	

Script Generation Level

Indicates the level of the script to generate.

You can set the following options:

- **Script generation level.** The level of the script and the type of API functions to use when generating the script. The available levels are **High**, **Low** or **Raw**.
  - **High.** Generate high level scripts. Keyboard events are translated to **rdp_type** calls. Two consecutive mouse clicks with the same coordinates are translated as a double-click.

  - **Low.** Generate low level scripts. Key up/down events are translated into rdp_key events. Modifier keys (Alt, Ctrl, Shift) are used as a KeyModifier parameter for other functions. Mouse up/down/ move events are translated to mouse click/drag events.

  - **Raw.** Generates a script on a raw level, by extracting input events from network buffers and generating calls in their simplest form: key up/down, mouse up/down/move. The KeyModifier parameter is not used.

- **Generate mouse movement calls**. Generates **rdp_mouse_move** calls in the script. When enabled, this option significantly increases the script size (disabled by default).

- **Generate raw mouse calls**. If enabled, VuGen generates **rdp_mouse_button_up/down** calls as if the script level was set to Raw. Keyboard calls will still be generated according to the script level. If disabled, VuGen generates Mouse calls according to the script level. If the script level is set to Raw, this option is ignored (disabled by default).

- **Generate raw keyboard calls.** If enabled, VuGen generates **rdp_raw_key_up/down** calls as if the script level was set to Raw. Mouse calls will still be generated according to the script level. If disabled, VuGen generates Keyboard calls according to the script level. If the script level is set to Raw, this option is ignored (disabled by default).

- **Always generate connection name.** If selected, function call will contain the ConnectionName parameter. If not selected, the functions will only contain this parameter if more than a single **rdp_connect_server** appears in the script (disabled by default).

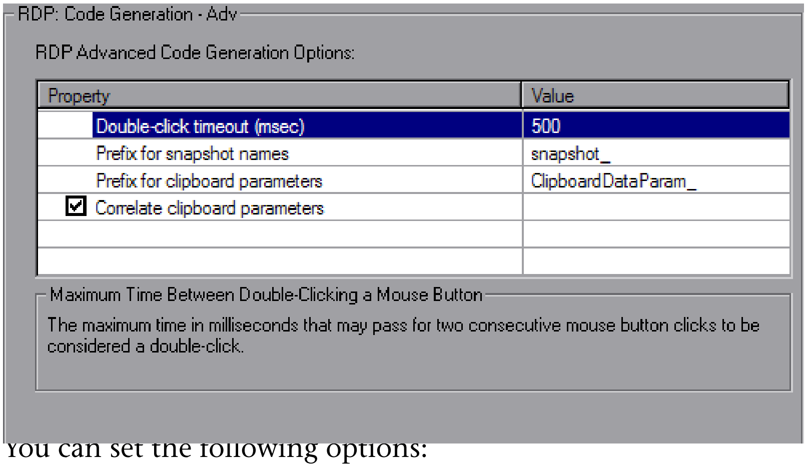
- **Automatic generation of syncronization points.** Synchronization points allow the script to pause in the replay while waiting for a window or dialog to pop-up or some other control to fulfil a certain condition. This option automatically generates **sync_on_image** functions before mouse clicks and drags (enabled by default). The **Sync radius** is the distance from the mouse operation to the sides of the rectangle which defines the synchronization area. The default is 20 pixels.

  - **None.** No synchronization points are automatically added.

- **Rectangular**. Creates synchronization points as rectangular boxes centered around the click or drag location. The **Sync Radius** is the distance from the mouse operation to the sides of the rectangle which defines the synchronization area. The default is 20 pixels.
- **Enhanced**. Creates synchronization points designed to select only the desired location (e.g. a button) and to react to changes in the UI (e.g. the button moves). If a synchronization region is not recognized, the rectangular synchronization settings are used.

RDP Advanced Code Generation Options

The Advanced Code Generation Options control the way VuGen creates a script - default parameter name prefixes and other options. Only advanced users are advised to modify these settings.



RDP: Code Generation - Adv

RDP Advanced Code Generation Options:

Property	Value
Double-click timeout (msec)	500
Prefix for snapshot names	snapshot_
Prefix for clipboard parameters	ClipboardDataParam_
☑ Correlate clipboard parameters	

Maximum Time Between Double-Clicking a Mouse Button

The maximum time in milliseconds that may pass for two consecutive mouse button clicks to be considered a double-click.

You can set the following options:

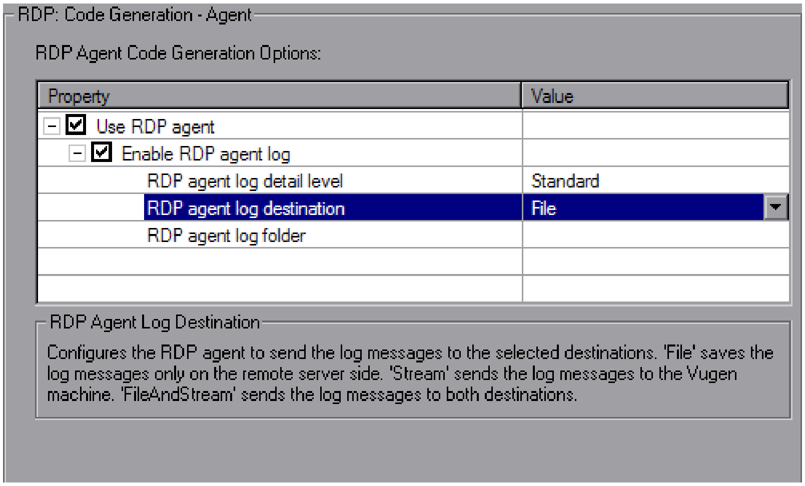
- **Double-click timeout (msec).** The maximum time in milliseconds between two consecutive mouse button clicks, to be considered a double-click. The default is 500 milliseconds.
- **Prefix for snapshot names.** The prefix for snapshot file names generated in the current script. This is useful when merging scripts—you can specify a different prefix for each script. The default is **snapshot_**.

- **Prefix for clipboard parameters.** The prefix for clipboard parameters generated in the current script. This is useful when merging scripts—you can specify a different prefix for each script. The default is **ClipboardDataParam_**.

- **Correlate clipboard parameters.** Replaces the recorded clipboard text sent by the user, with the correlated parameter containing the same text as received from the server.

RDP Agent Code Generation Options

The Agent Code Generation Options control the way the agent for Microsoft Agent for Terminal Server functions with VuGen during recording.



RDP: Code Generation - Agent

RDP Agent Code Generation Options:

Property	Value
⊟ ☑ Use RDP agent	
⊟ ☑ Enable RDP agent log	
RDP agent log detail level	Standard
RDP agent log destination	File
RDP agent log folder	

RDP Agent Log Destination

Configures the RDP agent to send the log messages to the selected destinations. 'File' saves the log messages only on the remote server side. 'Stream' sends the log messages to the Vugen machine. 'FileAndStream' sends the log messages to both destinations.

You can set the following options:

- **Use RDP agent.** Generates script using information gathered by the RDP agent during the recording session. LoadRunner RDP agent must be installed on the server.

- **Enable RDP agent log.** Enables the RDP agent log.

  - **RDP agent log detail level.** Configures the level of detail generated in the RDP agent log with **Standard** being the lowest level of detail and **Extended Debug** being the highest level of detail.

- **RDP agent log destination.** Configures the destination of the RDP agent log data. **File** saves the log messages only on the remote server side. **Stream** sends the log messages to the Vugen machine. **FileAndStream** sends the log messages to both destinations."

- **RDP agent log folder.** The folder path on the remote server that the RDP agent log file will be generated in.

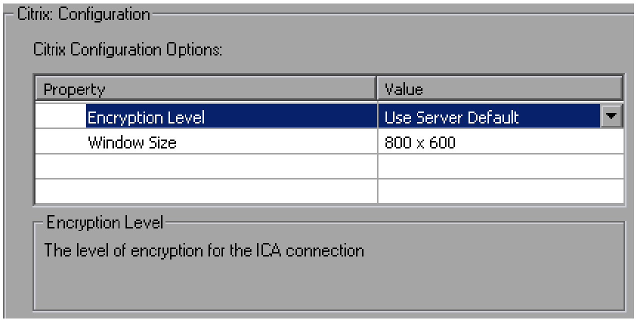
Understanding Citrix Recording Options

You can set the Citrix Recording options in the following areas.

- Configuration Recording Options
- Recorder Recording Options
- Code Generation Recording Options
- Login Recording Options (only for single protocol Citrix ICA scripts)

Configuration Recording Options

In the **Citrix:Configuration** Recording options, you set the window properties and encryption settings for the Citrix client during the recording session.



Citrix: Configuration

Citrix Configuration Options:

Property	Value
Encryption Level	Use Server Default
Window Size	800 x 600

Encryption Level

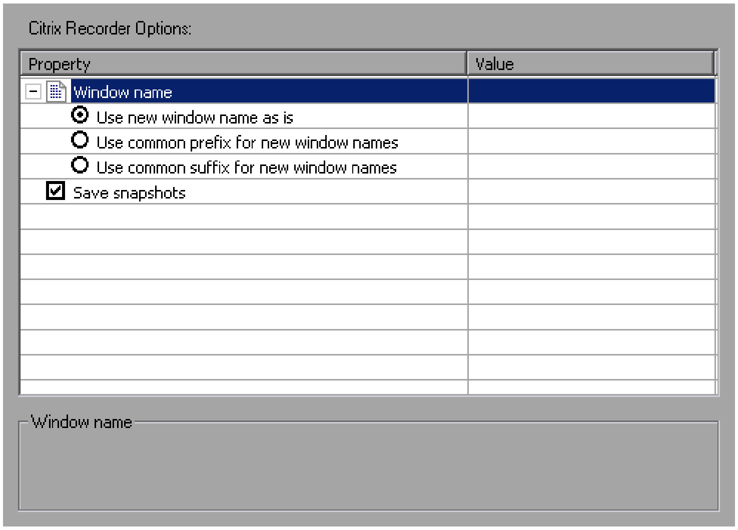
The level of encryption for the ICA connection

- **Encryption Level.** The level of encryption for the ICA connection: **Basic**, **128 bit for login only**, **40 bit**, **56 bit**, **128 bit**, or **Use Server Default** to use the machine’s default.

➤ **Window Size.** The size of the client window: **640 x 480**, **800 x 600** (default), **1024 x 768, 1280 x 1024,** or **1600 x 1200**.

Recorder Recording Options

The **Citrix:Recorder** Recording options let you specify how to generate window names where the window titles change during recording. You can also specify whether to save snapshots of the screens together with the script files and whether to generate text synchronization functions.



Citrix Recorder Options:

Property	Value
Window name	
● Use new window name as is	
○ Use common prefix for new window names	
○ Use common suffix for new window names	
☑ Save snapshots	

Window name

Window Name

In some applications, the active window name changes while you are recording. If you try to replay the script as is, the Vuser uses the original window name and the replay may fail. Using the recording options, you can specify a naming convention for the windows in which VuGen uses a common prefix or common suffix to identify the window.

For example, if the original window's name is "untitled - Notepad" where the name changes during application's run to "my_test - Notepad", you can instruct VuGen to use the common suffix only, "Notepad".

The following options are available for generating window names during recording.

- **Use new window name as is.** Set the window name as it appears in the window title. (default)
- **Use common prefix for new window names.** Use the common string from the beginning of the window titles, as a window name.
- **Use common suffix for new window names.** Use the common string from the end of the window titles, as a name.

Note: Alternatively, you can modify the window names in the actual script after recording. In the Script view, locate the window name, and replace the beginning or end of the window name with the "*" wildcard notation.

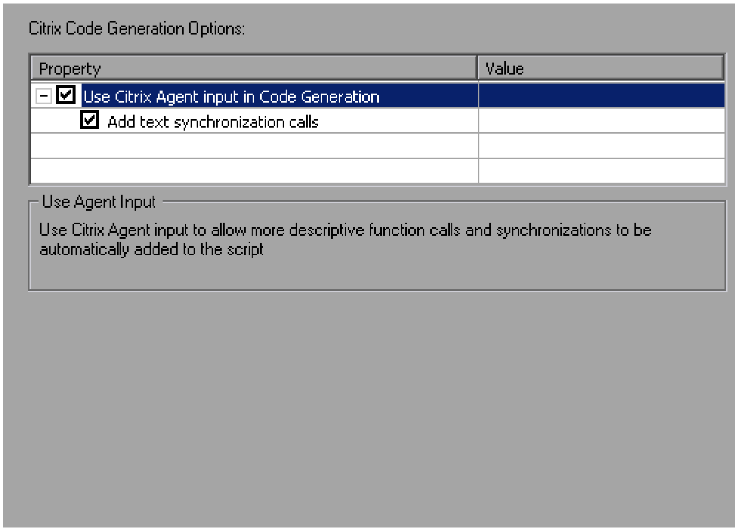
```
ctrx_sync_on_window ("My Application*", ACTIVATE, …CTRX_LAST);
```

Snapshots

The **Save snapshots** option instructs VuGen to save a snapshot of the Citrix client window for each script step, when relevant. We recommend that you enable this option to provide you with a better understanding of the recorded actions. Saving snapshots, however, uses more disk space and slows down the recording session.

Code Generation Recording Options

The **Recording:Code Generation** Recording options let you configure the way VuGen captures information during recording.



Citrix Code Generation Options:

Property	Value
☑ Use Citrix Agent input in Code Generation	
☑ Add text synchronization calls	

Use Agent Input

Use Citrix Agent input to allow more descriptive function calls and synchronizations to be automatically added to the script

- **Use Citrix Agent input in Code Generation.** Use the Citrix Agent input to generate a more descriptive script with additional synchronization functions (enabled by default).
- **Add text synchronization calls.** Adds text synchronization **Sync on Text** steps before each mouse click (disabled by default).

Text synchronization steps that you add manually during the recording are not affected by the above settings—they appear in the script even if you disable the above options. For more information about adding **Sync on Text** steps during recording, see Citrix Protocol in *Volume II-Protocols*.

The above options are also available for regenerating a script. For example, if you originally recorded a script with **Add text synchronization calls** disabled, you can regenerate after to recording to include text synchronization. For more information about regenerating your script, see "Regenerating a Vuser Script" on page 104.

Note: If you use the VuGen 8.1 or higher to regenerate a script from an earlier version of VuGen, the script will no longer be compatible with earlier versions—it behaves as if you created a new script.

Login Recording Options

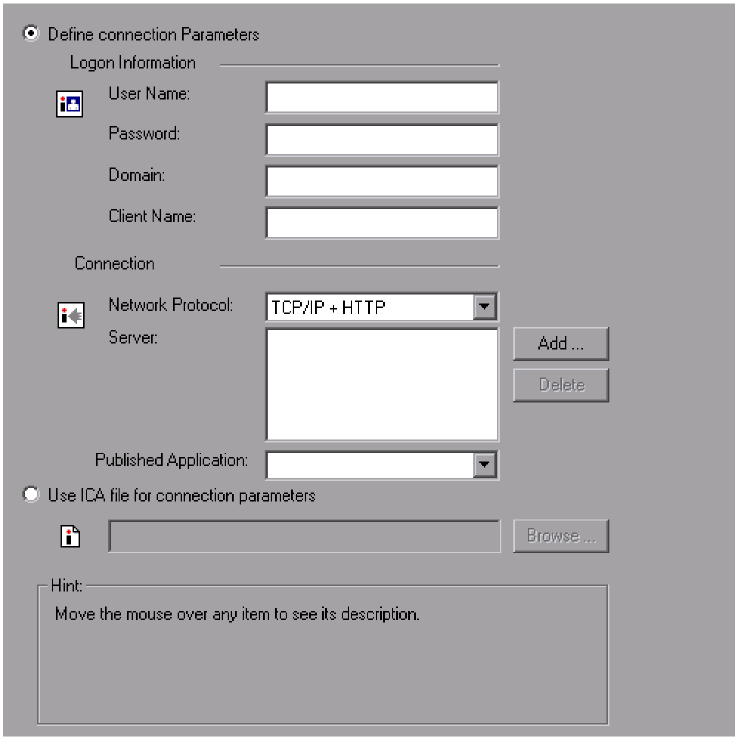
In the **Citrix:Login** Recording options, you set the connection and login information for the recording session. (When working with NFUSE, the Login options are not available since the login is done through the Web pages.)

You can provide direct login information or instruct VuGen to use an existing configuration stored in an **ica** file.

You must provide the name of the server—otherwise the connection VuGen generates a ctrx_connect_server function:

```
ctrx_connect_server("steel", "test", "test", "testlab", CTRX_LAST);
```

If you do not provide login information, you are prompted for the information when the client locates the specified server.



● Define connection Parameters

Network Protocol: TCP/IP + HTTP

○ Use ICA file for connection parameters

Hint:
Move the mouse over any item to see its description.

You provide the following user and server information for the Citrix session.

Logon Information. This section contains the login information:

- the **User Name** for the Citrix user.
- the **Password** for the Citrix user.
- the **Domain** of the Citrix user.
- the **Client Name**, by which the MetaFrame server identifies the client (optional).

Connection. This section contains the server information:

- **Network Protocol**. the preferred TCP/IP or TCP/IP+HTTP. Most Citrix Servers support TCP/IP. Certain servers, however, are configured by the administrators to allow only TCP/IP with specific HTTP headers. If you encounter a communication problem, select the TCP/IP+HTTP option.
- **Server.** The Citrix server name. To add a new server to the list, click **Add**, and enter the server name (and its port for TCP/IP + HTTP). Note that multiple servers apply only when you specify a Published Application. If you are connecting to the desktop without a specific application, then list only one server.
- **Published Application.** The name of the **Published Application** as it is recognized on Citrix server. The drop-down menu contains a list of the available applications. If you do not specify a published application, VuGen uses the server's desktop. If you added or renamed a published application, close the Recording options and reopen them to view the new list.

  To change the name of the published application on the Citrix client, you must make the change on the Citrix Server machine. Select **Manage Console** > **Application** and create a new application or rename an existing one.

  Note that if you do not specify a published application, Citrix load balancing will not work. To use load balancing when accessing the server's desktop, register the desktop as a published application on the server machine, and select this name from the **Published Application** drop-down list.

Using an ICA File with Connection Parameters

If you have an existing .ica file with all of the relevant configuration information, select **Use ICA file for connection parameters**. In the following row, specify the full path of the .ica file.

For information about the format of an ICA file, see "Understanding ICA Files" on page 316 in *Volume II-Protocols* and the Citrix Website, www.citrix.com.

Setting the Citrix Recording Options

Before recording, you set the desired recording options.

To set the Citrix recording options:

1 Open the Recording Options dialog box. Select **Tools** > **Recording Options** or click the **Options** button in the Start Recording dialog box. The keyboard shortcut key is CTRL+F7.

2 Select the **Citrix:Login** node (only for single protocol Citrix ICA scripts).

- ➤ If you have an existing ica file with all of the relevant configuration information, select **Use ICA file for connection parameters**. Specify the full path of the ica file, or click the **Browse** button and locate the file on the local disk or network.

- ➤ If you do not have an ica file, select **Define connection parameters**. This is the default setting. Enter the **Connection** and **Identification** information.

3 Select the **Citrix:Configuration** node. Select an encryption level and a window size.

4 Select the **Citrix:Code Generation** node. To use information from the Citrix agent for a more descriptive script, enable the node's options.

5 Select the **Citrix:Recorder** node. Specify how to generate window names for windows whose titles change during the recording session.

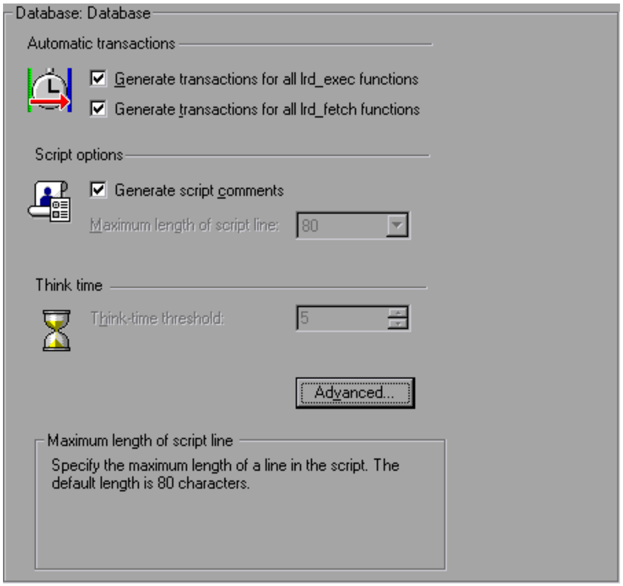
6 To prevent VuGen from saving a snapshot for each step, clear **Save snapshots**.

7 When recording an NFUSE session, set the Web recording mode to URL-based. Select the **General:Recording** recording option and select **URL-based script**.

8 Click **OK** to accept the setting and close the dialog box.

Database Recording Options

Before you record a database session, you set the recording options. You can set basic recording options for automatic function generation, script options, and think time:



Database: Database

Automatic transactions

☑ Generate transactions for all lrd_exec functions

☑ Generate transactions for all lrd_fetch functions

Script options

☑ Generate script comments

Maximum length of script line: 80

Think time

Think-time threshold: 5

Maximum length of script line

Specify the maximum length of a line in the script. The default length is 80 characters.

- **Automatic Transactions.** Marks every **lrd_exec** and **lrd_fetch** function as a transaction. When these options are enabled, VuGen inserts **lr_start_transaction** and **lr_end_transaction** around every **lrd_exec** or **lrd_fetch** function. By default, automatic transactions are disabled.

- **Script Options.** Generates comments into recorded scripts, describing the **lrd_stmt** option values. In addition, you can specify the maximum length of a line in the script. The default length is 80 characters.

- **Think Time.** VuGen automatically records the operator's think time. You can set a threshold level, below which the recorded think time will be ignored. If the recorded think time exceeds the threshold level, VuGen places an **lr_think_time** statement before LRD functions. If the recorded think time is below the threshold level, an **lr_think_time** statement is not generated. The default value is five seconds.

To set the Database recording options:

1 Select **Tools** > **Recording Options.** The Recording Options dialog box opens.

2 Select **Generate transactions for all lrd_exec functions** to enable automatic transactions for **lrd_exec** statements.

Select **Generate transaction for all lrd_fetch functions** to enable automatic transactions for **lrd_fetch** statements.

3 Select **Generate script comments** to instruct VuGen to insert descriptive comments within the script.

4 To change the maximum length of a line in the VuGen editor, specify the desired value in the **Maximum length of script line** box.

5 To change the think-time threshold value from the five second default, specify the desired value in the **Think-time threshold** box.

You can also set advanced recording options relating to the trace level, Ctlib function generation, and the code generation buffer.

Database Advanced Recording Options

In addition to the basic recording options, you can configure advanced options for the log file detail, CtLib specific functions, buffer size, and the recording engine.

➤ **Recording Log Options.** You can set the detail level for the trace and ASCII log files. The available levels for the trace file are **Off**, **Error Trace**, **Brief Trace**, or **Full Trace**. The error trace only logs error messages. The Brief Trace logs errors and lists the functions generated during recording. The Full Trace logs all messages, notifications, and warnings.

You can also instruct VuGen to generate ASCII type logs of the recording session. The available levels are **Off**, **Brief detail**, and **Full detail.** The Brief detail logs all of the functions, and the Full detail logs all of the generated functions and messages in ASCII code.

➤ **CtLib Function Options.** You can instruct VuGen to generate a send data time stamp or an extended result set statement.

  - **Time Stamp.** By default, VuGen generates **lrd_send_data** statements with the **TotalLen** and **Log** keywords for the **mpszReqSpec** parameter. The Advanced Recording Options dialog box lets you instruct VuGen to also generate the **TimeStamp** keyword. If you change this setting on an existing script, you must regenerate the Vuser script by choosing **Tools** > **Regenerate Script.** It is not recommended to generate the **Timestamp** keyword by default. The timestamp generated during recording is different than that generated during replay and script execution will fail. You should use this option only after a failed attempt in running a script, where an **lrd_result_set** following an **lrd_send_data** fails. The generated timestamp can now be correlated with a timestamp generated by an earlier **lrd_send_data**.

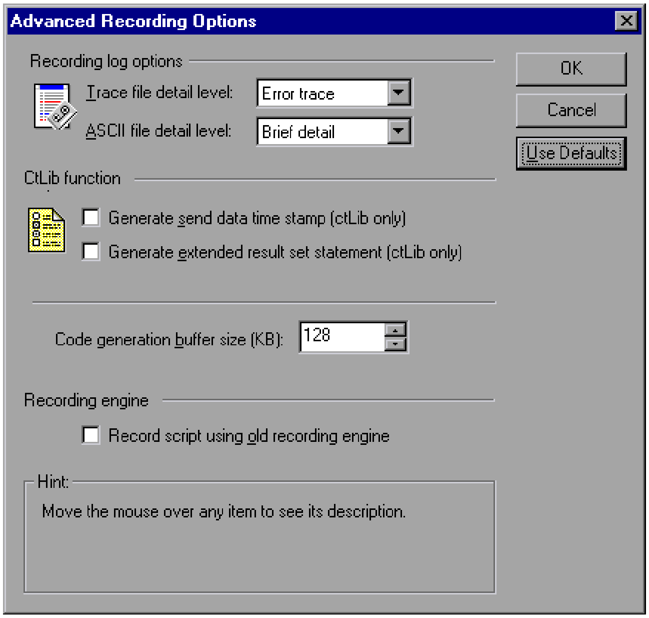
  - **Extended Result Set.** By default, VuGen generates an **lrd_result_set** function when preparing the result set. This setting instructs VuGen to generate the extended form of the **lrd_result_set** function, **lrd_result_set_ext**. In addition to preparing a result set, this function also issues a return code and type from **ct_results**.

- **Code Generation Buffer Size.** Specify in kilobytes the maximum size of the code generation buffer. The default value is 128 kilobytes. For long database sessions, you can specify a larger size.

- **Recording Engine.** You can instruct VuGen to record scripts with the older LRD recording engine for compatibility with previous versions of VuGen. This option is only available for single-protocol scripts.

To set advanced recording options:

1 Click the **Advanced** button in the Database node of the Recording Options dialog box. The Advanced Recording Options dialog box opens.



Advanced Recording Options

Recording log options

Trace file detail level: Error trace

ASCII file detail level: Brief detail

CtLib function

□ Generate send data time stamp (ctLib only)

□ Generate extended result set statement (ctLib only)

Code generation buffer size (KB): 128

Recording engine

□ Record script using old recording engine

Hint:

Move the mouse over any item to see its description.

2 Select a **Trace file detail level**. To disable the trace file, select **Off**.

3 To generate an ASCII log file, select the desired detail level from the **ASCII file detail level** box.

4 For CtLib: To instruct VuGen to generate the TimeStamp keyword for **lrd_send_data** functions, select the **Generate send data time stamp** option.

5 For CtLib: To instruct VuGen to generate **lrd_result_set_ext** instead of **lrd_result_set**, select the **Generate extended result set statement** option.

6 To modify the size of the code generation buffer from the default value of 128 kilobytes, enter the desired value in the **Code generation buffer size** box.

7 To use the old VuGen recording engine to allow backwards compatibility, select the **Record script using old recording engine** option.

8 Click **OK** to save your settings and close the Advanced Recording Options dialog box.

Setting the AMF Recording Mode

You can instruct VuGen how to generate a script from a Flash Remoting session using the AMF and Web Protocols. The options are:

- AMF and Web
- AMF Only
- Web Only

By default, VuGen generates only AMF calls in the script. To configure the recorded protocols, open the Recording options (**Tools** > **Recording Options**) and select the **General:Protocols** node. For more information, see "Adding and Removing Protocols" on page 87.

Note: If you record with one of the above options, you can regenerate the script afterwards to include or exclude other protocols. For more information about regenerating your script, see "Regenerating a Vuser Script" on page 104.

AMF and Web

If you enable both AMF and Web protocols, VuGen generates functions for the entire business process. When it encounters AMF data, it generates the appropriate AMF functions.

In the following segment, VuGen generated both Web (**web_url**) and AMF (**amf_call**, **amf_define_envelope_header_set**) functions.

```
web_url("flash",
    "URL=http://testlab:8200/flash/", "Resource=0",
    …
    "Snapshot=t1.inf",
    EXTRARES,
    "Url=movies/XMLExample.swf", "Referer=", ENDITEM,
    "Url=movies/JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);

web_link("Sample JavaBean Movie Source",
    "Text=Sample JavaBean Movie Source",
    "Snapshot=t2.inf",
    EXTRARES,
    "Url=XMLExample.swf", "Referer=", ENDITEM,
    "Url=JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);

amf_set_version("0");

amf_define_header_set("Id=amf_header_set",
    HEADER,
    "Name=amf_server_debug",
    "MustUnderstand=true",
    "Data=<object><boolean key=\"coldfusion\">true</boolean>
             <boolean key=\"""amfheaders\">false</boolean>…
    LAST);

amf_call("flashgateway.samples.FlashJavaBean.testDocument",
    "Gateway=http://testlab:8200/flashservices/gateway",
    "AMFHeaderSetId=amf_header_set",
    "Snapshot=t3.inf",
    MESSAGE,
    "Method=flashgateway.samples.FlashJavaBean.testDocument",
    "TargetObjectId=/1",
    BEGIN_ARGUMENTS,
    "<xmlString><![CDATA[<TEST message=\"test\"><INSIDETEST/>
      </TEST>]]></""xmlString>",
    END_ARGUMENTS,
    LAST);
```

AMF Only

If you are just interested in the AMF calls to emulate the Flash Remoting, you can disable the Web calls and only generate the AMF calls.

The following example shows the above session recorded with the AMF protocol enabled and the Web protocol disabled.

```
Action()
{
amf_set_version("0");

amf_define_header_set("Id=amf_header_set",
    HEADER,
    "Name=amf_server_debug",
    "MustUnderstand=true",
    "Data=<object><boolean key=\"coldfusion\">true</boolean>
             <boolean key=\"""amfheaders\">false</boolean>…
    LAST);

amf_call("flashgateway.samples.FlashJavaBean.testDocument",
    "Gateway=http://testlab:8200/flashservices/gateway",
    "AMFHeaderSetId=amf_header_set",
    "Snapshot=t3.inf",
    MESSAGE,
    "Method=flashgateway.samples.FlashJavaBean.testDocument",
    "TargetObjectId=/1",
    BEGIN_ARGUMENTS,
    "<xmlString><![CDATA[<TEST message=\"test\"><INSIDETEST
      /></TEST>]]></"
    "xmlString>",
    END_ARGUMENTS,
    LAST);
…
```

Note that this recording method may not represent a complete business process—it only displays the Flash Remoting calls that use AMF.

Web Only

The Web Only option provides a fallback to the Web HTTP technology—VuGen does not generate any AMF calls. Instead it generates **web_custom_request** functions with the Flash Remoting information.

The following shows the above segment regenerated without AMF:

```
web_url("flash",
    "URL=http://testlab:8200/flash/",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t1.inf",
    "Mode=HTML",
    EXTRARES,
    "Url=movies/XMLExample.swf", "Referer=", ENDITEM,
    "Url=movies/JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);

web_link("Sample JavaBean Movie Source",
    "Text=Sample JavaBean Movie Source",
    "Snapshot=t2.inf",
    EXTRARES,
    "Url=XMLExample.swf", "Referer=", ENDITEM,
    "Url=JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);

web_custom_request("gateway",
    "URL=http://testlab:8200/flashservices/gateway",
    "Method=POST",
    "Resource=0",
    "RecContentType=application/x-amf",
    "Referer=",
    "Snapshot=t3.inf",
    "Mode=HTML",
    "EncType=application/x-amf",

"BodyBinary=\\x00\\x00\\x00\\x01\\x00\\x10amf_server_debug\\x01\\x00\\x00\\x00`\\x0
3\\x00\ncoldfusion\\x01\\x01\\x00\namfheaders\\x01\\x00\\x00\\x03amf\\x01\\x00\\x00\\
x0Bhttpheaders\\x01\\x00\\x00\trecordset\\x01\\x01\\x00\\x05error\\x01\\x01\\x00\\x05tr
ace\\x01\\x01\\x00\\x07m_debug\\x01\\x01\\x00\\x00\t\\x00\\x01\\x00/flashgateway.sam
ples.FlashJavaBean.testDocument\\x00\\x02/1\\x00\\x00\\x004\n\\x00\\x00\\x00\\x01\\x
0F\\x00\\x00\\x00*<TEST message=\"test\"><INSIDETEST /></TEST>",
        LAST);
```

AMF Code Generation Options

When recording Flex applications, in certain cases VuGen may be unable to decode externalizable objects. This is due to a proprietary encoding scheme employed by the Flex 2 application.

To overcome this issue, you can instruct VuGen to generate a Custom Request function in the case the code is not decipherable.

The AMF Recording option instructs VuGen to attempt to generate AMF Call requests for externalizable objects. It decodes all externalizable objects as standard AMF3 objects, generating **amf_call** functions.

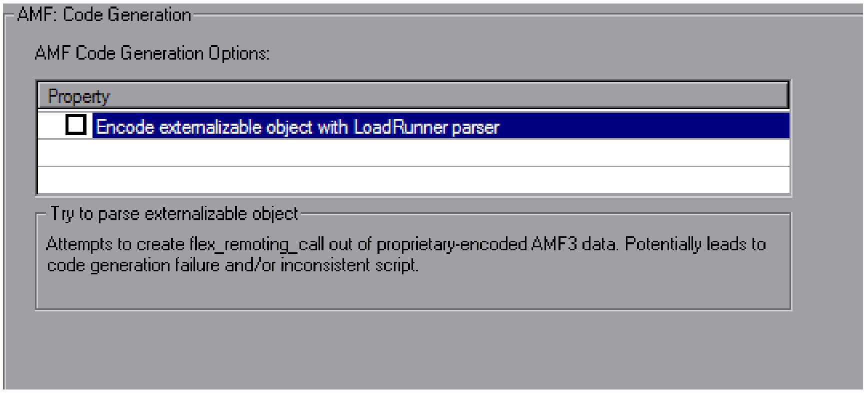
If you disable this option (default), VuGen generates a Custom Request function containing unparsed AMF3 binary data when encountering an externalizable object.

Note: Enabling this option may reduce the stability of code generation.

To set the AMF code generation for externalizable objects:

1 Open the Recording Options dialog box. Select **Tools** > **Recording Options** or click the **Options** button in the Start Recording dialog box. The keyboard shortcut key is CTRL+F7.

2 Select the **AMF:Code Generation** node. To generate standard **amf_call** functions for externalizable objects, enable the recording option.



AMF: Code Generation

AMF Code Generation Options:

Property
□ Encode externalizable object with LoadRunner parser

Try to parse externalizable object

Attempts to create flex_remoting_call out of proprietary-encoded AMF3 data. Potentially leads to code generation failure and/or inconsistent script.

Flex Code Generation Options

When recording Flex applications, in certain cases, VuGen may be unable to decode externalizable objects. This is due to a proprietary encoding scheme employed by the Flex 2 application.

To overcome this issue, you can instruct VuGen to generate a **flex_web_request** function in case the code is not decipherable.

There are two options to overcome this issue:

1) You can use the LoadRunner parser to attempt to generate Flex Remote requests for externalizable objects. It decodes all externalizable objects as standard AMF3 objects, generating flex functions.

2) You can use your server's parser to attempt to generate Flex Remote requests for externalizable objects. It decodes all externalizable objects as standard AMF3 objects, generating **flex_amf_call** functions.

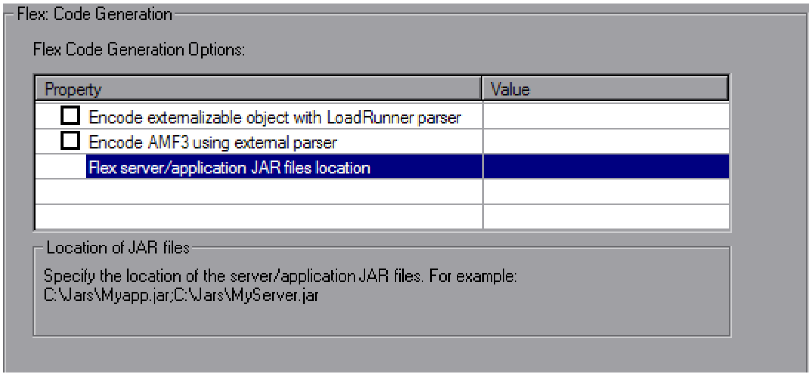
If you disable both options (default), VuGen generates a **flex_web_request** function containing unparsed AMF3 binary data when encountering an externalizable object.

Note: Enabling the encoding using the LoadRunner parse may reduce the stability of code generation.

To set the Flex code generation for externalizable objects:

1 Open the Recording Options dialog box. Select **Tools** > **Recording Options** or click the **Options** button in the Start Recording dialog box.

2 Select the **Flex:Code Generation** node. Check **Encode externalizable object with LoadRunner parser** and click **OK**.



Flex: Code Generation

Flex Code Generation Options:

Property	Value
□ Encode externalizable object with LoadRunner parser	
□ Encode AMF3 using external parser	
Flex server/application JAR files location	

Location of JAR files

Specify the location of the server/application JAR files. For example:
C:\Jars\Myapp.jar;C:\Jars\MyServer.jar

3 Regenerate your script. Check for errors in the code gereration log. If there were no **web_custom_request** or no **flex_custom_request** steps in the script, the issue has been solved and procedure is complete.

4 If the above step was not successful, go back to **Tools** > **Recording Options** and select the **Flex:Code Generation** node.

5 Check **Encode AMF3 using external parser**.

6 Copy the server and application jars to any directory on your machine and to the same directory on the load generator. For information about which servers are supported and which jars to use contact HP software support.

7 Select **Tools** > **Recording Options** and select the **Flex:Code Generation** node. To the right of the **Flex server/application JAR files location** option, specify the location of each jar file on your machine seperated with semicolons. For example C:\Jars\Myapp.jar;C:\Jars\MyServer.jar.

8 Click **OK**.

9 Regenerate your script. Check for errors in the code gereration log. If there were no **web_custom_request** or no **flex_custom_request** steps in the script, the issue has been solved and procedure is complete. If not, contact HP software support.

Microsoft .NET Recording Options

You can set both Script and Microsoft .NET-specific recording options. This section describes the recording options specific to Microsoft .NET. For information on the Script recording options, see "Setting Script Generation Preferences" on page 375.

The recording options specific to Microsoft .NET are in the area of recording settings and .NET filters.

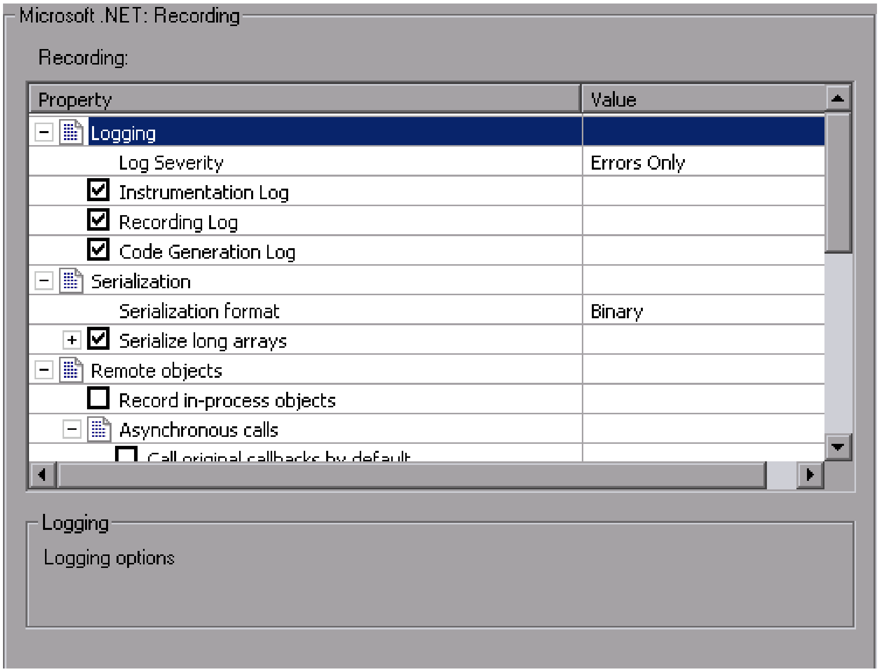
The recording settings let you control the logging, serialization, debugging, and the trace level of the logging. For more information, see "To open the .NET Recording Options dialog box, select Tools > Recording Options and select the Recording node." on page 312.

The filtering options let you select a filter for the recording, using a standard .NET Remoting, ADO.NET, Enterprise Services, or WCF filter. You can also create custom filters and configure them according to your needs. For more information, see Chapter 36, "Microsoft .NET Filters" in *Volume II-Protocols*.

For information about recording WCF duplex communication, see "WCF Recording Options" on page 318.

Some of the recording options are also relevant to code regeneration. After recording a script with certain options, you can modify the code generation options and regenerate the script with different settings or even in a different language. To change the options and regenerate a script, select **Tools** > **Regenerate Script**. In the Regenerate Script dialog box, click **Options**. For more information, see "Regenerating a Vuser Script" on page 104.

To open the .NET Recording Options dialog box, select **Tools** > **Recording Options** and select the **Recording** node.



Microsoft .NET: Recording

Recording:

Property	Value
Logging	
Log Severity	Errors Only
☑ Instrumentation Log	
☑ Recording Log	
☑ Code Generation Log	
Serialization	
Serialization format	Binary
☑ Serialize long arrays	
Remote objects	
□ Record in-process objects	
Asynchronous calls	
□ Call original callbacks by default	

Logging

Logging options

You can configure your script in the following areas:

- Logging
- Serialization Settings
- Remote Objects
- WCF Duplex Binding
- Debug Options
- Filters
- Code Generation

Logging

- The Logging options let you set the level of detail in the recording log file: **Log Severity**. Sets the level of logging to **Errors Only** (default), or **Debug**. The severity setting applies for all the logs that you enable below. You should always use the **Errors Only** log unless specifically instructed to do otherwise by HP support, since detailed logging may significantly increase the recording time.
- **Instrumentation Log.** Logs messages related to the instrumentation process (enabled by default).
- **Recording Log.** Logs messages issued during recording (enabled by default).
- **Code Generation Log.** Logs messages issued during the code generation stage (enabled by default).

Serialization Settings

The Serialization settings let you set the serialization format.

VuGen uses serialization when it encounters an unknown object during the recording, provided that the object supports serialization. An unknown object can be an input argument which was not included by the filter and therefore its construction was not recorded. Serialization helps prevent compilation errors caused by the passing of an unknown argument to a method. If an object is serialized, it is often advisable to set a custom filter to record this object.

- **Serialization format.** The format of the serialization file that VuGen creates while recording a class that supports serialization: **Binary**, **XML**, or **Both**. The advantage of the binary format is that since it is more compressed, it is quicker. The disadvantage of the binary format is that you do not have the ability to manipulate the data as you do with XML.
- **Serialize long arrays.** For long arrays containing serializable objects (for example, an array of primitives), use VuGen's serialization mechanism. Enabling this option generates LrReplayUtils.GetSerializedObject calls if the array size is equal to or larger than the threshold value.
- **Threshold value for long array size.** The threshold size for an array to be considered a **long** array. If the array size is equal to or larger than this size, VuGen serializes it when detecting serializable objects.

Tip: For XML serialization, you can view the content of the XML file. To view the file, select **View XML** from the right-click menu.

Remote Objects

This section lets you define the proxy-related recording options.

- **Record in-process objects.** Records activity between the client and server when the server is hosted in the same process as the client. Since the actions are not true client/server traffic, it is usually not of interest. When in-process methods are relevant, for example, in certain Enterprise Service applications, you can enable this option to capture them (disabled by default).

Asynchronous Calls

In the following section, you specify how VuGen should handle asynchronous calls on remote objects and their callback methods. These options are mostly relevant for .NET Remoting and WCF environments.

- **Call original callbacks by default.** Uses the recorded application's original callback when generating and replaying the script. If the callback method is explicitly excluded by a filter, the callback will be excluded even if you enable this option. If your callbacks perform actions that are not directly related to the business process, such as updating the GUI, then make sure to disable this option.
- **Generate asynchronous callbacks.** This option defines how VuGen will handle callbacks when the original callbacks are not recorded. This is relevant when the above option, **Call original callbacks** is disabled or when the callbacks are explicitly excluded.

  When you enable this option, it creates a dummy method which will be called during replay instead of the original callback. This dummy callback will be generated in the **callbacks.cs** section of the script.

  When you disable this option, VuGen inserts a NULL value for the callback and records the events as they occur.

The following segment shows script generation for a Calculator client, when **Generate asynchronous callbacks** is enabled.

```
lr.log("Event 2: CalculatorClient_1.Add(2, 3);");
Int32RetVal = CalculatorClient_1.Add(2, 3);
// Int32RetVal = 5;

callback_1 = new AsyncCallback(this.OnComplete1);
lr.log("Event 3: CalculatorClient_1.BeginAdd(2, 3, callback_1, null);");
IAsyncResult_1 = CalculatorClient_1.BeginAdd(2, 3, callback_1, null);
```

To display the callback method, OnComplete1, you click on the **callback.cs** file in the left pane.

The following segment shows script generation when the option is disabled. VuGen generates a NULL in place of the callback and records the events of the callback as they occur.

```
lr.log("Event 3: CalculatorClient_1.BeginAdd(2, 3, null, null);");
IAsyncResult_1 = CalculatorClient_1.BeginAdd(2, 3, null, null);

lr.log("Event 5: CalculatorClient_1.EndAdd(IAsyncResult_1);");
Int32RetVal = CalculatorClient_1.EndAdd(IAsyncResult_1);
// Int32RetVal = 5;

lr.log("Event 6: ((ManualResetEvent)(IAsyncResult_1.AsyncWaitHandle));");
ManualResetEvent_1 = ((ManualResetEvent)(IAsyncResult_1.AsyncWaitHandle));

lr.log("Event 7: ManualResetEvent_1.Close();");
ManualResetEvent_1.Close();
```

Note: If you recorded a script with specific recording options, and you want to modify them, you do not need to re-record the script. Instead you can regenerate the script with the new settings. For more information, see "Regenerating a Vuser Script" on page 104.

WCF Duplex Binding

You can set options for applications that use the WCF (Windows Communication Foundation). For more information, see Chapter 35, "Microsoft .NET Protocol" in *Volume II-Protocols*.

Debug Options

The debug options allow you to trace the stack and specify its size.

- **Stack Trace.** Traces the contents of the stack for each invocation within the script. It allows you to determine which classes and methods were used by your application. This can be useful in determining which references, namespaces, classes, or methods to include in your filter. Enabling the trace may affect your application's performance during recording. This trace is disabled by default.

- **Stack Trace Limit.** The maximum number of calls to be stored in the stack. The default is 20 calls. If the number of calls exceeds the limit, VuGen truncates it.

Filters

Ignore all assemblies by default. Ignore all assemblies that are not explicitly included by the selected filter. If you disable this option, VuGen looks for a matching filter rule for all assemblies loaded during the recording.

Code Generation

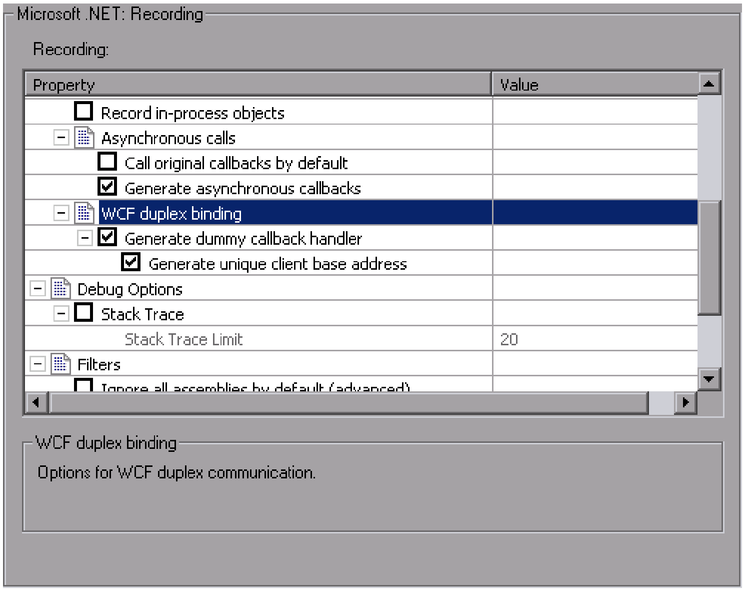
The Code Generation options let you indicate whether to show warnings and a stack trace during code generation.

- **Show Warnings.** Shows warning messages that are issued during the code generation process.
- **Show Stack Trace.** Shows the Recorded stack trace if it is available.
- **Show All Event Subscriptions.** Generate code for all event subscriptions that were recorded (disabled by default). If this option is disabled, VuGen will only generate code for events in which both the publisher (the object which invokes the event) and the subscriber (the object informed of the event) are included in the filter.

WCF Recording Options

VuGen's recording options for WCF's duplex communications enable you to generate a script that will be effective for load testing. You can set recording options on the following areas:

- Generating Dummy Callback Implementations
- Recording Dual HTTP Bindings in *Volume II-Protocols*



Microsoft .NET: Recording

Recording:

Property	Value
□ Record in-process objects	
− Asynchronous calls	
□ Call original callbacks by default	
☑ Generate asynchronous callbacks	
− WCF duplex binding	
− ☑ Generate dummy callback handler	
☑ Generate unique client base address	
− Debug Options	
− □ Stack Trace	
Stack Trace Limit	20
− Filters	
□ Ignore all assemblies by default (advanced)	

WCF duplex binding

Options for WCF duplex communication.

Generating Dummy Callback Implementations

The **Generate Dummy Callback Handler** recording option instructs VuGen to replace the original callback in duplex communication with a dummy callback.

The dummy callback implementation performs the following actions:

- **Store arguments.** When the server calls the handler during replay, it saves the method arguments to a key-value in memory map.

- **Synchronize replay.** It stops the script execution until the next response arrives. VuGen places the synchronization at the point that the callback occurred during recording. This is represented in the script by a warning:

```
#warning:  Code Generation Warning
     // Wait here for the next response.
     // The original callback during record was:…
```

As part of the synchronization, the script calls GetNextResponse to get the stored value.

```
Vuser<Callback_Name>.GetNextResponse();
```

Enabling the Dummy Callback Recording Option

By default, this option is enabled.

To enable this recording option:

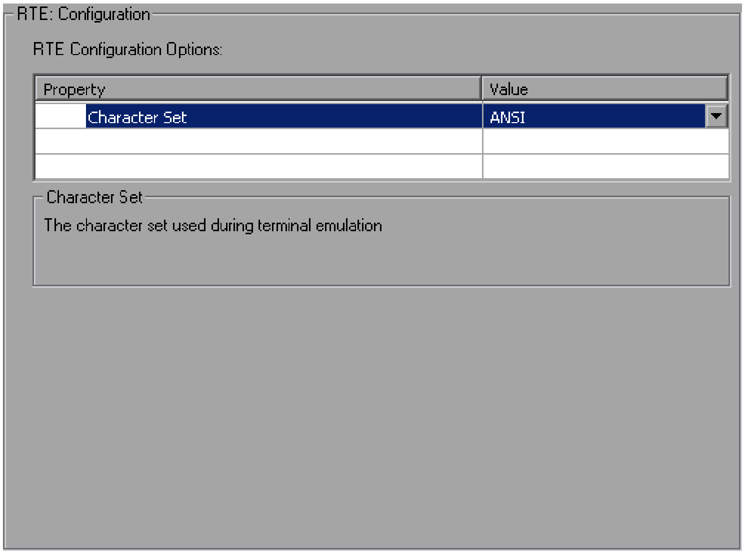
1. Select **Tools** > **Recording Options**.
2. Select the **Microsoft .NET:Recording** node.

Select **Generate Dummy Callback Handler**.

RTE Configuration Options

You can set the recording options to match the character set used during terminal emulation. The default character set is ANSI. For Kanji and other multi-byte platforms, you can specify DBCS (Double-byte Character Set).

To open the Configuration Recording Options, click the **Recording Options** button on the toolbar or select **Tools** > **Recording Options**. Select the **RTE:Configuration** node.



RTE: Configuration

RTE Configuration Options:

Property	Value
Character Set	ANSI

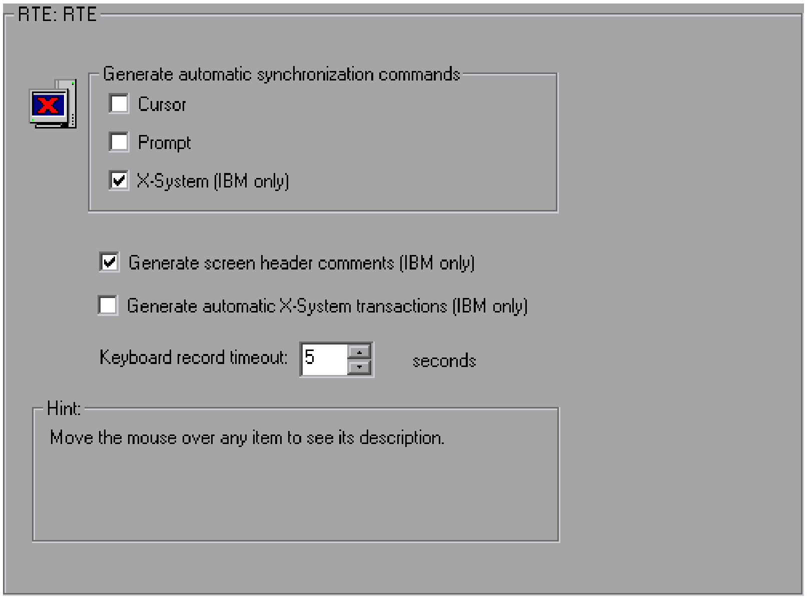
Character Set

The character set used during terminal emulation

RTE Recording Options

By setting the recording options, you can customize the code that VuGen generates for RTE functions. You use the Recording Options dialog box to set the recording options.

To open the Recording Options dialog box, click the **Recording Options** button on the toolbar, or select **Tools** > **Recording Options**. Select the **RTE:RTE** node.



RTE: RTE

Generate automatic synchronization commands

☐ Cursor

☐ Prompt

☑ X-System (IBM only)

☑ Generate screen header comments (IBM only)

☐ Generate automatic X-System transactions (IBM only)

Keyboard record timeout: 5 seconds

Hint:

Move the mouse over any item to see its description.

You can set the following recording options:

- Automatic Synchronization Commands
- Automatic Screen Header Comments (IBM terminals only)
- Automatic X-System Transactions (IBM terminals only)
- Keyboard Recording Timeout

Automatic Synchronization Commands

VuGen can automatically generate a number of TE-synchronization functions, and insert them into the script while you record.

1 You can specify that VuGen generate a **TE_wait_sync** function each time a new screen is displayed while recording. To do so, select the **X-System** check box in the Recording Options dialog box.

By default, VuGen does automatically generate a **TE_wait_sync** function each time a new screen is displayed while recording.

Note: VuGen generates **TE_wait_sync** functions when recording IBM block mode terminals only.

2 You can specify that VuGen generate a **TE_wait_cursor** function before each **TE_type** function. To do so, select the **Cursor** check box in the Recording Options dialog box.

By default, VuGen does not automatically generate **TE_wait_cursor** functions.

3 You can specify that VuGen generate a **TE_wait_text** function before each **TE_type** function (where appropriate). To do so, select the **Prompt** check box in the Recording Options dialog box.

By default, VuGen does not automatically generate a **TE_wait_text** function before each **TE_type** function.

Note: VuGen generates meaningful **TE_wait_text** functions when recording VT type terminals only. Do not use automatic **TE_wait_text** function generation when recording block-mode (IBM) terminals.

Automatic Screen Header Comments (IBM terminals only)

You can instruct VuGen to automatically generate screen header comments while recording a Vuser script, and insert the comments into the script.

Generated comments make a recorded script easier to read by identifying each new screen as it is displayed in the terminal emulator. A generated comment contains the text that appears on the first line of the terminal emulator window. The following generated comment shows that the Office Tasks screen was displayed in the terminal emulator:

```
/* OFCTSK                    Office Tasks                         */
```

To make sure that VuGen automatically generates comments while you record a script, select **Generate screen header comments** in the Recording Options dialog box.

By default, VuGen does not automatically generate screen comments.

Note: You can generate comments automatically only when using block-mode terminal emulators such as the IBM 5250.

Automatic X-System Transactions (IBM terminals only)

You can specify that VuGen record the time that the system was in the X SYSTEM mode during a scenario run. To do so, VuGen inserts a **TE_wait_sync_transaction** function after each **TE_wait_sync** function. Each **TE_wait_sync_transaction** function creates a transaction with the name "default". Each **TE_wait_sync_transaction** function records the time that the system spent in the previous X SYSTEM state.

To instruct VuGen to insert **TE_wait_sync_transaction** statements while recording, select the **Generate automatic X SYSTEM transactions** check box in the Recording Options dialog box.

By default, VuGen does not automatically generate transactions.

Keyboard Recording Timeout

When you type text into a terminal emulator while recording, VuGen monitors the text input. After each keystroke, VuGen waits up to a specified amount of time for the next key stroke. If there is no subsequent keystroke within the specified time, VuGen assumes that the command is complete. VuGen then generates and inserts the appropriate **TE_type** function into the script.

To set the maximum amount of time that VuGen waits between successive keystrokes, enter an amount in the **Keyboard record timeout** box.

By default, VuGen waits a maximum of 5 seconds between successive keystrokes before generating the appropriate **TE_type** function.

SAPGUI Recording Options

You use the recording options to set your SAP-related preferences for the recording session. To open the Recording Options dialog box, select **Tools > Recording Options** or click **Options** in the Start Recording dialog box. The keyboard shortcut is CTRL+F7.

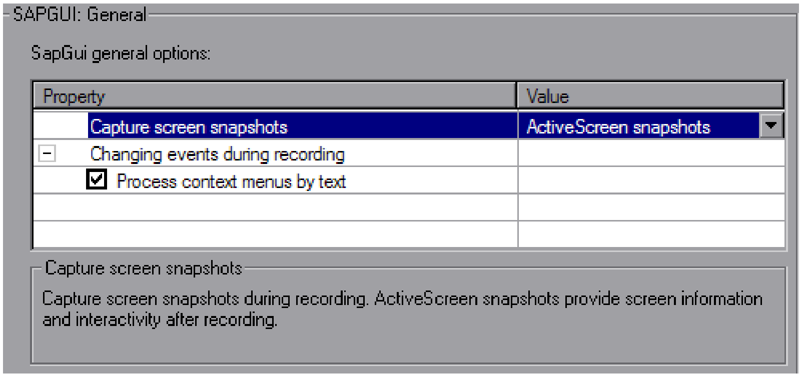
You can set recording options in the following areas:

- SAPGUI General Recording Options
- SAPGUI Code Generation Recording Options
- SAPGUI Auto Logon Recording Options

If you are recording a multi-protocol Vuser script with a SAP-Web Vuser type, see "Web, Wireless, and Oracle NCA Recording Options" on page 332 for additional recording options.

SAPGUI General Recording Options

You use these recording options to set your general preferences during the recording session in the following areas:



SAPGUI: General

SapGui general options:

Property	Value
Capture screen snapshots	ActiveScreen snapshots
− Changing events during recording	
☑ Process context menus by text	

Capture screen snapshots

Capture screen snapshots during recording. ActiveScreen snapshots provide screen information and interactivity after recording.

- **Capture Screen Snapshots.** Indicates how to save the snapshots of the SAPGUI screens as they appear during recording: **ActiveScreen snapshots**, **Regular snapshots**, or **None.** ActiveScreen snapshots provide more interactivity and screen information after recording, but they require more resources.

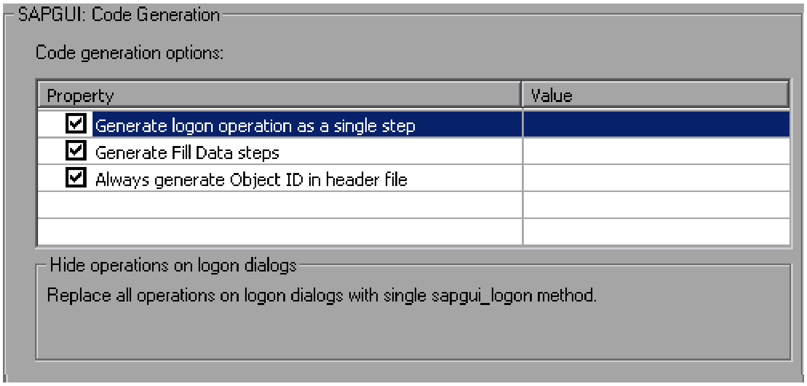
- **Process Context menus by text.** Instructs VuGen to process context menus by their text, generating a **sapgui_toolbar_select_context_menu_item_by_text** function. When disabled, VuGen processes context menus by their IDs, an advantage when working with Japanese characters. In the latter case, VuGen generates a **sapgui_toolbar_select_context_menu_item** for context menus.

To set the General recording options:

1. Open the Recording Options dialog box and select the **SAPGUI:General** node.
2. For the **Capture screen snapshots** option, indicate how to save the snapshots of the SAPGUI screens.
3. Select the method by which to process context menus: Enable **Process context menus by text**, or disable it to process context menus by their IDs.
4. Click **OK** to accept the settings and close the dialog box.

SAPGUI Code Generation Recording Options

The following recording options indicate your code generation preferences.



SAPGUI: Code Generation

Code generation options:

Property	Value
☑ Generate logon operation as a single step	
☑ Generate Fill Data steps	
☑ Always generate Object ID in header file	

Hide operations on logon dialogs

Replace all operations on logon dialogs with single sapgui_logon method.

To set the Code Generation recording options:

1 Open the Recording Options dialog box and select the **SAPGUI:Code Generation** node.

2 Select **Generate logon operation as a single step** to instruct VuGen to generate a single **sapgui_logon** method for all of the logon operations. This helps simplify the code. If you encounter login problems, disable this option.

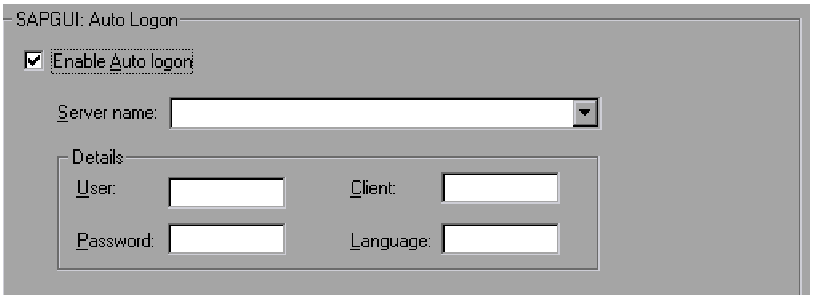
3 To generate Fill Data steps for table and grid controls—instead of separate steps for each cell, select **Generate Fill Data Steps**.

4 To create a more compact and cleaner script, select **Always generate Object IDs in header file** which places the Object IDs in a separate header file instead of in the script. When you disable this option, VuGen generates the IDs according to the specified string length in the general script setting. Note that disabling this option only increases readability—there is no difference in overhead.

5 Click **OK** to accept the settings and close the dialog box.

SAPGUI Auto Logon Recording Options

You set these recording options to log on automatically when you begin recording. The logon functions are placed in the vuser_init section of the script. The server name list contains all of the servers on the SAP Logon description list



To enable and set the Auto Logon recording options:

1. Open the Recording Options dialog box and select the **SAPGUI:Auto Logon** node.
2. Select **Enable Auto logon**.
3. Enter the Login information:
   - ➤ the SAP **Server name**
   - ➤ the **User** name for the SAP server
   - ➤ the **Password** for the SAP server
   - ➤ the **Client** name by which the SAP server identifies the client
   - ➤ the interface **Language**
4. Click **OK** to accept the settings and close the dialog box.

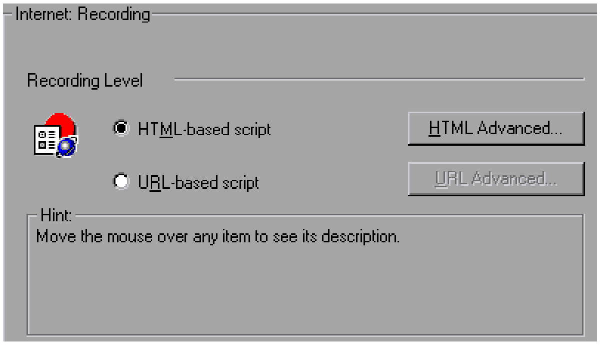
SAP-Web Recording Options

You use the recording options to set your preferences for how VuGen generates the Vuser script.

The recommended settings for the **General:Recording** node are:

For SAP Workplace recordings: URL-based script

For SAP Portal recordings: HTML-based script (the default)



Internet: Recording

Recording Level

● HTML-based script

○ URL-based script

Hint:
Move the mouse over any item to see its description.

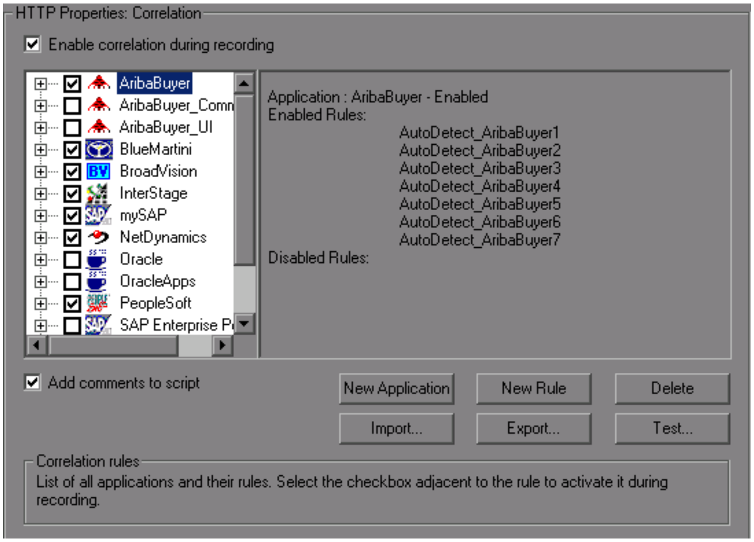
For information about the other Web related recording options, see "Web, Wireless, and Oracle NCA Recording Options" on page 332

Setting the Correlation Recording Options

To instruct VuGen to correlate your statements during recording, you set the Correlation recording options. You set these options after opening a Web Vuser script but before you begin recording the session.

To set the correlation recording options:

1 After you create a script, but before you begin recording, select **Tools** > **Recording Options** and select the **HTTP Properties:Correlation** node in the Recording Options tree.



HTTP Properties: Correlation

☑ Enable correlation during recording

- ⊞ ☑ AribaBuyer
- ⊞ □ AribaBuyer_Comn
- ⊞ □ AribaBuyer_UI
- ⊞ ☑ BlueMartini
- ⊞ ☑ BroadVision
- ⊞ ☑ InterStage
- ⊞ ☑ mySAP
- ⊞ ☑ NetDynamics
- ⊞ □ Oracle
- ⊞ □ OracleApps
- ⊞ ☑ PeopleSoft
- ⊞ □ SAP Enterprise P

Application : AribaBuyer - Enabled
Enabled Rules:
AutoDetect_AribaBuyer1
AutoDetect_AribaBuyer2
AutoDetect_AribaBuyer3
AutoDetect_AribaBuyer4
AutoDetect_AribaBuyer5
AutoDetect_AribaBuyer6
AutoDetect_AribaBuyer7
Disabled Rules:

☑ Add comments to script

Correlation rules
List of all applications and their rules. Select the checkbox adjacent to the rule to activate it during recording.

2 Select the **Enable correlation during recording** option.

3 Indicate the servers to which you want to apply the correlation rules. Select the check boxes adjacent to the server names to enable the rules for that server. To enable specific rules within a server group, click the plus sign to expand the tree and select the desired rules.

4 To add a new rule to an existing server, select one of the existing entries and click **New Rule**. Set the properties for the rule in the right pane. For more information, see "Setting Correlation Rules" on page 639 in *Volume II-Protocols*.

5 To add a set of rules for a new application, click **New Application**. Then click **New Rule** to create a rule for the application.

6 To modify the properties of an existing rule, select the rule in the left pane and modify the rules in the right pane.

7 To delete an application or rule, select it and click **Delete**. VuGen prompts you to confirm your choice before deleting the selection.

8 To export a set of correlation rules, click **Export** and save the **.cor** file to the desired location. To import a set of correlation rules created during an earlier session, click **Import** and open the file from its location.

9 Click **OK**.

Web, Wireless, and Oracle NCA Recording Options

Use the **HTTP Properties:Advanced** settings to set the recording options in the following areas:

- Internet Preferences Recording Options
- Selecting a Recording Engine
- Setting a Recording Scheme

Internet Preferences Recording Options

The Internet Preference options allow the customization of code generation settings in the area of think time, resetting contexts, saving snapshots, and the generation of **web_reg_find** functions. Note that some of these options are not available in multi-protocol mode.

- **Record think time.** (Wireless only) This setting, enabled by default, tells VuGen to record the think times and generate **lr_think_time** functions. You can also set a **Think-time Threshold** value—if the actual think-time is less than the threshold, VuGen does not generate a **lr_think_time** function.
- **Reset context for each action.** (Web, Oracle NCA only) This setting, enabled by default, tells VuGen to reset all HTTP contexts between actions. Resetting contexts allows the Vuser to more accurately emulate a new user beginning a browsing session. This option resets the HTML context, so that a contextless function is always recorded in the beginning of the action. It also clears the cache and resets the user names and passwords.
- **Save snapshot resources locally.** This option instructs VuGen to save a local copy of the snapshot resources during record and replay. This feature lets VuGen create snapshots more accurately and display them quicker.

- **Generate web_reg_find functions for page titles.** (Web, Oracle NCA only) This option enables the generation of **web_reg_find** functions for all HTML page titles. VuGen adds the string from the page's title tag and uses it as an argument for **web_reg_find**.

  - **Generate web_reg_find functions for sub-frames.** Enables the generation of **web_reg_find** functions for page titles in all sub-frames of the recorded page.

- **Add comment to script for HTTP errors while recording.** This option adds a comment to the script for each HTTP request error. An error request is defined as one that generated a server response value of 400 or greater during recording.

- **Support charset.**

  - **UTF-8.** This option enables support for UTF-8 encoding. This instructs VuGen to convert non-ASCII UTF-8 characters to the encoding of your locale's machine in order to display them properly in VuGen. You should enable this option only on non-English UTF-8 encoded pages. The recorded site's language must match the operating system language. You cannot record non-English Web pages with different encodings (for example, UTF-8 together with ISO-8859-1 or shift_jis) within the same script.

  - **EUC-JP.** For users of Japanese Windows, select this option to enable support for Web sites that use EUC-JP character encoding. This instructs VuGen to convert EUC-JP strings to the encoding of your locale's machine in order to display them properly in VuGen. VuGen converts all EUC-JP (Japanese UNIX) strings to the SJIS (Japanese Windows) encoding of your locale's machine, and adds a **web_sjis_to_euc_param** function to the script. (Kanji only)

Selecting a Recording Engine

By default, for Web(HTTP/HTML) Vusers, VuGen uses the multi-protocol recording engine for all recordings, even if you are only recording a single protocol.

To use the single-protocol recording engine for backward compatibility, select the **Record script using earlier recording engine** option in the Advanced Recording Options. If you enable this option, VuGen will use the single-protocol engine the next time you record a Web(HTTP/HTML) session.

Setting a Recording Scheme

You can further customize the recording by specifying a recording scheme in the following areas:

- Recording Custom Headers
- Filtering Content Type
- Specifying Non-Resource Content Types

Recording Custom Headers

Web Vusers automatically send several standard HTTP headers with every HTTP request submitted to the server. Click **Headers** to instruct VuGen to record additional HTTP headers. You can work in three modes: **Do not Record headers**, **Record headers in list,** or **Record headers not in list.** When you work in the first mode, VuGen does not record any headers. In the second mode, VuGen only records the checked custom headers. If you specify **Record headers not in list,** VuGen records all custom headers except for those that are checked and other risky headers.

The following standard headers are known as **risky** headers: Authorization, Connection, Content-Length, Cookie, Host, If-Modified-Since, Proxy-Authenticate, Proxy-Authorization, Proxy-Connection, Referer, and WWW-Authenticate. They are not recorded unless selected in the Header list. The default option is **Do not record headers**.

In the **Record headers in list** mode, VuGen inserts a **web_add_auto_header** function into your script for each of the checked headers that it detects. This mode is ideal for recording risky headers that are not recorded unless explicitly stated.

In the **Record headers not in list** mode, VuGen inserts a **web_add_auto_header** function into your script for each of the unchecked headers that it detects during recording.

To determine which custom headers to record, you can perform a recording session indicating to VuGen to record all headers (see procedure below). Afterwards, you can decide which headers to record and which to exclude.

In this example, the Content-type header was specified in the Record headers in list mode. VuGen detected the header and added the following statement to the script:

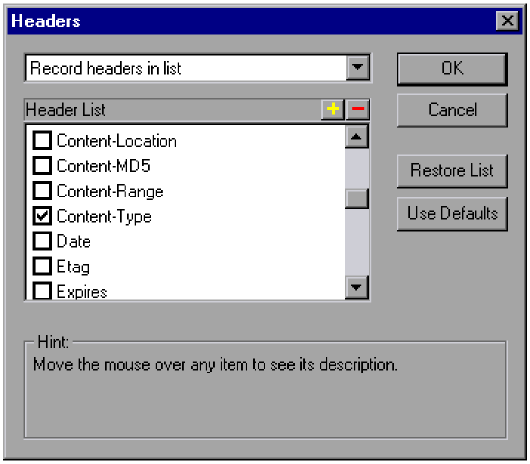
```
web_add_auto_header("Content-Type",
                        "application/x-www-form-urlencoded");
```

indicating to the server that the Content-type of the application is x-www-form-urlencode.

To control the recording of custom headers:

1 In the Recording Options tree, select the **HTTP Properties:Advanced** node.

2 Click **Headers.** The Headers dialog box opens.



Headers

Record headers in list

Header List

□ Content-Location
□ Content-MD5
□ Content-Range
☑ Content-Type
□ Date
□ Etag
□ Expires

Hint:
Move the mouse over any item to see its description.

3 Use one of the following methods:

- To instruct VuGen not to record any Headers, select **Do not record headers**.
- To record only specific headers, select **Record headers in list** and select the desired custom headers in the header list. Note that standard headers (such as **Accept**), are selected by default.
- To record all headers, select **Record headers not in list** and do not select any items in the list.
- To exclude only specific headers, select **Record headers not in list** and select the headers you want to exclude.

4 Click **Restore List** to restore the list to the corresponding default list. The **Record headers in list** and **Record headers not in list** each have a corresponding default list.

5 Click **OK** to accept the settings and close the Headers dialog box.

Filtering Content Type

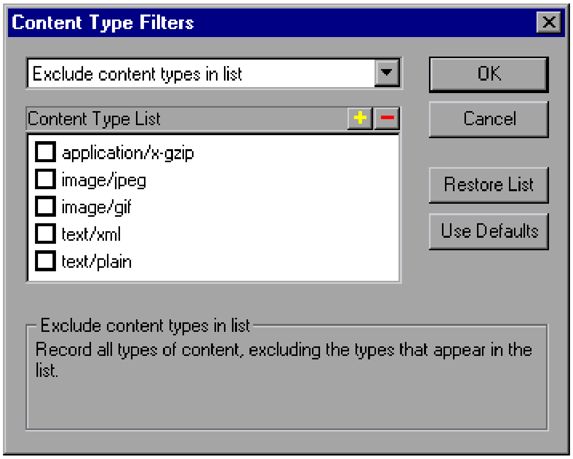
VuGen allows you to filter the content type for your recorded script. You specify the type of the content you want to record or exclude from your script. You can work in three modes: **Do not filter content types**, **Exclude content types in list,** or **Exclude content types not in list.** When you work in the first mode, VuGen does not filter any content type. In the second mode, VuGen only excludes the selected content types. If you specify **Exclude content types not in list,** VuGen filters all content type except for the ones that are checked. By default, no filters are active.

For example, if you are only interested in the text and images on your Web site, you select **Exclude content types not in list** and specify the types **text/html**, **image/gif**, and **image/jpeg.** VuGen will record all HTML pages and images, and exclude resources such as **text/css**, **application/x-javascript**, or other resources that appear on the site.

To filter content during recording:

1 In the Recording Options tree, select the **HTTP Properties:Advanced** node.

2 Click **Content Types.** The Content Type Filters dialog box opens.



Exclude content types in list

☐ application/x-gzip
☐ image/jpeg
☐ image/gif
☐ text/xml
☐ text/plain

Exclude content types in list
Record all types of content, excluding the types that appear in the list.

3 Use one of the following methods:

- ➤ To instruct VuGen not to filter any content, select **Do not filter content types**.
- ➤ To exclude only specific content types, select **Exclude content types in list** and select the desired content types from the list.
- ➤ To include only specific content types, select **Exclude content types not in list** and select the content types you want to include.

4 Click **Restore List** to restore the list to the corresponding default list. The **Exclude content types in list** and **Exclude content types not in list** each have a corresponding default list.

5 Click **OK** to accept the settings and close the Content Type Filters dialog box.

Specifying Non-Resource Content Types

When you record a script, VuGen indicates whether or not it will retrieve the resource during replay using the Resource attribute in the web_url function. If the Resource attribute is set to 0, the resource is retrieved during script execution. If the Resource attribute is set to 1, the Vuser skips the resource type.

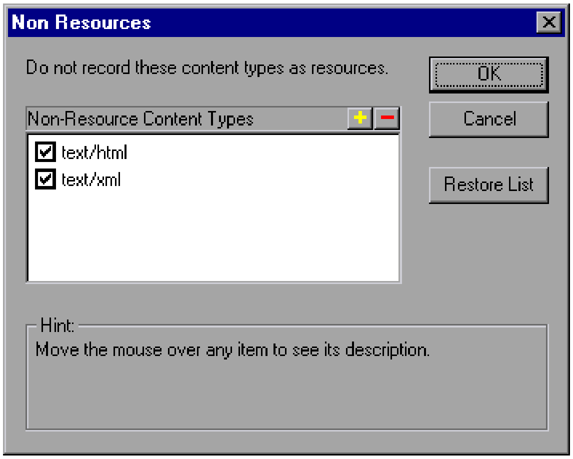
```
web_url("WebTours",
        "URL=http://localhost/WebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);
```

You can exclude specific content types from being handled as resources. For example, you can indicate to VuGen that **gif** type resources should not be handled as a resource and therefore be downloaded unconditionally. When VuGen encounters a **gif** type resource, it sets the **Resource** attribute to 0, indicating to VuGen to download gifs unconditionally during replay.

To specify which content should not be recorded as resources:

1 In the Recording Options tree, select the **HTTP Properties:Advanced** node.

2 Click **Non-Resources** to open the dialog box and display the list of content types which should not be recorded as resources.



Do not record these content types as resources.

☑ text/html
☑ text/xml

Hint:
Move the mouse over any item to see its description.

3 Click the "+" sign to add a content type to the list. Click the "-" sign to remove an existing entry.

4 Select the check boxes adjacent to the items you want to enable.

5 Click **Restore List** to restore the list to the default list.

6 Click **OK** to accept the settings and close the Non-Resources list.

17

Click and Script Recording

The Click and Script solution lets you record Web sessions on a user-action. The following chapter contains information about Click and Script recording and recording options.

This chapter includes:

- About Recording with Click and Script on page 341
- Viewing Web (Click and Script) Vuser Scripts on page 343
- Setting Click and Script Recording Options on page 344
- Setting Advanced GUI Properties on page 345
- Configuring Web Event Recording on page 348

The following information applies to the AJAX (Click and Script), Web (Click and Script), SAP (Click and Script), Oracle Web Applications 11i, and PeopleSoft Enterprise protocols.

About Recording with Click and Script

The Click and Script mechanism lets you record Web sessions on a user-action GUI level. VuGen creates a GUI-level script that intuitively represents actions in the Web interface. For example, it generates a **web_button** function when you click a button to submit information, and generates a **web_edit_field** function when you enter text into an edit box. For SAP applications, VuGen records a **sap_button** function.

For pure Web sessions, the Web (HTTP/HTML) Vuser type create a lower level script. For more information about choosing a script type for your Web session, see Chapter 37, "Web (HTTP/HTML, Click and Script) Protocols" in *Volume II-Protocols*.

Viewing Web (Click and Script) Vuser Scripts

Click and Script Vuser scripts typically contain several actions which make up a business process. By viewing the recorded functions that were generated on a GUI level, you can determine the user's exact actions during the recorded session.

For example, in a typical recording, the first stage may contain a sign-in process. The browser opens on the sign-in page, and a user signs in by submitting a user name and password and clicking **Sign In**.

For the Web (Click and Script) Vuser, VuGen generates a **web_edit_field** function that represents the data entered into an edit field. In the example that follows, a user entered text into the userid field, and a password into the pwd field which is encrypted.

```
vuser_init() {
    web_browser("WebTours",
        DESCRIPTION,
        ACTION,
        "Navigate=http://localhost:1080/WebTours/",
        LAST);

    web_edit_field("username",
        "Snapshot=t2.inf",
        DESCRIPTION,
        "Type=text",
        "Name=username",
        "FrameName=navbar",
        ACTION,
        "SetValue=jojo",
        LAST);

    web_edit_field("password",
        "Snapshot=t3.inf",
        DESCRIPTION,
        "Type=password",
        "Name=password",
        "FrameName=navbar",
        ACTION,
        "SetEncryptedValue=440315c7c093c20e",
        LAST);…
```

Setting Click and Script Recording Options

Before recording a script, you can set options that indicate what to record and how to generate the script after the recording.

You can set common recording options in the following areas: General, HTTP Properties, and Network.

The following sections discuss the GUI Properties recording options that are specific for all Vuser types that use the Click and Script mechanism: AJAX (Click and Script), Web (Click and Script), SAP (Click and Script), Oracle Applications, and PeopleSoft Enterprise Vusers. These recording options let you indicate the events to be recorded and which properties to include for each object.

- Setting Advanced GUI Properties
- Configuring Web Event Recording

For information on the other Recording Options, see the appropriate section:

- **General: Script.** See Chapter 19, "Setting Script Generation Preferences."
- **General: Recording.** See Chapter 20, "Setting Recording Options for Web Vusers."
- **Network: Port Mapping.** See "Configuring the Port Mappings" on page 397.
- **HTTP Properties: Advanced.** See "Web, Wireless, and Oracle NCA Recording Options" on page 332. Note that the **Save snapshot resources locally** and **Generate web_reg_find functions for page titles** options do not apply to GUI-based scripts (see explanation of GUI-based scripts below).
- **HTTP Properties: Correlation.** See in *Volume II-Protocols*.Note that there are built-in rules for the Oracle and PeopleSoft servers. To enable them, select the check box adjacent to your server name.

Several additional HTTP properties are only configurable for Web (HTTP/HTML) scripts.

Setting Advanced GUI Properties

VuGen lets you set Click and Script advanced options in the following areas:

- Recording Settings
- Code Generation Settings

Recording Settings

The Recording settings instruct VuGen what to record. You can enable or disable the following features:

Record rendering-related property values

Records the values of the rendering-related properties of DOM objects (for example, **offsetTop**), so that they can be used during replay. Note that this may significantly decrease the replay speed (disabled by default).

Record 'click' by mouse events

Records mouse clicks by capturing mouse events instead of capturing the click() method. Enable when the recorded application uses the DOM click() method, to prevent the generation of multiple functions for the same user action (enabled by default).

Record socket level data

Enables the recording of socket level data. If you disable this option you will need to manually add the starting URL before recording. In addition, you will be unable to regenerate the script on an HTML level. (enabled by default).

Generate snapshots for AJAX steps

Enables generation of snapshots for AJAX steps. This is unchecked by default. Checking this option can result in errors during recording.

Code Generation Settings

The Code Generation settings instruct VuGen how to generate the script after the recording. You can enable or disable the following features:

Enable generation of out-of-context steps

You can instruct VuGen to create a URL-based script for ActiveX controls and Java applets, so that they will be replayed. Since these functions are not part of the native recording, they are referred to as out-of-context recording (disabled by default).

In the following example, the script was regenerated with the out-of-context recording option enabled.

```
web_image_link("Search Flights Button",
        "Snapshot=t5.inf",
        DESCRIPTION,
        "Alt=Search Flights Button",
        "FrameName=navbar",
        ACTION,
        "ClickCoordinates=58,9",
        LAST);

web_add_cookie("MSO=SID&1141052844; DOMAIN=localhost");

web_add_cookie("MTUserInfo=hash&47&firstName&Joseph&expDate&%0A&creditCa
rd&&address1&234%20Willow%20Drive&lastName&Marshall%0A&address2&San%2
0Jose%2FCA%2F94085&username&jojo; DOMAIN=localhost");

web_url("FormDateUpdate.class",
        "URL=http://localhost:1080/WebTours/FormDateUpdate.class",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "UserAgent=Mozilla/4.0 (Windows 2000 5.0) Java/1.4.2_08",
        "Mode=HTTP",
        LAST);
…
```

If you disable this option, VuGen does not generate code for the ActiveX controls and Java applets. In the following example, VuGen only generated the **web_image_link** function—not the **web_url** functions containing the class files.

```
web_image_link("Search Flights Button",
        "Snapshot=t5.inf",
        DESCRIPTION,
        "Alt=Search Flights Button",
        "FrameName=navbar",
        ACTION,
        "ClickCoordinates=58,9",
        LAST);
```

Enable automatic browser title verification

Enables automatic browser title verification (disabled by default).

You can also customize the type of title verification.

- **Perform a title verification for.**

  each navigation. Performs a title verification only after a navigation. When a user performs several operations on the same page, such as filling out a multi-field form, the title remains the same and verification is not required.

  each step. Performs a title verification for each step to make sure that no step modified the browser title. A modified browser title may cause the script to fail.

- **Perform a title verification using the URL if the title is missing.** For browser windows without a title, perform a title verification for each step using its URL.

Configuring Web Event Recording

VuGen creates a script by recording HTML object events triggered by user actions, such as clicking the mouse or pressing a key while viewing the document.

You may find that you need to record more or fewer events than VuGen automatically records by default. You can modify the default event recording settings by using the Web Event Recording Configuration dialog box to select one of three standard configurations, or you can customize the individual event recording configuration settings to meet your specific needs.

This section describes how to configure VuGen's handling of Web Events:

- ➤ Selecting a Standard Event Recording Configuration
- ➤ Customizing the Event Recording Configuration
- ➤ Adding and Deleting Listening Events for an Object
- ➤ Modifying the Listening and Recording Settings for an Event
- ➤ Resetting Event Recording Configuration Settings

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

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

Changing the event configuration settings does not affect tests that have already been recorded. If you find that VuGen 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.

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

Selecting a Standard Event Recording Configuration

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

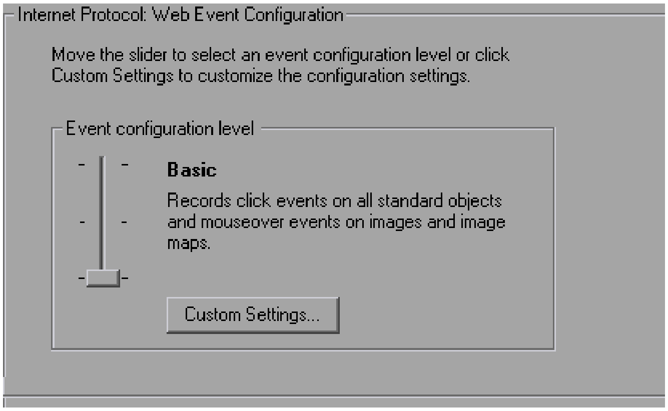
Level	Description
Basic	**Default** ➤ Always records click events on standard Web objects such as images, buttons, and radio buttons. ➤ Always records the submit event within forms. ➤ Records click events on other objects with a handler or behavior connected. For more information on handlers and behaviors, see "Listening Criteria" on page 355. ➤ Records the mouseover event on images and image maps only if the event following the mouseover is performed on the same object.

Level	Description
Medium	In addition to the objects recorded in the **Basic** level, it records click events on the <DIV>, <SPAN>, and <TD> HTML tag objects.
High	In addition to the objects recorded in the **Medium** level, it records mouseover, mousedown, and double-click events on objects with handlers or behaviors attached. For more information on handlers and behaviors, see "Listening Criteria" on page 355.

To set a standard event-recording configuration:

1 Open the Recording Options dialog box. Select **Tools** > **Recording Options**.

2 Select the **GUI Properties:Web Event Configuration** node.



Internet Protocol: Web Event Configuration

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

Event configuration level

Basic

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

3 Use the slider to select your preferred standard event recording configuration: **Basic**, **Medium**, or **High**.

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

4 Click **OK**.

Customizing the Event Recording Configuration

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

You can customize Web events for standard Web elements, such as an image, link, WebArea, WebButton, and so forth. You can also set the recording behavior for any HTML tag that you choose. You add the desired HTML tag object and then set its recording behavior. For example, you can configure VuGen to record all **mouseover** events for all **DIV** tags.

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

- Enable or disable objects to which VuGen should apply special listening or recording settings
- Add or remove events for which VuGen should listen
- Modify the listening and recording settings for an event

You can modify the event recording configuration using the following options:

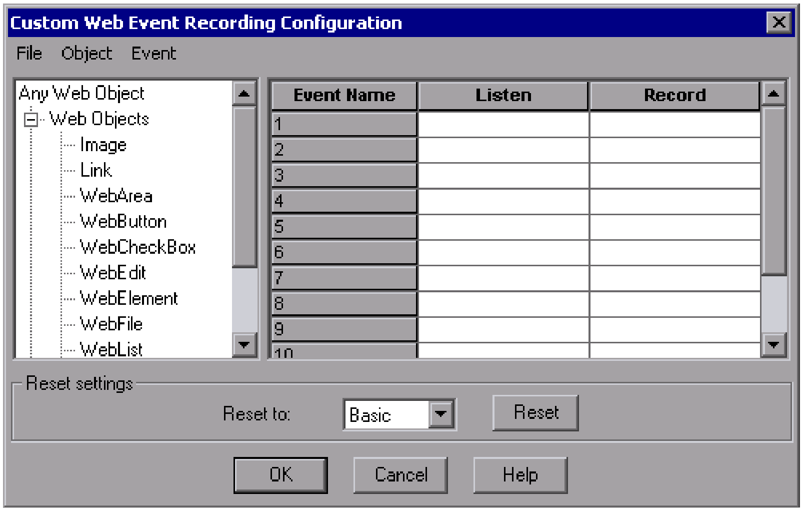
Option	Description
Event Name	Displays a list of events associated with the object. ➤ To add an event to the Events pane, select **Event** > **Add**. Select the desired event. ➤ To delete an event, click on the event in the Event Name columns, and select **Event** > **Delete**. For more information, see "Adding and Deleting Listening Events for an Object" on page 354.
Listen	The criteria for when VuGen listens to the event. ➤ **Always**. Always listens to the event. ➤ **If Handler.** Listens to the event if a handler is attached to it. A handler is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs. ➤ **If Behavior.** Listens to the event if a DHTML behavior is attached to it. A DHTML behavior encapsulates specific functionality or behavior on a page. When applied to a standard HTML element on a page, a behavior enhances that element's default behavior. ➤ **If Handler or Behavior.** Listens to the event if a handler or behavior is attached to it. ➤ **Never.** Never listens to the event. For more information, see "Modifying the Listening and Recording Settings for an Event" on page 355.
Record	Enables or disables recording of the event for the selected object, or enables recording of the event only if the subsequent event occurs on the same object.
Reset	Resets your settings to a pre-configured level: **Basic**, **Medium**, or **High**.

To customize the recording configuration for an event:

1 Open the Recording Options dialog box. Select **Tools** > **Recording Options**.

2 Select the **GUI Properties:Web Event Configuration** node.

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



Event Name	Listen	Record
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Reset to: Basic

4 Specify an object:

- ➤ To configure one of the built-in Web objects, select it in the left pane.
- ➤ To specify a different HTML tag object, select **Object** > **Add** and enter the name of the HTML tag whose event you want to record.

5 In the right pane, specify the **Listen** and **Record** behavior for each event as described above. If the event you want to record does not appear in the right pane, select **Event** > **Add** to add the event, as described below.

6 Click **OK** to save the customization.

7 Click **Reset** to reset your settings to a pre-configured level: **Basic**, **Medium**, or **High**.

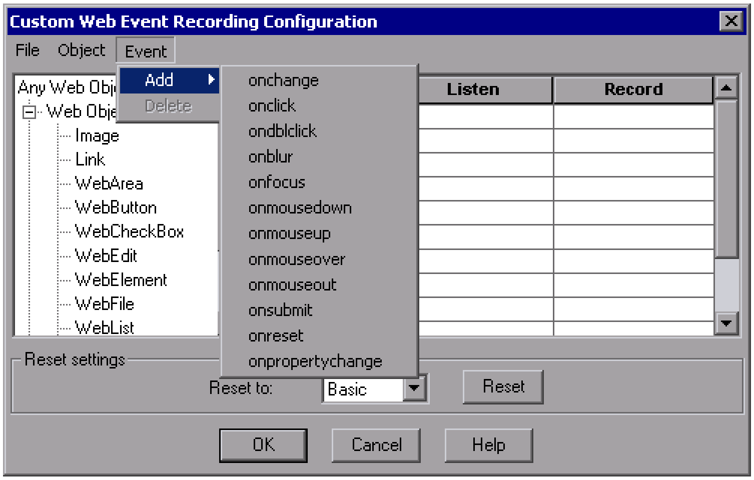
Adding and Deleting Listening Events for an Object

You can modify the list of events that trigger VuGen to listen to an object.

To add listening events for an object:

1 In the Custom Web Event Recording Configuration dialog box, select the object to which you want to add an event—one of the built-in Web objects or an HTML tag object.

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



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

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

To delete listening events for an object:

1 In the Custom Web Event Recording Configuration dialog box, select the object from which you want delete an event in the left pane.

2 In the **Event Name** column, select the event you want to delete.

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

Modifying the Listening and Recording Settings for an Event

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

Note: The listen and record settings are mutually independent. This means that you can listen to an event for a particular object, but not record it, or you can choose not to listen to an event for an object, but still record the event. For more information, see "Tips for Working with Event Listening and Recording" on page 357.

Listening Criteria

For each event, you can instruct VuGen when to listen for an event:

- **Always.** Listen every time the event occurs on the object.
- **If Handler.** Listen only if an event handler is attached to the event.
- **If Behavior.** Listen only if a DHTML behavior is attached to the event.
- **If Handler or Behavior.** Listen if an event handler or a DHTML behavior is attached to the event.
- **Never.** Never listen to the event.

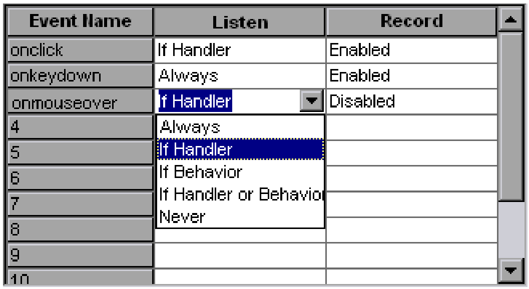
An event **handler** is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs.

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

To specify the listening criterion for an event:

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

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



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

Select a listening criteria from the list: **Always, If Handler**, **If Behavior**, **If Handler or Behavior**, or **Never**.

Recording Status

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

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

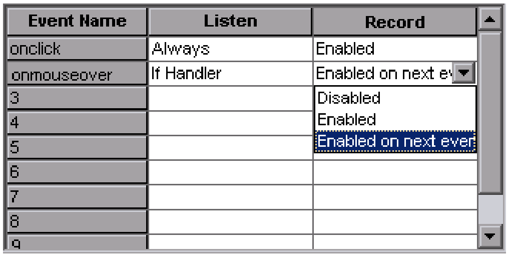
  Bubbling is the process whereby, when an event occurs on a child object, the event can travel up the chain of hierarchy within the HTML code until it encounters an event handler to process the event.

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

- ➤ **Enabled on next event.** (only applicable to the Image and WebArea objects) Same as **Enabled**, except that it records the event only if the subsequent event occurs on the same object. For example, suppose a mouseover behavior modifies an image link. You may not want to record the mouseover event each time you happen to move the mouse over this image. Because only the image that is displayed after the mouseover event enables the link event, however, it is essential that the mouseover event is recorded before a click event on the same object.

To set the recording status for an event:

1. In the Custom Web Event Recording Configuration dialog box, select the object for which you want to modify the recording status. Select **Any Web Object** to set the recording status for all Web objects in the recorded pages.
2. In the row of the event you want to modify, select a recording status from the Record column.



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

Tips for Working with Event Listening and Recording

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

- ➤ To record an event on an object, you must instruct VuGen to listen for the event, and to record the event when it occurs. You can listen for an event on a child object, even if a parent object contains the handler or behavior, or you can listen for an event on a parent object, even if the child object contains the handler or behavior.

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

For example, suppose a table cell with an **onmouseover** event handler contains two images. When a user touches either of the images with the mouse pointer, the event also bubbles up to the cell, and the bubbling includes information on which image was actually touched. You can record this mouseover event by:

- ➤ Setting **Listen** on the WebTable mouseover event to **If Handler** (so that VuGen "hears" the event when it occurs), while disabling recording on it, and then setting **Listen** on the Image mouseover event to **Never**, while setting its recording status to **Enable** (to record the mouseover event on the image after it is listened to at the WebTable level).
- ➤ Setting **Listen** on the Image mouseover event to **Always** (to listen for the mouseover event even though the image tag does not contain a behavior or handler), and setting the recording status on the Image object to **Enabled** (to record the mouseover event on the image).

- ➤ Instructing VuGen to listen for many events on many objects may lower performance, so try to limit listening settings to the required objects.
- ➤ In rare situations, listening to the object on which the event occurs (the source object) may interfere with the event.

Resetting Event Recording Configuration Settings

After you set custom settings, you can restore standard settings by instructing VuGen to use the default Web Event configuration settings.

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

To restore basic level configuration settings:

1. In the Recording Options, select the **GUI Properties:Web Event Configuration** node.
2. Click **Use Defaults**. The standard configuration slider is displayed again and all event settings are restored to the **Basic** event recording configuration level.

You can also restore the settings to a specific (base) custom configuration: Basic, Medium, or High.

To reset the settings to a custom level:

1. In the Recording Options, select the **GUI Properties:Web Event Configuration** node.
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.

18

Java Recording Options

VuGen allows you to control the way in which you record your CORBA, RMI, JMS or Jacada application. You can use the default recording options, or customize them for your specific needs.

This chapter includes:

- About Setting Java Recording Options on page 362
- Java Virtual Machine (JVM) Recording Options on page 363
- Setting Classpath Recording Options on page 365
- Recorder Options on page 366
- Serialization Options on page 368
- Correlation Options on page 370
- Log Options on page 371
- CORBA Options on page 373

About Setting Java Recording Options

Using VuGen, you record a CORBA (Common Object Request Broker Architecture) or RMI (Remote Method Invocation) Java application or applet. For recording an EJB test, see Chapter 17, "Enterprise Java Beans (EJB) Protocol" in *Volume II-Protocols*.

Before recording, VuGen lets you set recording options for the Java Virtual Machine (JVM) and for the code generation stage. Setting the recording options is not mandatory; if you do not set them, VuGen uses the default values.

The options described in this chapter were previously handled by modifying the **mercury.properties** file.

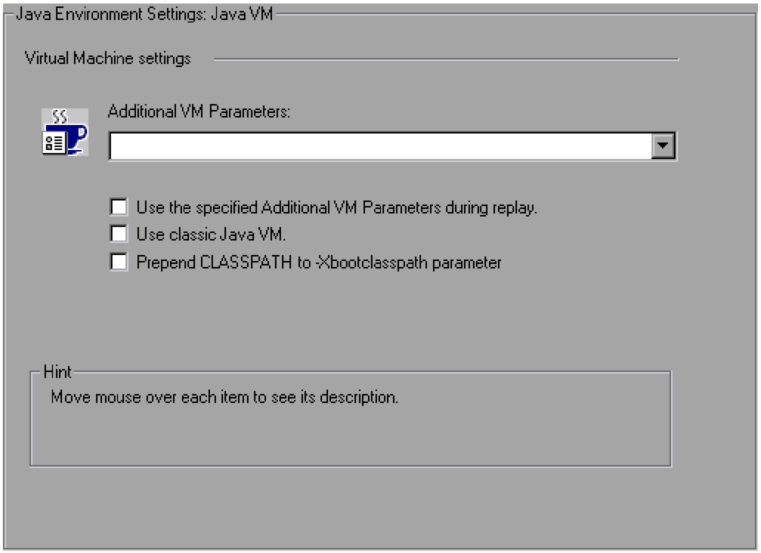
You can set recording options in the following areas:

- Java Virtual Machine (JVM) Recording Options
- Setting Classpath Recording Options
- Recorder Options
- Serialization Options
- Correlation Options
- Log Options

Java Virtual Machine (JVM) Recording Options

The **Java VM** options indicate additional parameters to use when recording Java applications.

When you record a Vuser, VuGen automatically sets the **Xbootclasspath** variable with default parameters. If you use this dialog box to set the **Xbootclasspath** with different parameters, it will use those command parameters—not the default ones.



Java Environment Settings: Java VM

Virtual Machine settings

Additional VM Parameters:

□ Use the specified Additional VM Parameters during replay.
□ Use classic Java VM.
□ Prepend CLASSPATH to -Xbootclasspath parameter

Hint

Move mouse over each item to see its description.

You can also instruct VuGen to add the Classpath before the **Xbootclasspath** (prepend the string) to create a single Classpath string.

By default, VuGen uses the classic VM during recording. You can also instruct VuGen to use another virtual machine (Sun's Java Hotspot VM).

To set the Java Virtual Machine recording options:

1 Click **Options** in the Start Recording dialog box. Select the **Java Environment Settings:Java VM** node in the Recording Options tree.

2 In the **Additional VM Parameters** box, list the Java command line parameters. These parameters may be any Java VM argument. The common arguments are the debug flag (**-verbose**) or memory settings (**-ms**, **-mx**). For more information about the Java VM flags, see the JVM documentation. In additional, you may also pass properties to Java applications in the form of a **-D** flag.

VuGen automatically sets the **-Xbootclasspath** variable (for JDK 1.2 and higher) with default parameters. When you specify **-Xbootclasspath** with parameter values as an additional parameter, VuGen uses this setting instead of the default one.

3 To use the same Additional VM parameters in replay, select the **Use the specified Additional VM Parameters during replay** check box.

4 To use the classic VM, select the **Use classic Java VM** check box (default). To use another VM (Sun's Java HotSpot), clear the check box.

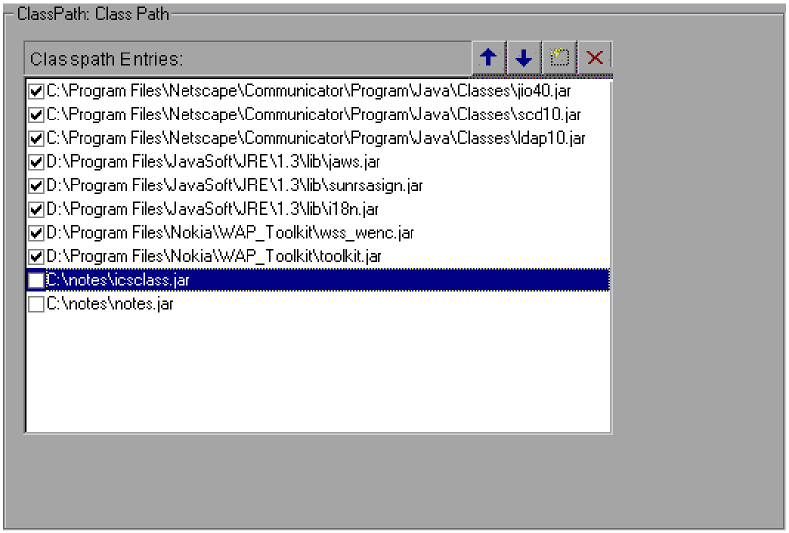
5 To add the Classpath before the **Xbootclasspath** (prepend the string), select the **Prepend CLASSPATH to -Xbootclasspath parameter** check box.

6 Click **OK** to close the dialog box and begin recording.

Setting Classpath Recording Options

The **Java Environment Settings:Classpath** node lets you specify the location of additional classes that were not included in the system's classpath environment variable. You may need these classes to run Java applications and insure proper recording.

You can browse for the required classes on your computer or network and disable them for a specific test. You can also manipulate the classpath entries by changing their order.



ClassPath: Class Path

Classpath Entries:

☑ C:\Program Files\Netscape\Communicator\Program\Java\Classes\jio40.jar
☑ C:\Program Files\Netscape\Communicator\Program\Java\Classes\scd10.jar
☑ C:\Program Files\Netscape\Communicator\Program\Java\Classes\ldap10.jar
☑ D:\Program Files\JavaSoft\JRE\1.3\lib\jaws.jar
☑ D:\Program Files\JavaSoft\JRE\1.3\lib\sunrsasign.jar
☑ D:\Program Files\JavaSoft\JRE\1.3\lib\i18n.jar
☑ D:\Program Files\Nokia\WAP_Toolkit\wss_wenc.jar
☑ D:\Program Files\Nokia\WAP_Toolkit\toolkit.jar
□ C:\notes\icsclass.jar
□ C:\notes\notes.jar

To set the Classpath recording options:

1 Click **Options** in the Start Recording dialog box. Select the **Java Environment Settings:Classpath** node in the Recording Options tree.

2 To add a classpath to the list:

Click the **Add Classpath** button. VuGen adds a new line to the classpath list.

Type in the path and name of the **jar**, **zip** or other archive file for your class. Alternatively, click the **Browse** button to the right of the field, and locate the desired file. VuGen adds the new location to the classpath list, with an enabled status.

3 To permanently remove an entry, select it and click the Delete button.

4 To disable a classpath entry for a specific test, clear the check box to the left of the entry.

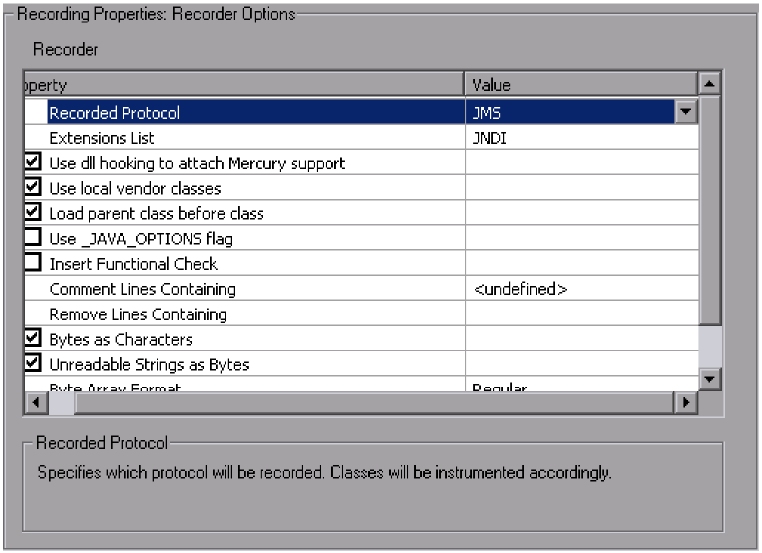
5 To move an entry down in the list, select it and click the Down arrow.

6 To move a classpath entry up within the list, select it and click the Up arrow.

7 Click **OK** to close the dialog box and begin recording.

Recorder Options

The **Recorder** options indicate which protocol to record and some of the protocol-specific settings.



Recording Properties: Recorder Options

Recorder

perty	Value
Recorded Protocol	JMS
Extensions List	JNDI
☑ Use dll hooking to attach Mercury support	
☑ Use local vendor classes	
☑ Load parent class before class	
☐ Use _JAVA_OPTIONS flag	
☐ Insert Functional Check	
Comment Lines Containing	<undefined>
Remove Lines Containing	
☑ Bytes as Characters	
☑ Unreadable Strings as Bytes	
Byte Array Format	Regular

Recorded Protocol

Specifies which protocol will be recorded. Classes will be instrumented accordingly.

- **Recorded protocol.** Specifies which protocol to record: RMI, CORBA, JMS, or Jacada. (RMI by default).
- **Extensions list.** A comma separated list of all supported extensions. Each extension has its own hooks file (JNDI by default).

- **Use DLL hooking to attach LoadRunner support.** Use DLL hooking to automatically attach LoadRunner support to any JVM.

- **Load parent class before class.** Change the loading order so that parent classed are loaded before child classes. This helps identify hooking for trees with deep inheritance. (enabled by default).

- **Use _JAVA_OPTION flag.** Forces JVM versions 1.2 and higher to use the _JAVA_OPTION environment variable which contains the desired JVM parameters (disabled by default).

- **Insert functional check.** Inserts verification code that compares the return value received during replay, to the expected return value generated during recording. This option only applies to primitive return values (disabled by default).

- **Comment lines containing.** Comment out all lines in the script containing one of the specified strings. To specify multiple strings, separate the entries with commas. By default, any line with a string containing <undefined>, will be commented out.

- **Remove lines containing.** Remove all lines containing one of the specified strings from the script. To specify multiple strings, separate the entries with commas. This feature is useful for customizing the script for a specific testing goal.

- **Bytes as characters.** Displays readable characters as characters with the necessary casting—not in byte or hexadecimal form (enabled by default).

- **Unreadable strings as bytes.** Represents strings containing unreadable characters as byte arrays. This option applies to strings that are passed as parameters to invocations (enabled by default).

- **Byte array format.** The format of byte arrays in a script: **Regular**, **Unfolded Serialized Objects**, or **Folded Serialized Objects**. Use one of the serialized object options when recording very long byte arrays. The default is **Regular**.

- **Record LoadRunner callback.** Records the LoadRunner stub object as a callback. If disabled, VuGen records the original class as the callback (enabled by default).

To set the Java Recorder options:

1 Click **Options** in the Start Recording dialog box and select the **Recording Properties:Recorder Options** node.

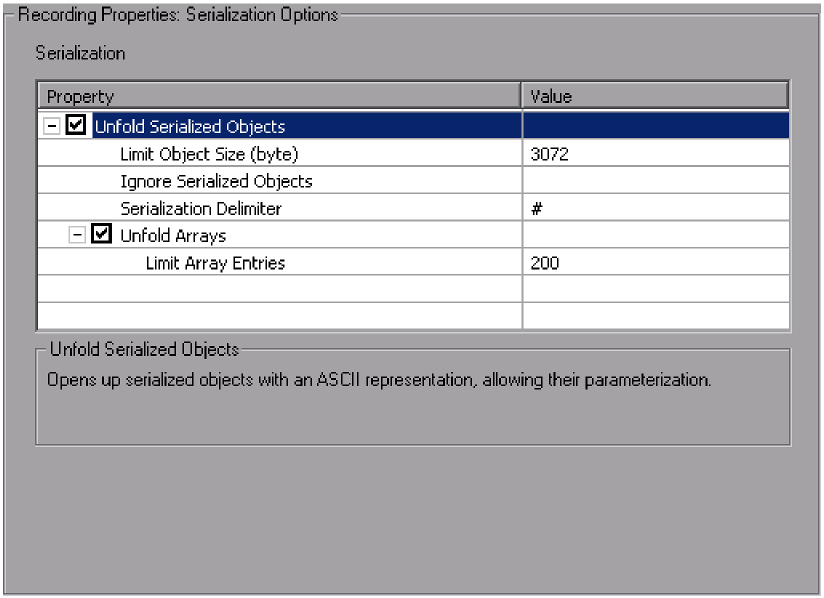
2 Set the options as desired. For the options with check boxes, select or clear the check box adjacent to the option. For options that require strings, type in the desired value.

3 To set all options to their default values, click **Use Defaults**.

4 Click **OK** to close the dialog box and begin recording.

Serialization Options

The **Serialization** options let you to control how objects are serialized. Serialization is often relevant to displaying objects in an ASCII representation in order to parameterize their values.



Recording Properties: Serialization Options

Serialization

Property	Value
− ☑ Unfold Serialized Objects	
Limit Object Size (byte)	3072
Ignore Serialized Objects	
Serialization Delimiter	#
− ☑ Unfold Arrays	
Limit Array Entries	200

Unfold Serialized Objects

Opens up serialized objects with an ASCII representation, allowing their parameterization.

The following options are available:

- **Unfold Serialized Objects.** Expands serialized objects in ASCII representation. This option allows you to view the ASCII values of the objects in order to perform parameterization (enabled by default).
  - **Limit Object Size (bytes).** Limits serializable objects to the specified value. Objects whose size exceeds this value, will not be given ASCII representation in the script. The default value is 3072.
  - **Ignore Serialized Objects.** Lists the serialized objects not to be unfolded when encountered in the recorded script. Separate objects with commas.
  - **Serialization Delimiter.** Indicates the delimiter separating the elements in the ASCII representation of objects. VuGen will only parameterize strings contained within these delimiters. The default delimiter is '#'.
- **Unfold Arrays.** Expands array elements of serialized objects in ASCII representation. If you disable this option and an object contains an array, the object will not be expanded. By default, this option is enabled—all deserialized objects are totally unfolded.
  - **Limit Array Entries.** Instructs the recorder not to open arrays with more than the specified number of elements. The default value is 200.

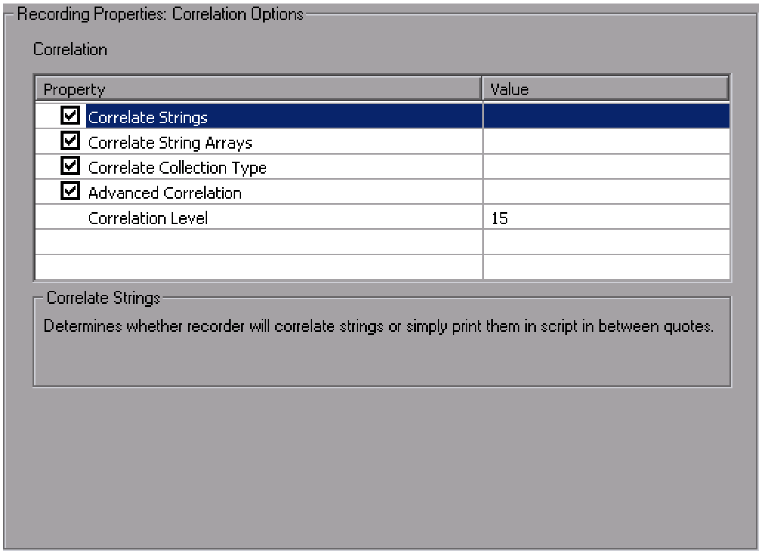
To set the Serialization options:

1. Click **Options** in the Start Recording dialog box and select the **Recording Properties:Serialization Options** node.
2. Set the options as desired. To set all options to their default values, click **Use Defaults**.
3. Click **OK** to close the dialog box and begin recording.

For more information on serialization, see Chapter 16, "Java - Correlating" in *Volume II-Protocols*.

Correlation Options

The **Correlation** options let you enable automatic correlation, and control its depth.



Recording Properties: Correlation Options

Correlation

Property	Value
☑ Correlate Strings	
☑ Correlate String Arrays	
☑ Correlate Collection Type	
☑ Advanced Correlation	
Correlation Level	15

Correlate Strings

Determines whether recorder will correlate strings or simply print them in script in between quotes.

The following options are available:

- **Correlate Strings.** Correlate all strings that require correlation. If this option is disabled, VuGen prints them in the script, wrapped in quotes (disabled by default).
- **Correlate String Arrays.** Correlate text within string arrays (enabled by default).
- **Correlate Collection Type.** Correlates objects from the Collection class for JDK 1.2 and higher (disabled by default).
- **Advanced Correlation.** Enables deep correlation in CORBA container constructs and arrays (enabled by default).
- **Correlation Level.** Indicates the level of deep correlation, the number of inner containers to be scanned (15 by default).

To set the Correlation options:

1 Click **Options** in the Start Recording dialog box and select the **Recording Properties:Correlation Options** node.

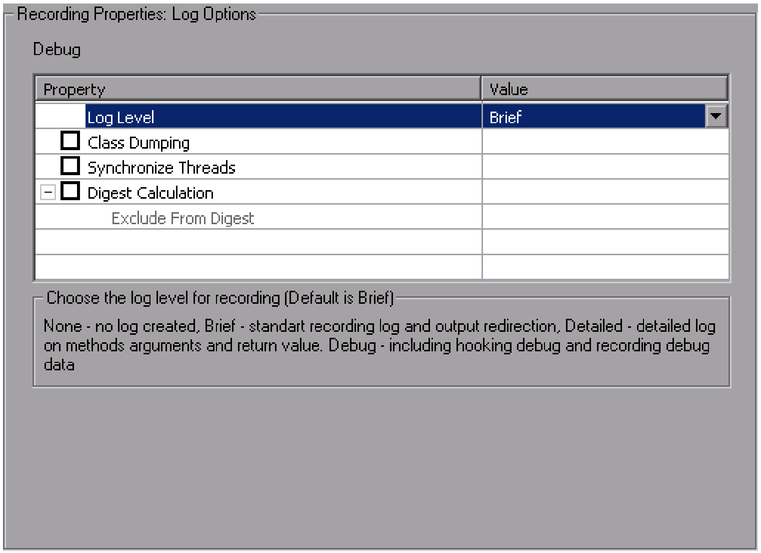
2 Enable the desired options, or for options that require values, enter the desired value. To set all options to their default values, click **Use Defaults**.

3 Click **OK** to close the dialog box and begin recording.

For more information about correlation, see Chapter 16, "Java - Correlating" in *Volume II-Protocols*.

Log Options

The **Log** recording options let you determine the level of debug information generated during recording.



Recording Properties: Log Options

Debug

Property	Value
Log Level	Brief
□ Class Dumping	
□ Synchronize Threads	
⊟ □ Digest Calculation	
Exclude From Digest	

Choose the log level for recording (Default is Brief)

None - no log created, Brief - standart recording log and output redirection, Detailed - detailed log on methods arguments and return value. Debug - including hooking debug and recording debug data

The following options are available:

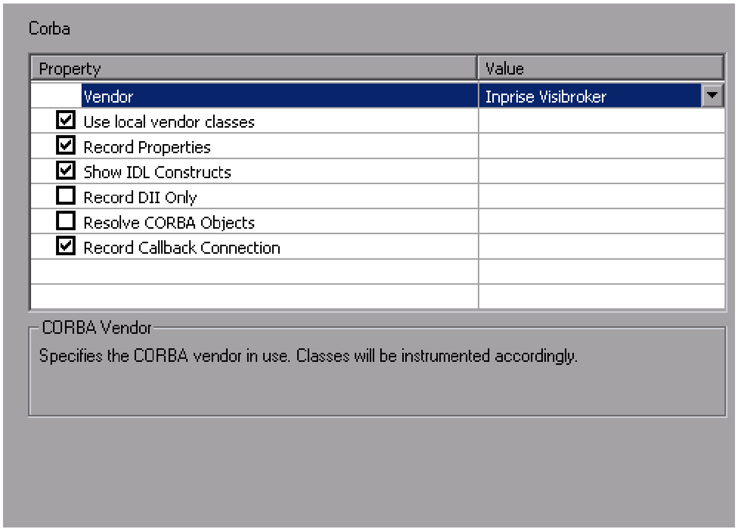
- **Log Level.** The level of recording log to generate.
  - **None**. No log file is created
  - **Brief**. Generates a standard recording log and output redirection
  - **Detailed**. Generates a detailed log for methods, arguments, and return values.
  - **Debug.** Records hooking and recording debug information, along with all of the above.
- **Class Dumping.** Dumps all of the loaded classes to the script directory. (disabled by default).
- **Synchronize Threads.** For multi-threaded applications, instructs VuGen to synchronize between the different threads (disabled by default).
- **Digest Calculation.** Generate a digest of all recorded objects (disabled by default).
  - **Exclude from Digest.** A list of objects not to be included in the digest calculation.

To set the Log options:

1. Click **Options** in the Start Recording dialog box and select the **Recording Properties:Log Options** node.
2. Select a Log level: None, Brief, Detailed, or Debug.
3. Enable the desired options, or for options that require values, enter the desired value.
4. To set all options to their default values, click **Use Defaults**.
5. Click **OK** to close the dialog box and begin recording.

CORBA Options

The following options are specific to CORBA-Java. These options let you set the CORBA specific recording properties and several callback options.



Corba

Property	Value
Vendor	Inprise Visibroker
☑ Use local vendor classes	
☑ Record Properties	
☑ Show IDL Constructs	
□ Record DII Only	
□ Resolve CORBA Objects	
☑ Record Callback Connection	

CORBA Vendor

Specifies the CORBA vendor in use. Classes will be instrumented accordingly.

The following options are available:

- **Vendor.** The CORBA vendors **Inprise Visibroker**, **Iona OrbixWeb**, or **Bea Weblogic**.
- **Use local vendor classes.** Use local vendor classes and add the **srv** folder to the BOOT classpath. If you disable this option, VuGen uses network classes and adds the script's classes to the classpath (enabled by default).
- **Record Properties.** Instructs VuGen to record system and custom properties related to the protocol (enabled by default).
- **Show IDL Constructs.** Displays the IDL construct that is used when passed as a parameter to a CORBA invocation (enabled by default).
- **Record DLL only.** Instructs VuGen to record only on a DLL level (disabled by default).
- **Resolve CORBA Objects.** When correlation fails to resolve a CORBA object, recreate it using its binary data (disabled by default).

- ➤ **Record CallBack Connection.** Instructs VuGen to generate a connect statement for the connection to the ORB, for each callback object (disabled by default).

To set the CORBA recording options:

1. Click **Options** in the Start Recording dialog box and select the **Recording Properties:Corba Options** node.
2. Enable or disable the options as desired.
3. To set all options to their default values, click **Use Defaults**.
4. Click **OK** to close the dialog box and begin recording.

19

Setting Script Generation Preferences

Before you record a script, you indicate the desired script language and the options that apply to that language.

This chapter includes:

- ➤ About Setting Script Generation Preferences on page 376
- ➤ Selecting a Script Language on page 376
- ➤ Applying the Basic Options on page 377
- ➤ Understanding the Correlation Options on page 379
- ➤ Setting Script Recording Options on page 380

The following information applies to all Vuser scripts that support multi-protocol recording.

About Setting Script Generation Preferences

Before you record a session, VuGen allows you to specify a language for script generation. The available languages for script generation vary per protocol. Some of the available languages are C, C#, Visual Basic, Visual Basic .NET, VB Script, and Javascript. By default, VuGen generates a script in the most common language for that protocol, but you can change this through the **Script** recording options.

Tip: If you record a script in one language, you can regenerate it in another language after the recording. For more information, see "Regenerating a Vuser Script" on page 104.

After you select a generation language, you can enable language-specific recording options which instruct the recorder what to include in the script and how to generate it.

If at least one of the protocols you are recording has multi-protocol capabilities, the *Script* options will be available. The only exception is when you record HTTP or WinSock as a single protocol script. In this case, the *Script* options are not available.

Selecting a Script Language

When you record a session VuGen creates a script that emulates your actions. The default script generation language is C or C# for MS .NET. For the FTP, COM/DCOM, and mail protocols (IMAP, POP3, and SMTP), VuGen can also generate a script in Visual Basic, VB Script, and Javascript.

- **C Language.** For recording applications that use complex COM constructs and C++ objects.
- **C # Language.** For recording applications that use complex applications and environments (MS .NET protocol only).
- **Visual Basic .NET Language.** For VB .NET applications, using the full capabilities of VB.

- **Visual Basic for Applications.** For VB-based applications, using the full capabilities of VB (unlike VBScript).
- **Visual Basic Scripting.** For VBscript-based applications, such as ASP.
- **Java Scripting.** For Javascript-based applications such as *js* files and dynamic HTML applications.

After the recording session, you can modify the script with regular C, C#, Visual Basic, VB Script, or Javascript code and control flow statements.

The following sections describe the scripting options. For all scripts, see "Applying the Basic Options" on page 377. To set the correlation options for non-C scripts, see "Understanding the Correlation Options" on page 379.

For further instructions, see "Setting Script Recording Options" on page 380.

Applying the Basic Options

The Basic script options allow you to control the level of detail in the generated script. Some options are only available for specific languages.

- **Close all AUT processes when recording stops.** Automatically closes all of the AUT’s (Application Under Test) processes when VuGen stops recording (disabled by default).
- **Declare primitives as locals**. Declares primitive value variables as local variables rather than class variables (C, C#, and .NET only, enabled by default).
- **Explicit variant declaration.** Declare variant types explicitly in order to handle *ByRef* variants (Visual Basic for Applications only, enabled by default).
- **Generate fixed think time after end transaction.** Add a fixed think time, in seconds, after the end of each transaction. When you enable this option, you can specify a value for the think time. The default is 3 seconds (disabled by default).

- **Generate recorded events log.** Generate a log of all events that took place during recording (disabled by default).
- **Generate think time greater than threshold.** Use a threshold value for think time. If the recorded think time is less than the threshold, VuGen does not generate a think time statement. You also specify the threshold value. The default values is 3—if the think time is less than 3 seconds, VuGen does not generate think time statements. If you disable this option, VuGen will not generate any think times (enabled by default).
- **Insert post-invocation info.** Insert informative logging messages after each message invocation (non-C only, enabled by default).
- **Insert output parameters values.** Insert output parameter values after each call (C, C#, and .NET only, disabled by default).
- **Insert pre-invocation info.** Insert informative logging messages before each message invocation (non-C only, enabled by default).
- **Maximum number of lines in action file.** Create a new file if the number of lines in the action exceeds the specified threshold. The default threshold is 60000 lines (C, C#, and .NET only, disabled by default).
- **Reuse variables for primitive return values.** Reuse the same variables for primitives received from method calls. This overrides the **Declare primitives as locals** setting (enabled by default).
- **Replace long strings with parameter.** Save strings exceeding the maximum length to a parameter. This option has an initial maximum length of 100 characters. The parameters and the complete strings are stored in the *lr_strings.h* file in the script's folder in the following format:

```
const char <paramName_uniqueID> =”string”.
```

  This option allows you to have a more readable script. It does not effect the performance of the script (enabled by default).

- **Use full type names.** Use the full type name when declaring a new variable (C# and .NET only, disabled by default).

- **Track processes created as COM local servers.** Track the activity of the recorded application if one of its sub-processes was created as a COM local server (C and COM only, enabled by default).

- **Use helpers for arrays.** Use helper functions to extract components in variant arrays (Java and VB Scripting only, disabled by default).

- **Use helpers for objects.** Use helper functions to extract object references from variants when passed as function arguments (Java and VB Scripting only, disabled by default).

For further instructions, see "Setting Script Recording Options" on page 380.

Understanding the Correlation Options

Correlation allows you to save dynamic values during test execution. These settings let you configure the extent of automatic correlation performed by VuGen while recording. All of correlation options are disabled by default. The Correlation options only apply to the non-C, such as VB Applications, VBScript, and JavaScript languages.

- **Correlate arrays.** Track and correlate arrays of all data types, such as string, structures, numbers, and so on (enabled by default).
- **Correlate large numbers.** Correlate long data types such as integers, long integers, 64-bit characters, float, and double (disabled by default).
- **Correlate simple strings.** Correlate simple, non-array strings and phrases (disabled by default).
- **Correlate small numbers.** Correlate short data types such as bytes, characters, and short integers (disabled by default).
- **Correlate structures.** Track and correlate complex structures (enabled by default).

For further instructions, see "Setting Script Recording Options" on page 380.

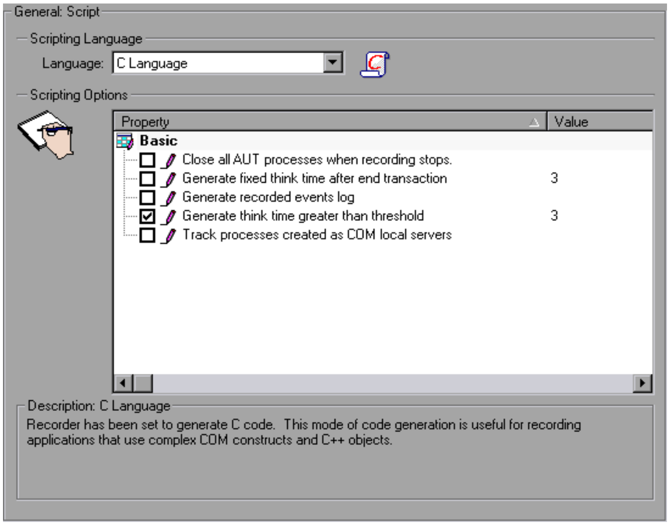
Setting Script Recording Options

You set the Recording Options before your script related initial recording. The number of available options depends on the script generation language.

To set the script recording options:

1 Open the Recording Options. Select **Tools** > **Recording Options** from the main menu or click **Options...** in the Start Recording dialog box. The Recording Options dialog box opens.

2 Select the **General:Script** node.



Language: C Language

Property	Value
Basic	
□ Close all AUT processes when recording stops.	
□ Generate fixed think time after end transaction	3
□ Generate recorded events log	
☑ Generate think time greater than threshold	3
□ Track processes created as COM local servers	

Description: C Language

Recorder has been set to generate C code. This mode of code generation is useful for recording applications that use complex COM constructs and C++ objects.

3 In the **Select Script Language** box, select a mode of code generation — *C Language* or *Visual Basic for Applications*. Use C to record applications that use complex constructs and C++ code. Use Visual Basic to record script-based applications.

4 In the **Scripting Options** section, enable the desired options by selecting the check box adjacent to it. The options are explained in the previous sections.

5 Click **OK** to save your settings and close the dialog box.

20

Setting Recording Options for Web Vusers

Before recording a Web session, you can customize the recording options.

This chapter includes:

➤ About Setting Recording Options on page 381

➤ Understanding the Recording Levels on page 382

➤ Setting the Recording Level on page 395

The following information applies to Web (HTTP/HTML), Web (Click and Script), Web/WinSocket, Oracle Web Applications 11i, and PeopleSoft Enterprise Vuser scripts.

About Setting Recording Options

VuGen enables you to generate Web Vuser scripts by recording typical processes that users perform on your Web site.

Before recording, you can configure the Recording Options and specify the information to record, the browser or client with which to record, and designate the content for your scripts.

You can set the common HTTP Properties recording options, such as proxy settings and other advanced settings. For more information see "Web, Wireless, and Oracle NCA Recording Options" on page 332

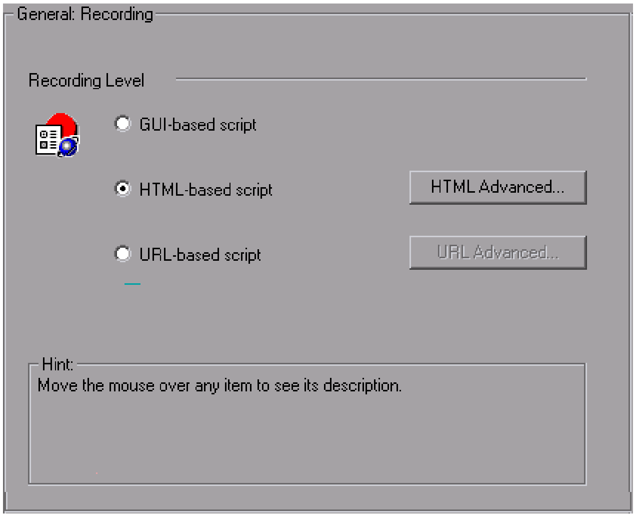
You can also set Correlation recording options for Web Vuser scripts. For more information, see Chapter 42, "Web (HTTP/HTML) Correlation Rules" in *Volume II-Protocols*.

Understanding the Recording Levels

VuGen lets you specify what information to record and which functions to use when generating a Vuser script, by selecting a recording level. The recording level you select, depends on your needs and environment. The available levels are **GUI-based script, HTML-based script,** and **URL-based script**. For Web HTTP/HTML Vusers, only the two latter options are available.

Use the following guidelines to decide which recording level to use:

- For most applications, including those with JavaScript, use a **GUI-based script**. This level is also recommended for PeopleSoft Enterprise and Oracle Web Applications 11i Vusers.
- For browser applications with applets and VB Script, create an **HTML-based script**.
- For non-browser applications, use a **URL-based script**.



General: Recording

Recording Level

○ GUI-based script

● HTML-based script

○ URL-based script

Hint:
Move the mouse over any item to see its description.

The **GUI-based script** option instructs VuGen to record HTML actions as context sensitive GUI functions such as **web_text_link**.

```
/* GUI-based mode - CS type functions with JavaScript support*//
vuser_init()
{
web_browser("WebTours",
        DESCRIPTION,
        ACTION,
        "Navigate=http://localhost:1080/WebTours/",
        LAST);

web_edit_field("username",
        "Snapshot=t2.inf",
        DESCRIPTION,
        "Type=text",
        "Name=username",
        "FrameName=navbar",
        ACTION,
        "SetValue=jojo",
        LAST);
…
```

The HTML-based script level generates a separate step for each HTML user action. The steps are also intuitive, but they do not reflect true emulation of the JavaScript code.

```
/* HTML-based mode - a script describing user actions*/
...
web_url("WebTours",
        "URL=http://localhost/WebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);

web_link("Click Here For Additional Restrictions",
        "Text=Click Here For Additional Restrictions",
        "Snapshot=t4.inf",
        LAST);

web_image("buttonhelp.gif",
        "Src=/images/buttonhelp.gif",
        "Snapshot=t5.inf",
        LAST);
…
```

The **URL-based script** mode option instructs VuGen to record all browser requests and resources from the server that were sent due to the user's actions. It automatically records every HTTP resource as URL steps (**web_url** statements). For normal browser recordings, it is not recommended to use the URL-based mode since is more prone to correlation related issues. If, however, you are recording pages such as applets and non-browser applications, this mode is ideal.

URL-based scripts are not as intuitive as the HTML-based scripts, since all actions are recorded as web_url steps instead of web_link, web_image, and so on.

```
/* URL-based mode - only web_url functions */
…
web_url("spacer.gif",
          "URL=http://graphics.hplab.com/images/spacer.gif",
          "Resource=1",
          "RecContentType=image/gif",
          "Referer=",
          "Mode=HTTP",
          LAST);

web_url("calendar_functions.js",
          "URL=http://www.im.hplab.com/travelp/calendar_functions.js",
          "Resource=1",
          "RecContentType=application/x-javascript",
          "Referer=",
          "Mode=HTTP",
          LAST);
…
```

You can switch recording levels and advanced recording options while recording, provided that you are not recording a multi-protocol script. The option of mixing recording levels is available for advanced users for performance testing.

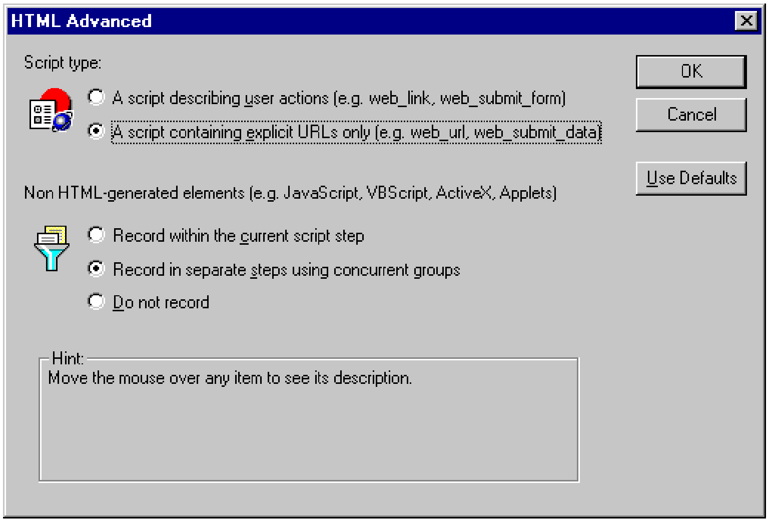
You can also regenerate a script after recording, using a different method than the original recording. For example, if your record a script on an HTML-based level, you can regenerate it on a URL-based level. To regenerate a script, select **Tools** > **Regenerate Script** and click **Options** to set the recording options for the regeneration.

Setting Advanced HTML-Based Options

The **HTML-based** option, which is the default recording level for Web (HTTP/HTML) Vusers, instructs VuGen to record HTML actions in the context of the current Web page. It does not record all resources during the recording session, but downloads them during replay.

VuGen lets you set advanced options for HTML-based script in the following areas:

- Specifying Script Types
- Handling Non HTML-Generated Elements



HTML Advanced

Script type:

○ A script describing user actions (e.g. web_link, web_submit_form)
● A script containing explicit URLs only (e.g. web_url, web_submit_data)

Non HTML-generated elements (e.g. JavaScript, VBScript, ActiveX, Applets)

○ Record within the current script step
● Record in separate steps using concurrent groups
○ Do not record

Hint:
Move the mouse over any item to see its description.

Specifying Script Types

For HTML-based scripts, you can specify the type of script:

- A script describing user actions
- A script containing explicit URLs only

The first option, **A script describing user actions**, is the default option. It generates functions that correspond directly to the action taken. It creates URL (**web_url**), link (**web_link**), image (**web_image**), and form submission (**web_submit_form**) functions. The resulting script is very intuitive and resembles a context sensitive recording.

```
/* HTML-based mode - a script describing user actions*//
...
web_url("WebTours",
        "URL=http://localhost/WebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);

web_link("Click Here For Additional Restrictions",
        "Text=Click Here For Additional Restrictions",
        "Snapshot=t4.inf",
        LAST);

web_image("buttonhelp.gif",
        "Src=/images/buttonhelp.gif",
        "Snapshot=t5.inf",
        LAST);
…
```

The second option, **A script containing explicit URLs only**, records all links, images and URLs as **web_url** statements, or in the case of forms, as **web_submit_data**. It does not generate the **web_link**, **web_image**, and **web_submit_form** functions. The resulting script is less intuitive. This mode is useful for instances where many links within your site have the same link text. If you record the site using the first option, it records an ordinal (instance) for the link, but if you record using the second option, each link is listed by its URL. This facilitates parameterization and correlation for that step.

The following segment illustrates a session recorded with a script containing explicit URLs only selected:

```
/* A HTML-based script containing explicit URLs only*//
…
web_url("Click Here For Additional Restrictions",
          "URL=http://www.hplab.com/restrictions.html",
          "TargetFrame=",
          "Resource=0",
          "RecContentType=text/html",
          "Referer=http://www.hplab.com/home?…
          "Snapshot=t4.inf",
          "Mode=HTML",
          LAST);

web_url("buttonhelp.gif",
          "URL=http://www.hplab.com/home?com/rstr?BV_EngineID...,
          "TargetFrame=main",
          "Resource=0",
          "RecContentType=text/html",
          "Referer=http://www.hplab.com/home?…
          "Snapshot=t5.inf",
          "Mode=HTML",
          LAST);
…
```

Handling Non HTML-Generated Elements

Many Web pages contain non-HTML elements, such as applets, XML, ActiveX elements, or JavaScript. These non-HTML elements usually contain or retrieve their own resources. For example, a JavaScript **js** file, called from the recorded Web page, may load several images. An applet may load an external text file. Using the following options, you can control how VuGen records non HTML-generated elements.

The following options are available:

- Record within the current script step (default)
- Record in separate steps using concurrent groups
- Do not record

The first option, **Record within the current script step**, does not generate a new function for each of the non HTML-generated resources. It lists all resources as arguments of the relevant functions, such as web_url, web_link, and web_submit_data. The resources, arguments of the Web functions, are indicated by the EXTRARES flag. In the following example, the web_url function lists all of the non HTML-generated resources loaded on the page:

```
web_url("index.asp",
    "URL=http://www.daisy.com/index.asp",
    "TargetFrame=",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t2.inf",
    "Mode=HTML",
    EXTRARES,
    "Url=http://www.daisy.com/ScrollApplet.class", "Referer=", ENDITEM,
    "Url=http://www.daisy.com/board.txt", "Referer=", ENDITEM,
    "Url=http://www.daisy.com/nav_login1.gif", ENDITEM,
    …
    LAST);
```

The second option, **Record in separate steps using concurrent groups**, creates a new function for each one of the non HTML-generated resources—it does not include them as items in the page's functions (such as **web_url**, **web_link**, and so on). All of the **web_url** functions generated for a resource, are placed in a concurrent group (surrounded by **web_concurrent_start** and **web_concurrent_end**).

In the following example, the above session was recorded with this option selected. A web_url function was generated for the applet and text file loaded with the applet:

```
web_url("index.asp",
        "URL=http://www.daisy.com/index.asp",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t2.inf",
        "Mode=HTML",
        LAST);

web_concurrent_start(NULL);
    web_url("ScrollApplet.class",
        "URL=http://www.daisy.com/ScrollApplet.class",
        "Resource=1",
        "RecContentType=application/octet-stream",
        "Referer=",
        LAST);

    web_url("board.txt",
        "URL=http://www.daisy.com/board.txt",
        "Resource=1",
        "RecContentType=text/plain",
        "Referer=",
        LAST);
web_concurrent_end(NULL);
```

The third option, **Do not record**, instructs VuGen not to record any of the resources generated by non-HTML elements.

Note that when you work in HTML-Based mode, VuGen inserts the TargetFrame attribute in the **web_url** statement. VuGen uses this information to display the Web page correctly in the run-time browser and Test Result report.

```
web_url("buttonhelp.gif",
        "URL=http://www.hplab.com/home?com/rstr?BV_EngineID...,
        "TargetFrame=main",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.hplab.com/home?…
        "Snapshot=t5.inf",
        "Mode=HTML",
        LAST);
```

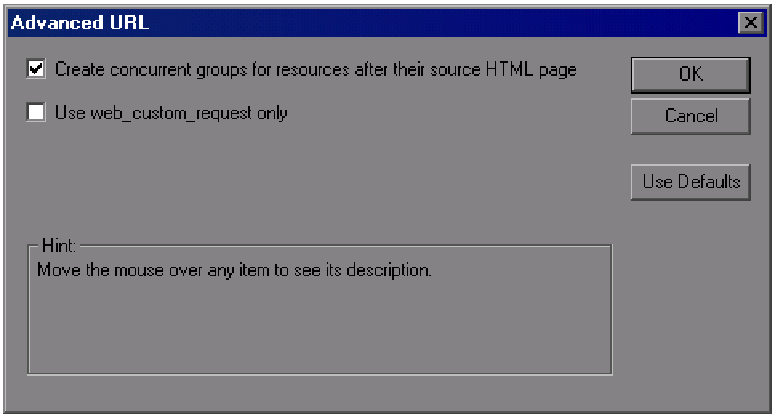
When you record the URL-based mode, VuGen records the content of all frames on the page and therefore omits the **TargetFrame** attribute.

Setting Advanced URL-Based Options

The **URL-based** mode option instructs VuGen to record all requests and resources from the server. It automatically records every HTTP resource as URL steps (**web_url** statements), or in the case of forms, as **web_submit_data**. It does not generate the **web_link**, **web_image**, and **web_submit_form** functions, nor does it record frames.

VuGen lets you set advanced options for the URL recording mode in the following area:

- Resource Handling
- Generating Custom HTTP Requests



Advanced URL

☑ Create concurrent groups for resources after their source HTML page

□ Use web_custom_request only

Hint:
Move the mouse over any item to see its description.

Resource Handling

In URL-based recording, VuGen captures all resources downloaded as a result of a browser request. By default, this option is enabled and VuGen records the resources in a concurrent group (enclosed by **web_concurrent_start** and **web_concurrent_end** statements) after the URL. Resources include files such as images, and **js** files. If you disable this option, the resources are listed as separate **web_url** steps, but not marked as a concurrent group.

The following segment illustrates a session recorded with the Create concurrent groups for resources after their source HTML page option enabled.

```
web_concurrent_start (NULL);
…
web_url("Click Here For Additional Restrictions",
        "URL=http://www.hplab.com/restrictions.html",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.hplab.com/home?…
        "Snapshot=t4.inf",
        "Mode=HTTP",
        LAST);

web_url("buttonhelp.gif",
        "URL=http://www.hplab.com/home?com/rstr?BV_EngineID...,
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.hplab.com/home?…
        "Snapshot=t5.inf",
        "Mode=HTTP",
        LAST);
…
web_concurrent_end (NULL);
```

Note that the script includes **gif**, and **js** files. This mode also includes other graphic files and imported file such as **imp**, **txt**, and cascading style sheet (**css**) files.

Generating Custom HTTP Requests

When recording non-browser applications, you can instruct VuGen to record all HTTP requests as custom requests. VuGen generates a **web_custom_request** function for all requests, regardless of their content:

```
web_custom_request("www.hplab.com",
        "URL=http://www.hplab.com/",
        "Method=GET",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTTP",
        LAST);
```

Enabling EUC-Encoded Web Pages

(This option is only for Japanese Windows.) When working with non-Windows standard character sets, you may need to perform a code conversion. A character set is a mapping from a set of characters to a set of integers. This mapping forms a unique character-integer combination, for a given alphabet. Extended UNIX Code (EUC) and Shift Japan Industry Standard (SJIS) are non-Windows standard character sets used to display Japanese writings on Web sites.

Windows uses SJIS encoding, while UNIX uses EUC encoding. When a Web server is on a UNIX machine and the client is Windows, the characters in a Web site are not displayed on the client machine properly due to the difference in the encoding methods. This affects the display of EUC-encoded Japanese characters in a Vuser script.

During recording, VuGen detects the encoding of a Web page through its HTTP header. If the information on the character set is not present in the HTTP header, it checks the HTML meta tag. If the page does not send the character set information to the HTTP header or meta tag, VuGen does not detect the EUC encoding.

If you know in advance that a Web page is encoded in EUC, you can instruct VuGen to use the correct encoding during record. To record a page in EUC-encoding, enable the **EUC** option in the Recording Options **Recording** tab (only visible for Japanese Windows).

Enabling the **EUC** option, forces VuGen to record a Web page in EUC encoding, even when it is not EUC-encoded. You should, therefore, only enable this option when VuGen cannot detect the encoding from the HTTP header or the HTML meta tag, and when you know in advance that the page is EUC-encoded.

During recording, VuGen receives an EUC-encoded string from the Web server and converts it to SJIS. The SJIS string is saved in the script's Action function. However, for replay to succeed, the string has to be converted back to EUC before being sent back to the Web server. Therefore, VuGen adds a **web_sjis_to_euc_param** function before the Action function, which converts the SJIS string back to EUC.

In the following example, the user goes to an EUC-encoded Web page and clicks a link. VuGen records the Action function and adds the web_sjis_to_euc_param function to the script before the Action function.

```
web_sjis_to_euc_param("param_link","Search");
web_link("LinkStep","Text={param_link}");
```

Setting the Recording Level

This section describes the procedure for setting the recording levels and their advanced options.

To set the recording options:

1 Select **Tools** > **Recording Options** to open the Recording Options.

2 Select the **General:Recording** node in the Recording Options tree.

3 Select a recording mode: **GUI-based** (when available), **HTML-based**, or **URL-based.**

4 For GUI-based recording, open the recording options (Ctrl + F7) and select the **GUI Properties:Advanced** node to set additional options for capturing events during recording.

For more information, see "Setting Advanced GUI Properties" on page 345.

5 For HTML-based recording, click **HTML Advanced** to set additional options for script types and the handling of non-HTML elements.

Select a script type.

Select a method for handling non-HTML resources. For more information, see "Setting Advanced HTML-Based Options" on page 386.

6 For URL-based recording, click **URL Advanced** to set additional script options for resource handling and cache enabling.

Select **Create concurrent groups for resources after their source HTML page** to enable the recording of resources and marking them as a concurrent group (surrounded by **web_concurrent_start** and **web_concurrent_end**).

Select **Enable cache** to use the browser cache during recording. If you enable this option, clear the **Clear cache before recording** check box to instruct VuGen not to clear the cache and use previously accessed pages.

Select **Use web_custom_request only** to generate all HTTP requests as **web_custom_request** functions. For more information about these options, see "Setting Advanced URL-Based Options" on page 391.

7 For users of Japanese Windows, select the **EUC** option to instruct VuGen to use EUC encoding.

If you are recording a Web site whose pages use only the EUC Encoding (Japanese content), select the **EUC** option. VuGen converts the EUC string to SJIS and adds a **web_sjis_to_euc_param** function. If the server sends this information to the browser (in an HTTP header or an HTML Meta tag), you do not need to enable this option.

21

Configuring the Port Mappings

When working with protocols that record network traffic on a socket level, you can indicate the port to which you want to map the traffic.

This chapter includes:

- ➤ About Configuring the Port Mappings on page 398
- ➤ Defining Port Mappings on page 398
- ➤ Adding a New Server Entry on page 401
- ➤ Setting the Advanced Port Mapping Options on page 403
- ➤ Setting the Port Mapping Recording Options on page 406

The following information applies to all Vuser scripts that record on a socket level: HTTP, SMTP, POP3, IMAP, Oracle NCA, and WinSocket.

About Configuring the Port Mappings

When recording Vuser scripts that record network traffic on a socket level (HTTP, SMTP, POP3, FTP, IMAP, Oracle NCA and WinSocket), you can set the Port Mapping options. Using these options, you can map the traffic from a specific server:port combination to the desired communication protocol.

The available communication protocols to which you can map are FTP, HTTP, IMAP, NCA, POP3, SMTP, and SOCKET. You create a mapping by specifying a server name, port number, or a complete server:port combination. For example, you can indicate that all traffic from the server *twilight* on port 25, should be handled as SMTP. You can also specify that all traffic from the server called *viper*, should be mapped to the FTP protocol, regardless of the port. Additionally, you can map all traffic on port 23 to SMTP, regardless of the server name.

When recording in multi-protocol mode, If at least one of the protocols records on a socket level, the *Port Mapping* options will be available. The only exception is when you record HTTP or WinSock as a single protocol script. In this case, the *Port Mapping* options are not available.

Defining Port Mappings

VuGen uses the Port Mapping settings to direct traffic via a specific server:port combination to the desired communication protocol. You can configure the Port Mapping settings in the following areas:

- **Capture level.** The level of data to capture (relevant only for HTTP based protocols):
  - **Socket level data.** Capture data using trapping on the socket level only. Port mappings apply in this case (default).
  - **WinINet level data.** Capture data using hooks on the WinINet.dll API used by certain HTTP applications. The most common application that uses these hooks is Internet Explorer. Port mappings are not relevant for this level.

  - **Socket level and WinINet level data.** Captures data using both mechanisms. WinINet level sends information for applications that use the WinINet DLL. Socket level sends data only if it determines that it did not originate from the WinINet dll. Port mapping applies to data that did not originate from WinINet.

- **Network-level server address mappings for.** Specifies the mappings per protocol. For example, to show only the FTP mappings, select FTP.
- **New Entry.** Opens the Server Entry dialog box, allowing you to add a new mapping. See "Adding a New Server Entry" on page 401.
- **Edit Entry.** Opens the Server Entry dialog box, allowing you to edit the selected entry.
- **Delete Entry.** Deletes the selected entry.
- **Options.** Opens the Advanced Settings dialog box to enable auto-detection of the communication protocol and SSL level. See "Setting the Advanced Port Mapping Options" on page 403.

If you do not specify all of the port and server names, VuGen uses the following priorities in assigning data to a service:

Priority	Port	Server
1	specified	specified
2	not specified <All>	specified
3	specified	not specified <All>
4	not specified <All>	not specified <All>

A map entry with a high priority does not get overridden by an entry with a lower priority. For example, if you specify that traffic on server *twilight* using port 25 be handled as SMTP and then you specify that all servers on port 25 be handled as HTTP, the data will be treated as SMTP.

In addition, the following guidelines apply:

- **Port 0.** Port number 0 indicates any port.

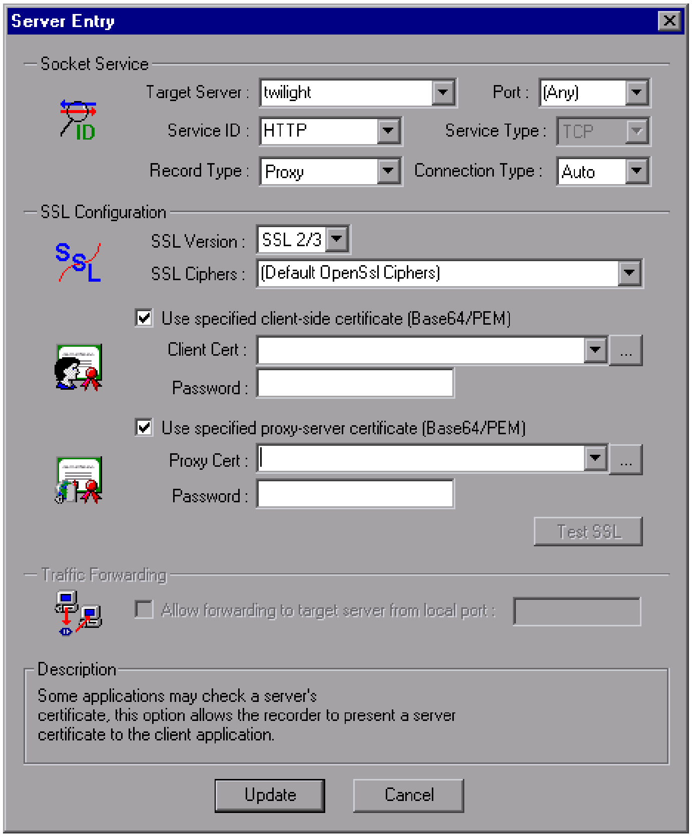
- **Forced mapping.** If you specify a mapping for a port number, server name, or combination server:port, VuGen forces the network traffic to use that service. For example, if you were to specify <Any> server on port 80 to use FTP, VuGen uses the FTP protocol to record that communication, even though the actual communication may be HTTP. In this instance, the Vuser script might be empty.

After you define a port mapping, it appears in the list of Port Mappings. You can temporarily disable any entry by clearing the check box adjacent to it. When you disable an entry, VuGen ignores all traffic to that server:port combination. You should disable the port entry when the data is irrelevant or if the protocol is not supported.

For further instructions, see "Setting the Port Mapping Recording Options" on page 406.

Adding a New Server Entry

You use the Server Entry dialog box to create a new entry in the list of port mappings.



Server Entry

Socket Service

Target Server : twilight

Port : (Any)

Service ID : HTTP

Service Type : TCP

Record Type : Proxy

Connection Type : Auto

SSL Configuration

SSL Version : SSL 2/3

SSL Ciphers : (Default OpenSsl Ciphers)

☑ Use specified client-side certificate (Base64/PEM)

Client Cert :

Password :

☑ Use specified proxy-server certificate (Base64/PEM)

Proxy Cert :

Password :

Traffic Forwarding

☐ Allow forwarding to target server from local port :

Description

Some applications may check a server's certificate, this option allows the recorder to present a server certificate to the client application.

Socket Service

- **Target Server.** The IP address or host name of the target server for which this entry applies. The default is All Servers.
- **Port.** The port of the target server for which this entry applies. Port 0 implies all ports.

- **Service ID.** A protocol or service name used by the recorder to identify the type of connection (i.e. HTTP, FTP, and so on). You can also specify a new name. The name may not exceed 8 characters.
- **Service Type.** The type of service, currently set to TCP.
- **Record Type.** The type of recording—directly or through a proxy server.
- **Connection Type.** The security level of the connection: Plain (non-secure), SSL, or Auto. If you select Auto, the recorder checks the first 4 bytes for an SSL signature. If it detects the SSL signature, it assumes that SSL is being used.

Note: SSL connections do not apply to the following Mailing Services and E-Business protocols: FTP, LDAP, SMTP, POP3, IMAP, DNS, and MAPI.

SSL Configuration

If you selected **SSL** or **auto** as the connection type, configure the relevant SSL settings in the section. These settings only apply to the new entry. You should only specify them if you have explicit information about your application's SSL encoding. Otherwise, accept the defaults.

- **SSL Version.** The preferred SSL version to use when communicating with the client application and the server. By default is SSL 2/3 is used. However some services require SSL 3.0 only or SSL 2.0 only. Some new wireless applications require TLS 1.0—a different security algorithm.
- **SSL Cipher.** The preferred SSL cipher to use when connecting with a remote secure server.
- **Use specified client-side certificate.** The default client-side certificate to use when connecting to a remote server. Specify or browse for a certificate file in *txt*, *crt*, or *pem* format, and supply a password.
- **Use specified proxy-server certificate.** The default server certificate to present to client applications that request a server certificate. Specify or browse for a certificate file in *txt*, *crt*, or *pem* format, and supply a password. Click **Test SSL** to check the authentication information against the server.

Traffic Forwarding

- **Allow forwarding to target server from local port.** This option forwards all traffic from a specific port to another server. This is particularly useful in cases where VuGen cannot run properly on the client, such as unique UNIX machines, or instances where it is impossible to launch the application server through VuGen. We configure VuGen to intercept the traffic from the problematic client machine, and pass it on to the server. In this way, VuGen can process the data and generate code for the actions.

For example, if you were working on a UNIX client called *host1*, which communicated with a server, *server1*, over port 8080, you would create a Port Mapping entry for *server1*, port 8080. In the **Traffic Forwarding** section of the Server Entry dialog box, you enable traffic forwarding by selecting the **Allow forwarding to target server from local port** check box. You specify the port from which you want to forward the traffic, in our example 8080.

You then connect the client, *host1*, to the machine running VuGen, instead of *server1*. VuGen receives the communication from the client machine and forwards it via the local port 8080, to the server. Since the traffic passes through VuGen, it can analyze it and generate the appropriate code.

For further instructions, see "Setting the Port Mapping Recording Options" on page 406.

Setting the Advanced Port Mapping Options

VuGen's advanced port-mapping options let you configure the **auto-detection** options. VuGen's auto-detection analyzes the data that is sent to the server. It checks the data for a signature, a pattern in the data's content, that identifies the protocol. For the purpose of detecting a signature, all of the send buffers until the first receive buffer, are combined. All send buffers that were sent until a receive buffer is returned, are considered a single data **transition**. By default, no mappings are defined and VuGen employs auto-detection. In some protocols, VuGen determines the type in a single transition, (such as HTTP). Other network protocols require several transitions before determining the type. For this purpose, VuGen creates a temporary buffer, per server-port combination. If VuGen cannot determine the protocol type by reading the first transition buffers, it stores the data in a temporary buffer. It continues to read the incoming buffers until it detects a signature of a specific protocol.

By default, VuGen allows 4 transitions and uses a temporary buffer of 2048 bytes in order to detect a protocol signature. If VuGen has not yet determined the type after reaching the maximum number of transitions, or after reaching the maximum buffer size, it assigns the data to the WinSock protocol. If you did not instruct VuGen to record the WinSock protocol (in the multi-protocol selection), VuGen discards the data.

You can change the maximum number of buffers you want VuGen to read in order to detect the protocol type. You can also specify the size of the temporary buffer. In instances where the amount of data in the first send buffers, is greater than the size of the temporary buffer, VuGen cannot auto-detect the protocol type. In this case, you should increase the size of the temporary buffer.

- **Enable auto SSL detection.** Automatically detects SSL communication. Specify the version and default cipher that you want to detect. Note that this only applies to port mappings that were defined as *auto* in the **Connection type** box, or not defined at all. If a server, port, or server:port combination was defined as either Plain or SSL, then auto SSL detection does not apply.
- **Enable auto detection of SOCKET based communication.** Automatically detects the type of communication. If required, raise the maximum number of transitions, one at a time until VuGen succeeds in detecting the protocol. You can also gradually increase the maximum buffer size by 1024 bytes (1 KB) at a time until VuGen succeeds in detecting the protocol. This allows VuGen to review a larger amount of data in order to find a signature.
- **Log Level.** Sets the logging level for the automatic socket detection: None, Standard (Default), Debug, or Advanced Debug.

When working with the above network level protocols, we recommend that you allow VuGen to use auto-detection to determine the protocol type. In most cases, VuGen's recorder is able to recognize the signatures of these protocols. It then automatically processes them according to the protocol specifications. In certain instances, however, VuGen may be unable to recognize the protocol. For example:

- The protocol signature closely resembles an existing protocol, resulting in erroneous processing.
- There is no unique signature for the protocol.
- The protocol uses SSL encryption, and therefore cannot be recognized on a WinSock level.

In all of the above cases, you can supply information to uniquely identify the server and port hosting the protocol.

For further instructions, see "Setting the Port Mapping Recording Options" on page 406.

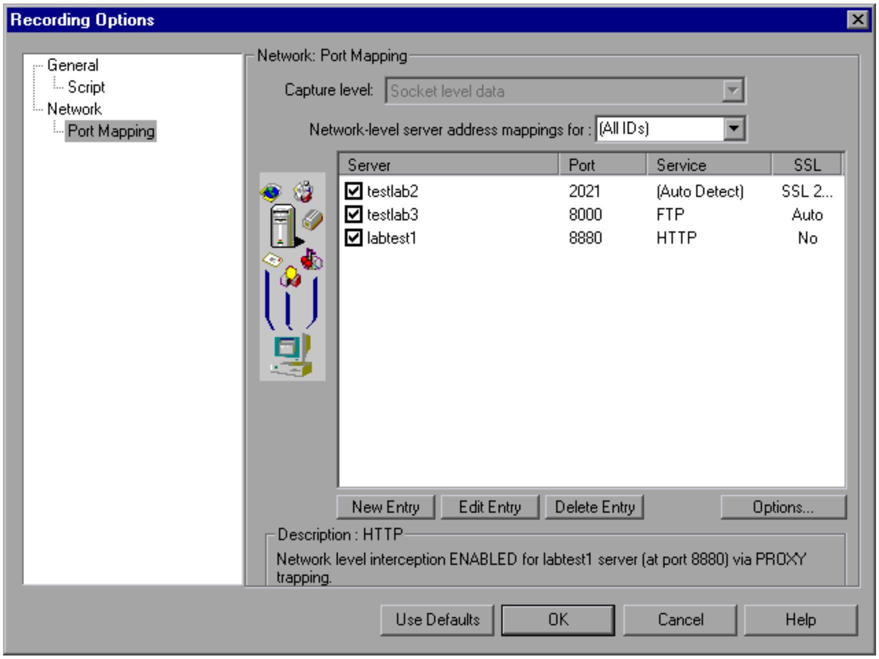
Setting the Port Mapping Recording Options

Note that you can open the Recording Options dialog box in several ways:

- The toolbar button:
- The keyboard shortcut: Ctrl+F7
- The Tools menu: select **Tools** > **Recording Options**

To set the port mapping recording options:

1 Open the Recording Options and select the **Network:Port Mapping** node.



Network: Port Mapping

Capture level: Socket level data

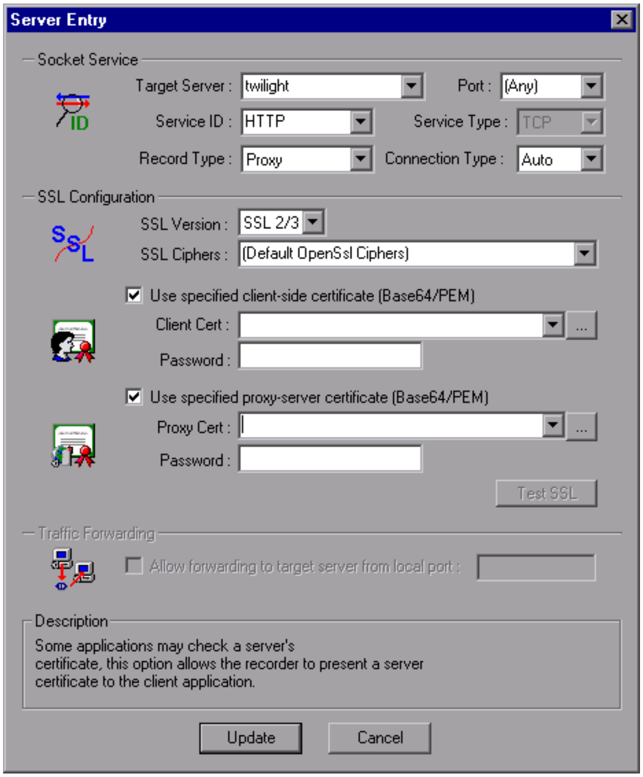
Network-level server address mappings for : (All IDs)

Server	Port	Service	SSL
☑ testlab2	2021	(Auto Detect)	SSL 2...
☑ testlab3	8000	FTP	Auto
☑ labtest1	8880	HTTP	No

Description : HTTP

Network level interception ENABLED for labtest1 server (at port 8880) via PROXY trapping.

2 To create a new server:port mapping, click **New Entry**. The Server Entry dialog box opens.



Server Entry

Socket Service

Target Server : twilight

Port : (Any)

Service ID : HTTP

Service Type : TCP

Record Type : Proxy

Connection Type : Auto

SSL Configuration

SSL Version : SSL 2/3

SSL Ciphers : (Default OpenSsl Ciphers)

☑ Use specified client-side certificate (Base64/PEM)

Client Cert :

Password :

☑ Use specified proxy-server certificate (Base64/PEM)

Proxy Cert :

Password :

Traffic Forwarding

□ Allow forwarding to target server from local port :

Description

Some applications may check a server's certificate, this option allows the recorder to present a server certificate to the client application.

3 Enter the **Service ID, Service Type, Target Server, Target Port,** and **Connection Type** in the Socket Service section:

4 If you selected **SSL** or **auto** as the connection type, configure the relevant SSL settings in the **SSL Configuration** section. These settings only apply to the new entry. You should only specify them if you have explicit information about your application's SSL encoding. Otherwise, accept the defaults.

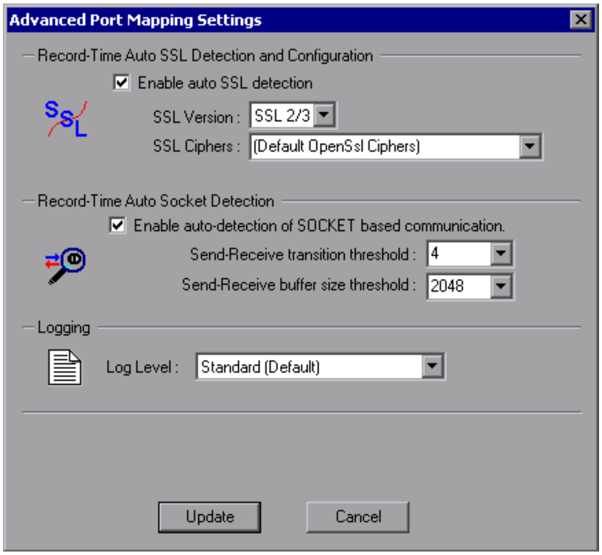
Specify the **SSL Version**, **SSL Cipher**. To use a certificate, select **Use specified client-side certificate** or **Use specified proxy-server certificate** and specify the user information.

Click **Test SSL** to check the authentication information against the server.

5 To allow traffic forwarding, select **Allow forwarding to target server from local port**, and specify a port number. Note that this option is only enabled when the **Target Server** and **Target Port** are unique (not <Any>).

6 Click **Update** to save the mapping and close the Server Entry dialog box.

7 To set automatic detection capabilities, click **Options**. The Advanced Port Mapping Setting dialog box opens.



Advanced Port Mapping Settings

Record-Time Auto SSL Detection and Configuration

☑ Enable auto SSL detection

SSL Version : SSL 2/3

SSL Ciphers : (Default OpenSsl Ciphers)

Record-Time Auto Socket Detection

☑ Enable auto-detection of SOCKET based communication.

Send-Receive transition threshold : 4

Send-Receive buffer size threshold : 2048

Logging

Log Level : Standard (Default)

To automatically detect SSL communication, select **Enable auto SSL detection** and specify the version and cipher information.

To automatically detect the type of communication, select **Enable auto detection of SOCKET based communication**. If required, raise the maximum number of transitions.

Select a **Log Level:** None, Standard, Debug, or Advanced Debug.

Click **Update** to accept the auto-detection options and close the dialog box.

8 To view all of the entries, select **All IDs** in the **Network-level server address mappings** box.

9. To modify an existing entry, select it and click **Edit Entry**. Note that you cannot change the server name or port number of an entry. You can only change the connection type and security settings.
10. To permanently delete a mapping, select the entry from the list and click **Delete Entry**. To temporarily disable the mapping settings for a specific entry, clear the check box adjacent to that item. To enable the mapping, select the check box.
11. Click **OK**.

Part 4

Run-Time Settings

22

Configuring Run-Time Settings

After you record a Vuser script, you configure the run-time settings for the script. These settings specify how the script behaves when it runs.

This chapter includes:

- About Run-Time Settings on page 414
- Configuring Run Logic Run-Time Settings (multi-action) on page 415
- Pacing Run-Time Settings on page 420
- Configuring Pacing Run-Time Settings (multi-action) on page 421
- Setting Pacing and Run Logic Options (single action) on page 422
- Configuring the Log Run-Time Settings on page 424
- Configuring the Think Time Settings on page 429
- Configuring Additional Attributes Run-Time Settings on page 431
- Configuring Miscellaneous Run-Time Settings on page 432
- Setting the VB Run-Time Settings on page 438

About Run-Time Settings

After you record a Vuser script, you can configure its run-time settings. The run-time settings define the way that the script runs. These settings are stored in the file *default.cfg,* located in the Vuser script directory. Run-time settings are applied to Vusers when you run a script using VuGen, the Controller, or Business Process Monitor.

Configuring run-time settings allows you to emulate different kinds of user activity. For example, you could emulate a user who responds immediately to output from the server, or a user who stops and thinks before each response. You can also configure the run-time settings to specify how many times the Vuser should repeat its set of actions.

You use the Run-Time Settings dialog box to display and configure the run-time settings tree. You can open these settings in one of the following ways:

- Click the **Run-Time Settings** button on the VuGen toolbar.
- Press the keyboard shortcut key **F4**.
- Select **Vuser** > **Run-Time Settings**.

You can also modify the run-time settings from the LoadRunner Controller. For more information, see the product's documentation.

Note: For LoadRunner, the default run-time setting support the debugging environment of VuGen and the load testing environment of the Controller. The default settings are:

- **Think Time.** Off in VuGen and Replay as Recorded in the Controller.
- **Log.** Standard in VuGen and off in the Controller.
- **Download non-HTML resources.** Enabled in VuGen and the Controller.

The General run-time settings described in this chapter, apply to all types of Vuser scripts. They include:

- Run Logic (Iterations)
  - Pacing
  - Log
  - Think Time
  - Miscellaneous
  - Additional Attributes

For protocols that do NOT support multiple actions, such as WinSocket and Database (Oracle 2-tier, Sybase, MSSQL, and so on), the Iteration and Pacing options are both handled from the Pacing tab. Many protocols have additional run-time settings. For information about the specific run-time settings for these protocols, see the appropriate sections.

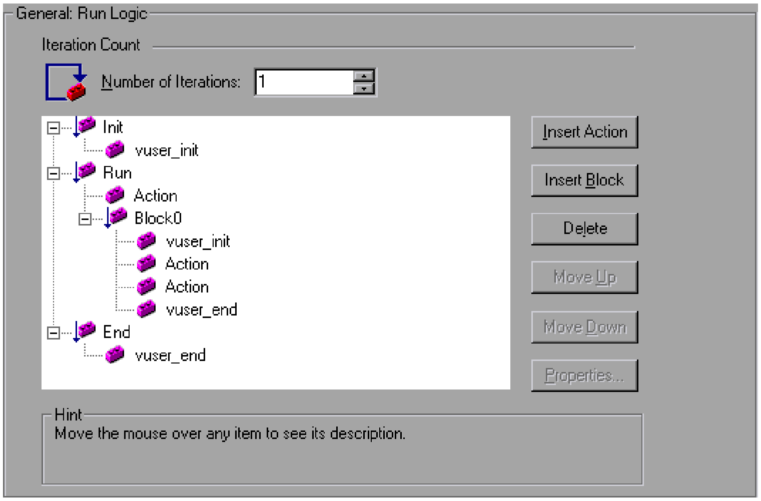
Configuring Run Logic Run-Time Settings (multi-action)

Note: The following section only applies to protocols that work with multiple actions. If the **Run Logic** node exists under the run-time settings, it is a multiple action protocol. For single action protocols, see "Setting Pacing and Run Logic Options (single action)" on page 422.

Every Vuser script contains three sections: *vuser_init*, *Run (Actions)*, and *vuser_end*. You can instruct a Vuser to repeat the *Run* section when you run the script. Each repetition is known as an *iteration*.

The *vuser_init* and *vuser_end* sections of a Vuser script are not repeated when you run multiple iterations.

Open the Run-Time Settings and select the **General:Run Logic** node.



General: Run Logic

Iteration Count

Number of Iterations: 1

```
Init
  vuser_init
Run
  Action
  Block0
    vuser_init
    Action
    Action
    vuser_end
End
  vuser_end
```

Hint

Move the mouse over any item to see its description.

- **Number of Iterations.** The number of iterations. The Vusers repeat all of the Actions the specified number of times.

Note: For the LoadRunner Controller: If you specify a scenario duration in the Scheduling settings, they override the Vuser iteration settings. This means that if the duration is set to five minutes (the default setting), the Vusers will continue to run as many iterations as required for five minutes, even if the run-time settings specify only one iteration.

When you run scripts with multiple actions, you can indicate how to execute the actions, and how the Vuser executes them:

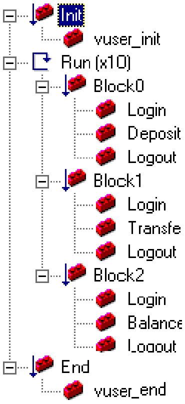
- **Action Blocks.** Action blocks are groups of actions within your script. You can set the properties of each block independently—its sequence, iterations, and weighting.
- **Sequence.** You can set the order of actions within your script. You can also indicate whether to perform actions sequentially or randomly.

- **Iterations.** In addition to setting the number of iterations for the entire *Run* section, you can set iterations for individual actions or action blocks. This is useful, for example, in emulating a commercial site where you perform many queries to locate a product, but only one purchase.
- **Weighting.** For action blocks running their actions randomly, you can set the *weight* or percentage of each action within a block.

Creating Action Blocks

Action blocks are groups of actions within the Vuser script. You can create separate action blocks for groups of actions, adding the same action to several blocks. You can instruct VuGen to execute action blocks or individual actions sequentially or randomly. In the default sequential mode, the Vuser executes the blocks or actions in the order in which they appear in the iteration tree view.

In the following example, *Block0* performs a deposit, *Block1* performs a transfer, and *Block2* submits a balance request. The *Login* and *Logout* actions are common to the three blocks.



```
Init
    vuser_init
Run (x10)
    Block0
        Login
        Deposit
        Logout
    Block1
        Login
        Transfe
        Logout
    Block2
        Login
        Balance
        Logout
End
    vuser_end
```

You configure each block independently—its sequence and iterations.

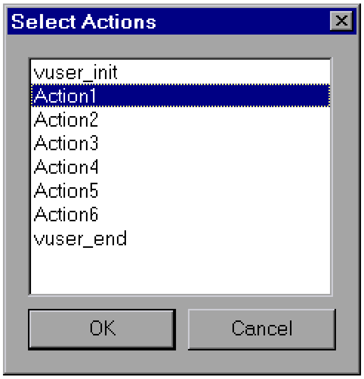
To configure actions and action blocks:

1 Create all of the desired actions through recording or programming.

2 Open the Run-Time setting. Select the **General:Run Logic** node.

3 Add a new action block. Click **Insert Block**. VuGen inserts a new Action block at the insertion point with the next available index (*Block0*, *Block1*, *Block2*).

4 Add actions to the block. Click **Insert Action**. The Select Actions list opens.



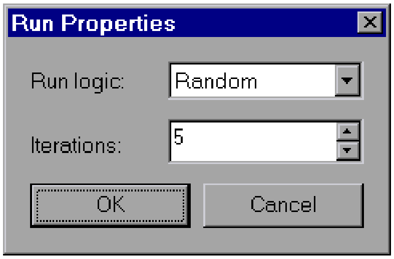
5 Select an action to add to the block and click **OK**. VuGen inserts a new action into the current block or section.

6 Repeat step 3 for each action you want to add to the block.

7 To remove an action or an action block, select it and click **Delete**.

8 Click **Move Up** or **Move Down** to modify an item's position.

9 Click **Properties** to set the number of iterations and run logic of the actions. The Run Properties dialog opens.



Run logic: Random

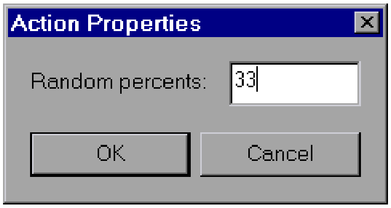
Iterations: 5

10 Select *Sequential* or *Random* from the **Run Logic** list, indicating to VuGen whether to run the actions sequentially or randomly.

11 Specify the number of iterations in the **Iterations** box. Note that if you define parameters within the action block, and you instruct VuGen to update their values each iteration, it refers to the global iteration—not the individual block iteration.

12 Click **OK**.

13 For blocks with Random run logic, set the weighting of each action. Right-click an action and select **Properties**. The Action Properties dialog opens.



Random percents: 33

Specify the desired percent for the selected block or action. In the **Random Percents** box, specify a percentage for the current action. The sum of all percentages must equal 100.

14 Repeat the above steps for each element whose properties you want to set.

Pacing Run-Time Settings

Note: The following section only applies to protocols that work with multiple actions. If the **Run Logic** node exists under the run-time settings, it is a multiple action protocol. For single action protocols, see "Setting Pacing and Run Logic Options (single action)" on page 422.

The Pacing Run-Time settings let you control the time between iterations. The pace tells the Vuser how long to wait between iterations of your actions. You instruct the Vusers to start each iteration using one of the following methods:

- **As soon as the previous iteration ends.** The new iteration begins as soon as possible after the previous iteration ends.
- **After the previous iteration ends with a fixed or random delay of ...** Starts each new iteration a specified amount of time after the end of the previous iteration. Specify either an exact number of seconds or a range of time. For example, you can specify to begin a new iteration at any time between 60 and 90 seconds after the previous iteration ends.

  When you run the script, VuGen shows the time the Vuser waited between the end of one iteration and the start of the next one, in the Execution Log.
- **At fixed or random intervals, every ... [to ...] seconds.** You specify the time between iteration—either a fixed number of seconds or a range of seconds from the beginning of the previous iteration. For example, you can specify to begin a new iteration every 30 seconds, or at a random rate ranging from 30 to 45 seconds from the beginning of the previous iteration. Each scheduled iterations will only begin when the previous iteration is complete.

Each scheduled iteration will only begin when the previous iteration is complete. When you run the script, VuGen shows the time the Vuser waited between the end of one iteration and the start of the next one, in the Execution Log.

For example, assume that you specify to start a new iteration every four seconds:

- If the first iteration takes three seconds, the Vuser waits one second.
- If the first iteration takes two seconds to complete, the Vuser waits two seconds.
- If the first iteration takes 8 seconds to complete, the second iteration will start 8 seconds after the first iteration began. VuGen displays a message in the Execution Log to indicate that the iteration pacing could not be achieved.

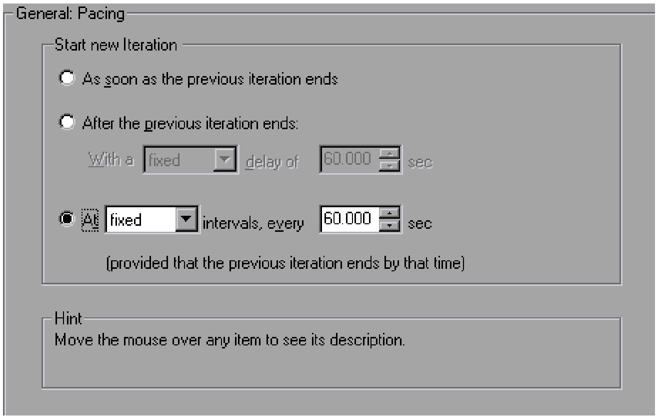
For further instructions about setting the Pacing options, see "Configuring Pacing Run-Time Settings (multi-action)" on page 421.

Configuring Pacing Run-Time Settings (multi-action)

You use the Pacing options to pace your actions by setting the time intervals between iterations.

To set the pacing between iterations:

1 Open the Run-Time Settings and select the **General:Pacing** node.



General: Pacing

Start new Iteration

○ As soon as the previous iteration ends

○ After the previous iteration ends:

With a fixed delay of 60.000 sec

● At fixed intervals, every 60.000 sec

(provided that the previous iteration ends by that time)

Hint

Move the mouse over any item to see its description.

2 In the **Start New Iteration** section, select one of the following options:

- As soon as the previous iteration ends
- After the previous iteration ends
- At fixed or random intervals

3 For the **After the previous iteration ends** option:

- Select a delay type: **fixed** or **random**.
- Specify a value for fixed, or a range of values for the random delay.

4 For the **At … intervals** option:

- Select a interval type: **fixed** or **random**.
- Specify a value for fixed, or a range of values for the random interval.

5 Click **OK**.

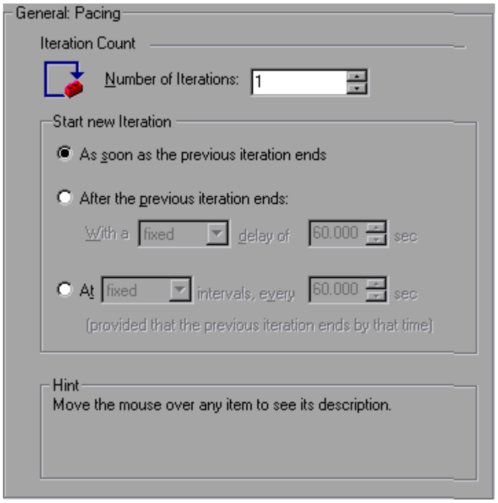
Setting Pacing and Run Logic Options (single action)

Note: The following section only applies to protocols that work with single actions—not multiple actions. If there is a **Pacing** node and not a **Run Logic** node under the General run-time settings, it is a single action protocol.

You can instruct a Vuser to repeat the *Action* section when you run the script. Each repetition is known as an *iteration*. The *vuser_init* and *vuser_end* sections of a Vuser script are not repeated when you run multiple iterations.

To set the iteration and pacing preferences:

1 Click the **Run-Time Settings** button on the VuGen toolbar or select **Vuser** > **Run-Time Settings**. Click the **Pacing** node to display the iteration and pacing options.



General: Pacing

Iteration Count

Number of Iterations: 1

Start new Iteration

● As soon as the previous iteration ends

○ After the previous iteration ends:

With a fixed delay of 60.000 sec

○ At fixed intervals, every 60.000 sec

(provided that the previous iteration ends by that time)

Hint

Move the mouse over any item to see its description.

2 Specify the number of iterations in the **Iteration Count** box. The Vuser repeats all of the Actions the specified number of times.

3 In the **Start New Iteration** section, select one of the following options:

- As soon as the previous iteration ends
- After the previous iteration ends
- At fixed or random intervals

4 For the **After the previous iteration ends** option:

- Select a delay type: **fixed** or **random**.
- Specify a value for fixed, or a range of values for the random delay.

5 For the **At ... intervals** option:

- Select a interval type: **fixed** or **random**.
- Specify a value for fixed, or a range of values for the random interval.

6 Click **OK**.

For an overview of the pacing options, see "Pacing Run-Time Settings" on page 420.

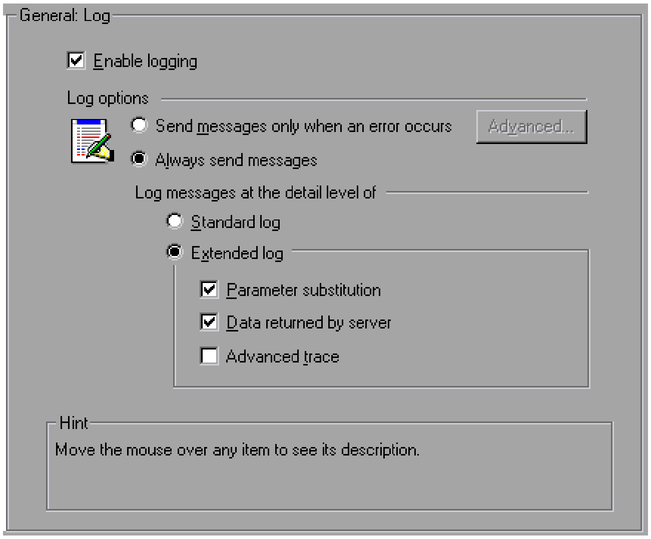
Configuring the Log Run-Time Settings

During execution, Vusers log information about themselves and their communication with the server. In a Windows environment, this information is stored in a file called *output.txt* in the script directory. In UNIX environments, the information is directed to the standard output. The log information is useful for debugging purposes.

The Log run-time settings let you determine how much information is logged to the output. You can select **Standard** or **Extended** log, or you can disable logging completely. Disabling the log is useful when working with many Vusers. If you have tens or hundreds of Vusers logging their run-time information to disk, the system may work slower than normal. During development, enable logging so that you will have information about the replay. You should only disable logging after verifying that the script is functional.

Note: You can program a Vuser script to send messages to an output log by using the **lr_error_message** and **lr_output_message** functions.

Click the **Run-Time Settings** button on select **Vuser** > **Run-Time Settings** to display the Run-Time Settings dialog box. Select the **General:Log** node to display the log options.



General: Log

☑ Enable logging

Log options

○ Send messages only when an error occurs

● Always send messages

Log messages at the detail level of

○ Standard log

● Extended log

☑ Parameter substitution

☑ Data returned by server

☐ Advanced trace

Hint

Move the mouse over any item to see its description.

Enable Logging

This option enables automatic logging during replay—VuGen writes log messages that you can view in the Execution log. This option only affects automatic logging and log messages issued through **lr_log_message**. Messages sent manually, using **lr_message**, **lr_output_message**, and **lr_error_message**, are still issued.

Log Options

The Log run-time settings allows you to adjust the logging level depending on your development stage.

You can indicate when to send log messages to the log: **Send messages only when an error occurs** or **Always send messages**. During development, you can enable all logging. Once you debug your script and verify that it is functional, you can enable logging for errors only.

If you choose to send messages only when errors occur, also known as JIT, (Just in Time) messaging, you can set an advanced option, indicating the size of the log cache. See "Setting the Log Cache Size" on page 427.

Setting the Log Detail Level

You can specify the type of information that is logged, or you can disable logging altogether.

Note: If you set **Error Handling** to "Continue on error" in the **General Run-Time Settings** folder, error messages are still sent to the Output window.

If you modify the script's Log Detail Level, the behavior of the **lr_message**, **lr_output_message**, and **lr_log_message** functions will not change—they will continue to send messages.

- **Standard Log**. Creates a standard log of functions and messages sent during script execution to use for debugging. Disable this option for large load testing scenarios or profiles.

  If the logging level is set to **Standard**, the logging mode is automatically set to **JIT logging** when adding it to a scenario or profile. If, however, the logging mode was disabled or set to **Extended**, then adding the script to a scenario or profile will not affect its logging settings.

- **Extended Log.** Creates an extended log, including warnings and other messages. Disable this option for large load testing scenarios or profiles.

You can specify which additional information should be added to the extended log using the Extended log options:

- **Parameter substitution.** Select this option to log all parameters assigned to the script along with their values. For more information on parameters, see Chapter 13, "Working with VuGen Parameters."
- **Data returned by server.** Select this option to log all of the data returned by the server.

- **Advanced trace.** Select this option to log all of the functions and messages sent by the Vuser during the session. This option is useful when you debug a Vuser script.

The degree to which VuGen logs events (Standard, Parameter substitution, and so forth) is also known as the *message class*. There are five message classes: Brief, Extended, Parameters, Result Data, and Full Trace.

You can manually set the message class within your script using the **lr_set_debug_message** function. This is useful if you to want to receive debug information about a small section of the script only.

For example, suppose you set Log run-time settings to Standard log and you want to get an Extended log for a specific section of the script. You would then use the **lr_set_debug_message** function to set the Extended message class at the desired point in your script. You must call the function again to specify what type of extended mode (Parameter, Result Data, or Full Trace). Return to the Standard log mode by calling **lr_set_debug_message**, specifying Brief mode. For more information about setting the message class, see the *Online Function Reference* (**Help** > **Function Reference**).

Setting the Log Cache Size

The Advanced options for the Log Run-Time settings, let you indicate the size of the log cache. The log cache stores raw data about the test execution, to make it available should an error occur. When the contents of the cache exceed the specified size, it deletes the oldest items. The default size is 1KB.

The following is the sequence of the logging:

1. You indicate to VuGen to log messages only when an error occurs, by selecting **Send messages only when an error occurs**.
2. VuGen stores information about the test execution in the log cache without writing it to a file. If this information exceeds 1 KB, it overwrites the oldest data. The Execution Log tab also remains empty, since it is a dump of the log file’s contents.
3. When an error occurs (either an internal error or a programmed error using **lr_error_message**), VuGen places the contents of the cache into the log file and Execution Log tab. This allows you to see the events that led up to the error.

When an error occurs and VuGen dumps its stored cache into the log file, the actual file size will be greater than the cache size. For example, if your cache size is 1KB, the log file size may be 50 KB. This is normal and only reflects the overhead required for formatting the raw data into meaningful sentences.

Note that in JIT mode, the output of **lr_message** and **lr_log_message**, are only sent to the Output window or log file, if their output was in the log cache at the time of the error. Check the Execution Log for the specified message strings.

Logging CtLib Server Messages

When you run a CtLib Vuser script, (Sybase CtLib, under the Client Server type protocols), all messages generated by the CtLib client are logged in the standard log and in the output file. By default, server messages are not logged. To enable logging of server messages (for debugging purposes), insert the following line into your Vuser script:

```
LRD_CTLIB_DB_SERVER_MSG_LOG;
```

VuGen logs all server messages in the Standard log.

To send the server messages to the output (in addition to the Standard log), type:

```
LRD_CTLIB_DB_SERVER_MSG_ERR;
```

To return to the default mode of not logging server errors, type the following line into your script:

```
LRD_CTLIB_DB_SERVER_MSG_NONE;
```

Note: Activate server message logging for only a specific block of code within your script, since the generated server messages are long and the logging can slow down your system.

Configuring the Think Time Settings

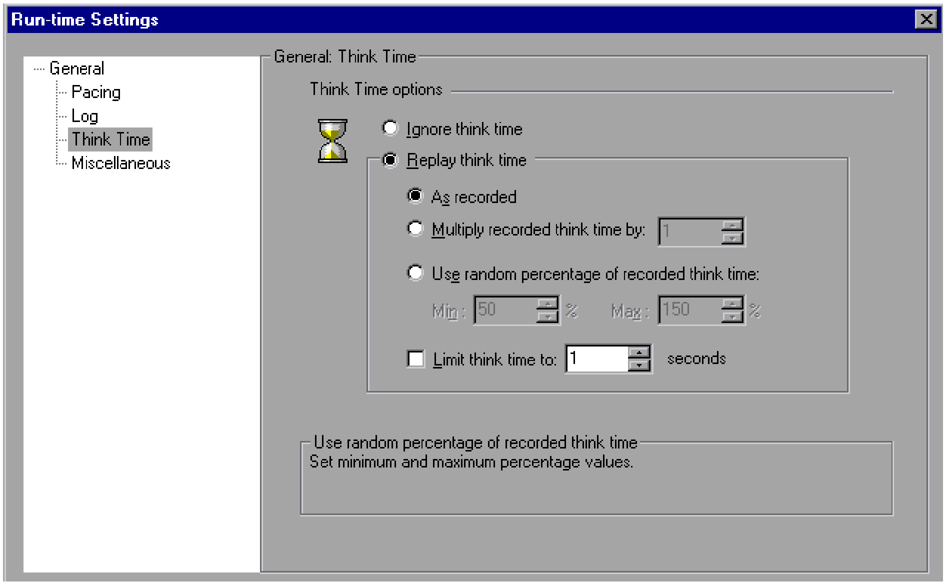
Vuser *think time* emulates the time that a real user waits between actions. For example, when a user receives data from a server, the user may wait several seconds to review the data before responding. This delay is known as the *think time*. VuGen uses **lr_think_time** functions to record think time values into your Vuser scripts. The following recorded function indicates that the user waited 8 seconds before performing the next action:

```
lr_think_time(8);
```

When you run the Vuser script and the Vuser encounters the above **lr_think_time** statement, by default, the Vuser waits 8 seconds before performing the next action. You can use the Think Time run-time settings to influence how the Vuser uses the recorded think time when you run the script.

For more information about the **lr_think_time** function and how to modify it manually, see the *Online Function Reference* (**Help** > **Function Reference**).

Click the **Run-Time Settings** button on the VuGen toolbar or select **Vuser** > **Run-Time Settings**. Select the **General:Think Time** node to display the Think Time options:



General: Think Time

Think Time options

○ Ignore think time

● Replay think time

● As recorded

○ Multiply recorded think time by: 1

○ Use random percentage of recorded think time:

Min : 50 % Max : 150 %

☐ Limit think time to: 1 seconds

Use random percentage of recorded think time

Set minimum and maximum percentage values.

Think Time Options

By default, when you run a Vuser script, the Vuser uses the think time values that were recorded into the script during the recording session. VuGen allows you to use the recorded think time, ignore it, or use a value related to the recorded time:

- **Ignore think time.** Ignore the recorded think time—replay the script ignoring all **lr_think_time** functions.
- **Replay the think time.** The second set of think times options let you use the recorded think time:
  - **As recorded.** During replay, use the argument that appears in the lr_think_time function. For example, lr_think_time(10) waits ten seconds.

- **Multiply recorded think time by.** During replay, use a multiple of the recorded think time. This can increase or decrease the think time applied during playback. For example, if a think time of four seconds was recorded, you can instruct your Vuser to multiply that value by two, for a total of eight seconds. To reduce the think time to two seconds, multiply the recorded time by 0.5.
- **Use random percentage of the recorded think time.** Use a random percentage of the recorded think time. You set a range for the think time value by specifying a range for the think time. For example, if the think time argument is 4, and you specify a minimum of 50% and a maximum of 150%, the lowest think time can be two (50%) and the highest value six (150%).
- **Limit think time to.** Limit the think time's maximum value.

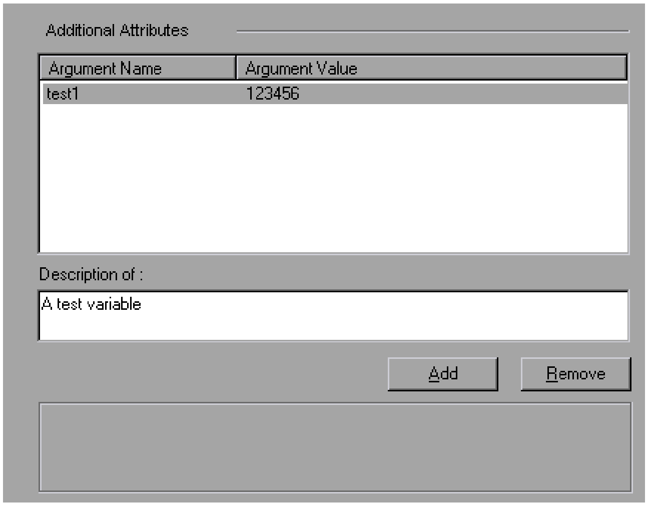
Configuring Additional Attributes Run-Time Settings

You can use the Additional Attributes node to provide additional arguments for a Vuser script. The Additional Attributes settings apply to all Vuser script types.

You specify command line arguments that you can retrieve at a later point during the test run, using **lr_get_attrib_string**. Using this node, you can pass external parameters to prepared scripts.

To set additional attributes:

1 Click the Run-Time Settings button or select **Vuser** > **Run-Time Settings** to display the Run-Time Settings dialog box. Select the **General:Additional Attributes** node from the tree in the left pane.



Additional Attributes

Argument Name	Argument Value
test1	123456

Description of :

A test variable

2 Click **Add** to add a new command line argument entry. Enter the argument name and its value.

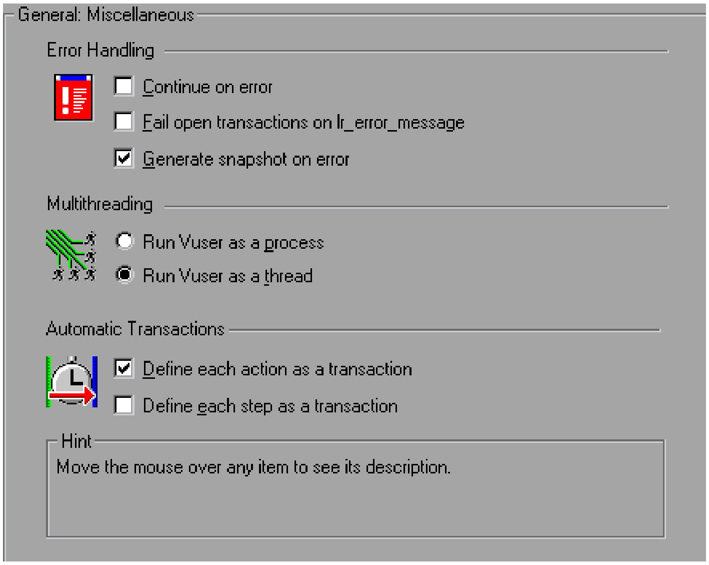
3 Click **Remove** to remove the selected argument.

Configuring Miscellaneous Run-Time Settings

You can set the following Miscellaneous run-time options for a Vuser script: Note that the Multithreading and Automatic Transaction options are not applicable to HP Business Availability Center.

- Error Handling
- Multithreading
- Automatic Transactions

Click the **Run-Time Settings button** or select **Vuser** > **Run-Time Settings** to display the Run-Time Settings dialog box. Select the **General:Miscellaneous** node from the tree in the left pane.



General: Miscellaneous

Error Handling

□ Continue on error

□ Fail open transactions on lr_error_message

☑ Generate snapshot on error

Multithreading

○ Run Vuser as a process

● Run Vuser as a thread

Automatic Transactions

☑ Define each action as a transaction

□ Define each step as a transaction

Hint

Move the mouse over any item to see its description.

The *Miscellaneous* settings apply to all Vuser script types.

Error Handling

- **Continue on Error.** This setting instructs Vusers to continue script execution when an error occurs. This option is turned off by default, indicating that the Vuser will exit if an error occurs.
- **Fail open transactions on lr_error_message.** This option instructs VuGen to mark all transactions in which an **lr_error_message** function was issued, as *Failed*. The **lr_error_message** function is issued through a programmed *If* statement, when a certain condition is met.
- **Generate Snapshot on Error.** This option generates a snapshot when an error occurs. You can see the snapshot by viewing the Vuser Log and double-clicking on the line at which the error occurred.

It is not recommended to enable both the **Continue on Error** and **Generate Snapshot on Error** options in a load test environment. This configuration may adversely affect the Vusers' performance.

Error Handling for Database Vusers

When working with database protocols (LRD), you can control error handling for a specific segment of a script. To mark a segment, enclose it with LRD_ON_ERROR_CONTINUE and LRD_ON_ERROR_EXIT statements. The Vuser applies the new error setting to the whole segment. If you specify Continue on Error, VuGen issues a messages indicating that it encountered an error and is ignoring it.

For example, if you enable the Continue on Error feature and the Vuser encounters an error during replay of the following script segment, it continues executing the script.

```
lrd_stmt(Csr1, "select…"…);
lrd_exec(…);
```

To instruct the Vuser to continue on error for the entire script except for a specific segment, select the Continue on Error option and enclose the segment with LRD_ON_ERROR_EXIT and LRD_ON_ERROR_CONTINUE statements:

```
LRD_ON_ERROR_EXIT;
lrd_stmt(Csr1, "select…"…);
lrd_exec(…);
LRD_ON_ERROR_CONTINUE;
```

In addition to the LRD_ON_ERROR statements, you can control error handling using *severity levels*. LRD_ON_ERROR statements detect all types of errors—database related, invalid parameters, and so on. If you want the Vuser to terminate only when a database operation error occurs (Error Code 2009), you can set a function's severity level. All functions that perform a database operation use severity levels, indicated by the function's final parameter, *miDBErrorSeverity*.

VuGen supports the following severity levels:

Definition	**Meaning**	**Value**
LRD_DB_ERROR_SEVERITY_ERROR	Terminate script execution upon database access errors. (default)	0
LRD_DB_ERROR_SEVERITY_WARNING	Continue script execution upon database access errors, but issue a warning.	1

For example, if the following database statement fails (e.g. the table does not exist), the script execution terminates.

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)\n", -1, 1, 1, 0);
```

To instruct VuGen to continue script execution, even when a database operation error occurs, change the statement's severity level from 0 to 1.

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)\n", -1, 1, 1, 1);
```

Note: When you enable Continue on Error, it overrides the "0" severity level; script execution continues even when database errors occur. However, if you disable Continue on Error, but you specify a severity level of "1", script execution continues when database errors occur.

Error Handling for RTE Vusers

When working with RTE Vusers, you can control error handling for specific functions. You insert an **lr_continue_on_error(0);** statement before the function whose behavior you want to change. The Vuser uses the new setting until the end of the script execution or until another **lr_continue_on_error** statement is issued.

For example, if you enable the Continue on Error feature and the Vuser encounters an error during replay of the following script segment, it continues executing the script.

```
TE_wait_sync();
TE_type(...);
```

To instruct the Vuser to continue on error for the entire script, except for the following segment, select the Continue on Error option and enclose the segment with lr_continue_on_error statements, using 0 to turn off Continue on Error and 1 to turn it back on:

```
lr_continue_on_error(0);
TE_wait_sync();
lr_continue_on_error(1);
....
```

Multithreading

Vusers support multithread environments. The primary advantage of a multithread environment is the ability to run more Vusers per load generator. Only threadsafe protocols should be run as threads. (not applicable to HP Business Availability Center)

Note: The following protocols are not threadsafe: Sybase-Ctlib, Sybase-Dblib, Informix, Tuxedo, and PeopleSoft-Tuxedo.

- ➤ To enable multithreading, click **Run Vuser as a thread**.
- ➤ To disable multithreading and run each Vuser as a separate process, click **Run Vuser as a process**.

The Controller uses a driver program (such as *mdrv.exe* or *r3vuser.exe*) to run your Vusers. If you run each Vuser as a process, then the same driver program is launched (and loaded) into the memory again and again for every instance of the Vuser. Loading the same driver program into memory uses up large amounts of RAM (random access memory) and other system resources. This limits the numbers of Vusers that can be run on any load generator.

Alternatively, if you run each Vuser as a thread, the Controller launches only one instance of the driver program (such as *mdrv.exe*), for every 50 Vusers (by default). This driver process/program launches several Vusers, each Vuser running as a thread. These threaded Vusers share segments of the memory of the parent driver process. This eliminates the need for multiple re-loading of the driver program/process saves much memory space, thereby enabling more Vusers to be run on a single load generator.

Automatic Transactions

You can instruct LoadRunner (not applicable to HP Business Availability Center) to handle every step or action in a Vuser script as a transaction. This is called using automatic transactions. LoadRunner assigns the step or action name as the name of the transaction. By default, automatic transactions per action are enabled.

- To disable automatic transactions per action, clear the **Define each action as a transaction** check box. (enabled by default)
- To enable automatic transactions per step, check the **Define each step as a transaction** check box. (disabled by default)

If you disable automatic transactions, you can still insert transactions manually during and after recording. For more information on manually inserting transactions, see Chapter 6, "Enhancing Vuser Scripts."

Note: If you require the Vusers to generate breakdown data for diagnostics (J2EE) during the scenario run, do not use automatic transactions. Instead, manually define the beginning and end of each transaction.

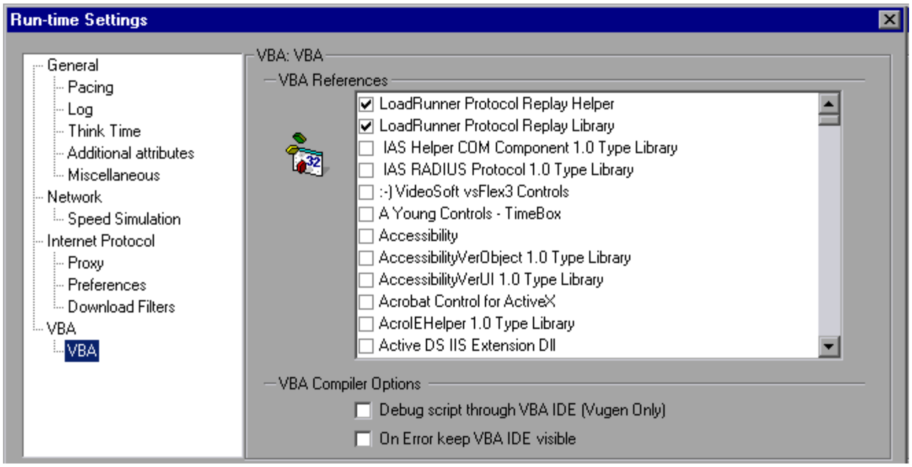
Setting the VB Run-Time Settings

Before running your Visual Basic script, you indicate which libraries to reference during replay. VuGen displays a list of all of the libraries stored on the machine.

You use the Run-Time Settings dialog box to display and configure the run-time settings. To display the Run-Time Settings dialog box, click the **Run-Time Settings** button on the VuGen toolbar.

To set the VBA Run-Time settings:

1 Open the **Run-Time Settings** dialog box and select the **VBA:VBA** node.



Run-time Settings

- General
  - Pacing
  - Log
  - Think Time
  - Additional attributes
  - Miscellaneous
- Network
  - Speed Simulation
- Internet Protocol
  - Proxy
  - Preferences
  - Download Filters
- VBA
  - VBA

VBA: VBA

VBA References

☑ LoadRunner Protocol Replay Helper
☑ LoadRunner Protocol Replay Library
□ IAS Helper COM Component 1.0 Type Library
□ IAS RADIUS Protocol 1.0 Type Library
□ :-) VideoSoft vsFlex3 Controls
□ A Young Controls - TimeBox
□ Accessibility
□ AccessibilityVerObject 1.0 Type Library
□ AccessibilityVerUI 1.0 Type Library
□ Acrobat Control for ActiveX
□ AcroIEHelper 1.0 Type Library
□ Active DS IIS Extension Dll

VBA Compiler Options

□ Debug script through VBA IDE (Vugen Only)
□ On Error keep VBA IDE visible

2 In the **VBA References** section, select the reference library that you want to use while running the script. Select a library to display its description and version in the bottom of the dialog box.

3 Select the appropriate compiler options:

Select **Debug script through VBA IDE** to enable debugging through the Visual Basic IDE (Integrated Development Environment).

Select **On Error keep VBA IDE visible** to keep the Visual Basic IDE visible during script execution.

4 Select **OK** to apply the run-time settings.

23

Configuring Network Run-Time Settings

To simulate the speed over a network, you configure the Network run-time settings.

This chapter includes:

- About Network Run-Time Settings on page 440
- Setting the Network Speed on page 440
- Setting Proxy Options on page 442
- Setting Browser Emulation Properties on page 447
- Setting Internet Preferences on page 452
- Filtering Web Sites on page 461
- Obtaining Debug Information on page 462
- Performing HTML Compression on page 463
- Checking Web Page Content on page 464

The following information applies to all Internet-related protocols, Citrix ICA, Oracle NCA, and WinSock.

For information about the general run-time settings that apply to all Vusers, see Chapter 22, "Configuring Run-Time Settings."

About Network Run-Time Settings

After developing a Vuser script, you set the run-time settings. These settings let you configure your Internet environment so that Vusers can accurately emulate real users. You can set Internet-related run-time settings for Proxy, Browser, Speed Simulation, and other advanced preferences.

You set the run-time settings by opening the Run-Time Settings dialog box and selecting the appropriate node:

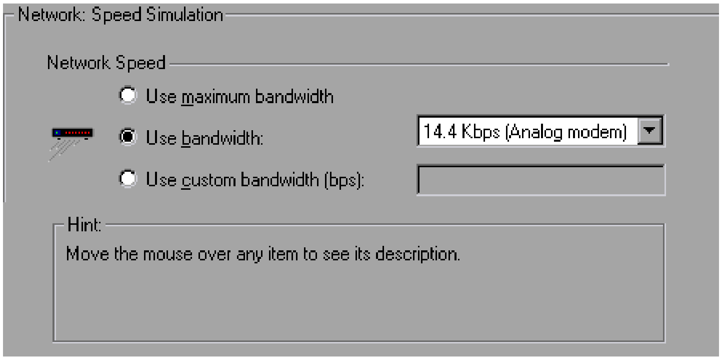
To display the Run-Time Settings dialog box:

- Click the **Run-Time Settings** button on the VuGen toolbar.
- Press the keyboard shortcut key **F4**.
- Select **Vuser** > **Run-Time Settings**.

Note that you can also modify the run-time settings from the LoadRunner Controller. For more information, see your product's documentation.

Setting the Network Speed

You use the **Network:Speed Simulation** node in the Run-Time Settings tree, to set the modem emulation for your testing environment.



Network: Speed Simulation

Network Speed

○ Use maximum bandwidth

● Use bandwidth: 14.4 Kbps (Analog modem)

○ Use custom bandwidth (bps):

Hint:

Move the mouse over any item to see its description.

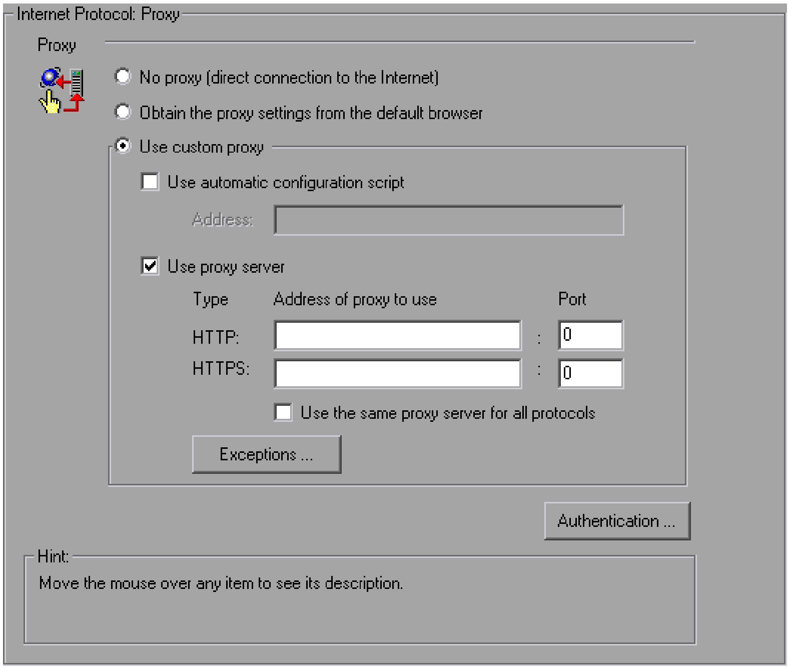
Speed Simulation

Using the Speed Simulation settings, you can select a bandwidth that best emulates the environment under test. The following options are available:

- **Use maximum bandwidth**. By default, bandwidth emulation is disabled and the Vusers run at the maximum bandwidth that is available over the network.
- **Use bandwidth.** Indicate a specific bandwidth level for your Vuser to emulate. You can select a speed ranging from 14.4 to 512 Kbps, emulating analog modems, ISDN, or DSL.
- **Use custom bandwidth.** Indicate a bandwidth limit for your Vuser to emulate. Specify the bandwidth in bits, where 1 Kilobit=1024 bits.

Setting Proxy Options

You use the **Internet Protocol:Proxy** node of the Run-Time Settings tree, to set the proxy-related settings.



Internet Protocol: Proxy

Proxy

○ No proxy (direct connection to the Internet)

○ Obtain the proxy settings from the default browser

● Use custom proxy

□ Use automatic configuration script

Address:

☑ Use proxy server

Type	Address of proxy to use	Port
HTTP:		0
HTTPS:		0

□ Use the same proxy server for all protocols

Hint:
Move the mouse over any item to see its description.

The following proxy options are available in the Run-Time settings.

- **No proxy.** All Vusers should use direct connections to the Internet. This means that the connection is made without using a proxy server.
- **Obtain the proxy settings from the default browser.** All Vusers use the proxy settings of the default browser from the machine upon which they are running.

- **Use custom proxy.** All Vusers use a custom proxy server. You can supply the actual proxy server details or the path of a proxy automatic configuration script (.**pac** file) that enables automatic configuration. (See "Setting the Automatic Proxy Configuration" on page 444.)

  To supply the details of the server, you specify its IP address or name and port. You can specify one proxy server for all HTTP sites, and another proxy server for all HTTPS (secure) sites.

  After providing the proxy information, you can specify Authentication information for the proxy server, and indicate Exceptions to the proxy rules.

Note: To instruct the Vusers to wait for the proxy response during replay, and not to assume that the proxy supports basic authentication, add the following statement:

```
web_set_sockets_option("PROXY_INITIAL_BASIC_AUTH", "0");
```

Authentication

If the proxy server requires authentication for each Vuser, use this dialog box to enter the relevant password and user name.

- **User Name.** Enter the user name that Vusers will use to access the proxy server.
- **Password.** Enter the password required by Vusers to access the proxy server.

Note: To add authentication dynamically during recording, or to add authentication for multiple proxy servers, use the **web_set_user** function. For more information, see the *Online Function Reference* (**Help** > **Function Reference**).

Exceptions

You can specify that all Vusers use a specified proxy server. In such a case, if there are any URLs that you want Vusers to access directly, that is, without using the proxy server, enter the list of these URLs in the text box.

- **Do not use proxy server for addresses beginning with.** Enter the addresses you want to exclude from the proxy server. Use semicolons to separate entries.
- **Do not use proxy server for local (intranet) addresses.** Select this check box to exclude local addresses, such as those from an Intranet, from the proxy server.

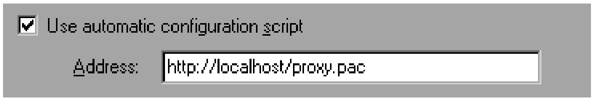
Setting the Automatic Proxy Configuration

Automatic Proxy Configuration is a feature supported by most browsers. This feature allows you to specify a JavaScript file (usually with a **.pac** extension) containing proxy assignment information. This script tells the browser when to access a proxy server and when to connect directly to the site, depending on the URL. In addition, it can instruct the browser to use a specific proxy server for certain addresses and another server for other addresses.

You can instruct VuGen or your Internet Explorer browser to work with a configuration script. You specify a file for the automatic proxy configuration, so that when the Vuser runs the test, it uses the rules from the proxy file.

To specify a configuration script in VuGen:

1. Select **Vuser** > **Run-TIme Settings**, and select the **Internet Protocol:Proxy** node.
2. Select **Use custom proxy** and select the **Use automatic configuration script** option. Specify the location of the script.



☑ Use automatic configuration script

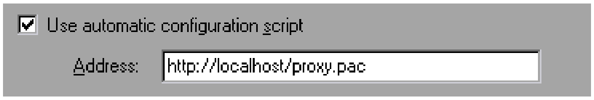
Address: http://localhost/proxy.pac

To specify a configuration script in Internet Explorer (IE):

1 Select **Tools** > **Internet Options**, and select the Connections tab.

2 Click the **LAN Settings** button. The LAN Settings dialog box opens.

3 Select the **Use automatic configuration script** option, and specify the location of the script.



☑ Use automatic configuration script

Address: http://localhost/proxy.pac

To track the behavior of the Vusers, generate a log during text execution and view the Execution Log tab or the mdrv.log file. The log shows the proxy servers that were used for each URL. In the following example, VuGen used a direct connection for the URL australia.com, but the proxy server aqua, for the URL http://www.google.com.

```
Action1.c(6): t=1141ms: FindProxyForURL returned DIRECT
Action1.c(6): t=1141ms: Resolving australia.com
Action1.c(6): t=1141ms: Connecting to host 199.203.78.255:80
…
Action1.c(6): t=1281ms: Request done "http://australia.com/GetElementByName.htm"

…
Action1.c(6): web_url was successful, 357 body bytes, 226 header bytes
Action1.c(15): web_add_cookie was successful
Action1.c(17): t=1391ms: FindProxyForURL returned PROXY aqua:2080
Action1.c(17): t=1391ms: Auto-proxy configuration selected proxy aqua:2080
Action1.c(17): t=1391ms: Resolving aqua
Action1.c(17): t=1391ms: Connecting to host 199.203.139.139:2080
…
Action1.c(17): t=1578ms: 168-byte request headers for "http://www.google.com/"
(RelFrameId=1)
Action1.c(17):     GET http://www.google.com/ HTTP/1.1\r\n
```

Setting Proxy Run-Time Settings

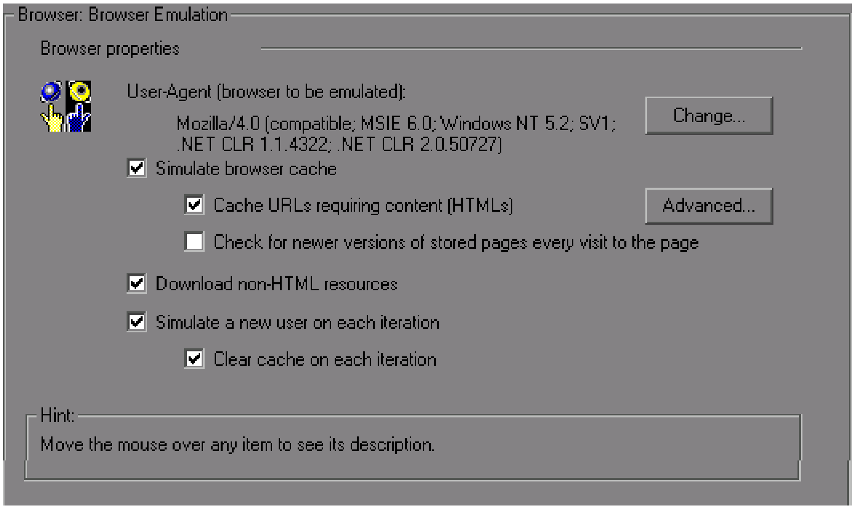
The following section discusses the steps required for configuring the proxy Run-Time settings.

To set the proxy settings:

1. Open the Run-Time settings. Click the **Run-Time Settings** button on the VuGen toolbar or select **Vuser** > **Run-Time Settings**.
2. Click the **Internet Protocol:Proxy** node.
3. Select the desired proxy option: **No proxy, Obtain the proxy settings from the default browser, or Use custom proxy.**
4. If you specified a custom proxy:
   - indicate the IP addresses for the HTTP and HTTPS proxy servers
   - To use a **pac** or JavaScript file to indicate the proxy, select the **Use automatic configuration script** option and specify the script location. You can specify either a web location beginning with http:// (for example, **http://hostname/proxy.pac**), or a location on the file server, for example, **C:\temp\proxy.pac**.
5. To specify URLs that you want Vusers to access directly, without the proxy server, click **Exceptions** and then supply the list of these URLs. In the Exceptions dialog box, you can also specify direct access to local (intranet) addresses.
6. If the proxy server requires authentication, click **Authentication**, and then supply the relevant password and user name.
7. Select the **Use the same proxy server for all protocols** check box to instruct the Vusers to use the same proxy server for all Internet protocols (HTTP, HTTPS) rather than specifying a specific server for secure sites.

Setting Browser Emulation Properties

You use the **Browser:Browser Emulation** node in the Run-Time Settings tree to set the browser properties of your testing environment.



Browser: Browser Emulation

Browser properties

User-Agent (browser to be emulated):

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)

☑ Simulate browser cache

☑ Cache URLs requiring content (HTMLs)

□ Check for newer versions of stored pages every visit to the page

☑ Download non-HTML resources

☑ Simulate a new user on each iteration

☑ Clear cache on each iteration

Hint:

Move the mouse over any item to see its description.

Browser Properties

You can set the browser properties in the following areas:

- User-Agent (browser to be emulated)
- Simulate browser cache
- Download non-HTML resources
- Simulate a new user each iteration

You can also set advanced options for caching and checking for newer resources.

User-Agent (browser to be emulated)

Whenever a Vuser sends a request to a Web server, the request includes an HTTP header. The first line of text contains a verb (usually "GET" or "POST"), the resource name (for example "pclt/default.htm"), and the version of the protocol (for example "HTTP/1.0"). Subsequent lines contain "header information" in the form of an attribute name, a colon, and some value. The request ends with a blank line.

All Internet Vuser headers include a **User-Agent** header that identifies the type of browser (or toolkit for Wireless) that is being emulated. For example,

```
User-Agent: Mozilla/3.01Gold (WinNT; I)
```

identifies the Browser as Netscape Navigator Gold version 3.01 running under Windows NT on an Intel machine.

Click **Change** from the Browser emulation node, to specify the browser information to include in the header. You can specify that a Web Vuser emulate any of the standard browsers. Alternatively, for non-browser HTTP applications, you can specify the HTTP client to match a specific user's application. In this case, you must supply a **Custom User Agent** string that is included in all subsequent HTTP headers. By default, the user-agent emulates the Microsoft Internet Explorer 5.5 browser agent.

Simulate browser cache

This option instructs the Vuser to simulate a browser with a cache. A cache is used to keep local copies of frequently accessed documents and thereby reduces the time connected to the network. By default, cache simulation is enabled. When the cache is disabled, Vusers will ignore all caching functionality and download all of the resources for every request.

Note that even if you disable the cache simulation, each resource is only downloaded once for each page, even if it appears multiple times. A resource can be an image, a frame, or another type of script file.

When running multiple Vusers as in LoadRunner and Performance Center, every Vuser uses its own cache and retrieves images from the cache. If you disable this option, all Vusers emulate a browser with no cache available.

You can modify your Run-Time settings to match your browser settings for Internet Explorer, as follows:

Browser Setting	Run-Time Setting
Every visit to the page	Select **Simulate Browser Cache** and enable **Check for newer versions of stored pages every visit to the page**.
Every time you start Internet Explorer	Select **Simulate Browser Cache** only
Automatically	Select **Simulate Browser Cache** only
Never	Select **Simulate Browser Cache** and disable **Check for newer versions of stored pages every visit to the page**.

You can also set the following two browser cache options:

- **Cache URLs requiring content (HTML).** This option instructs VuGen to cache only the URLs that require the HTML content. The content may be necessary for parsing, verification, or correlation. When you select this option, HTML content is automatically cached. This option is enabled by default.

Tip: To decrease the memory footprint of the virtual users, disable this option, unless it is an explicit requirement for your test.

To add more content types to the list of cached types, click **Advanced**. For more information, see "Cache URLs Requiring Content - Advanced" on page 451. Note that if you enable the parent option **Simulate browser cache**, but disable this option, VuGen nevertheless stores the graphic files.

- **Check for newer versions of stored pages every visit to the page.** This setting instructs the browser to check for later versions of the specified URL, than those stored in the cache. When you enable this option, VuGen adds the "If-modified-since" attribute to the HTTP header. This option brings up the most recent version of the page, but also generates more traffic during the scenario or session execution. By default, browsers do not check for newer resources, and therefore this option is disabled. Configure this option to match the settings in the browser that you want to emulate.

Download non-HTML resources

Instructs Vusers to load graphic images when accessing a Web page during replay. This includes both graphic images that were recorded with the page, and those which were not explicitly recorded along with the page. When real users access a Web page, they wait for the images to load. Therefore, enable this option if you are trying to test the entire system, including end-user time (enabled by default). To increase performance and not emulate real users, disable this option.

Tip: Disable this option if you experience discrepancies in image checks, since some images vary each time you access a Web page (for example, advertiser banners).

Simulate a new user each iteration

Instructs VuGen to reset all HTTP contexts between iterations to their states at the end of the **init** section. This setting allows the Vuser to more accurately emulate a new user beginning a browsing session. It deletes all cookies, closes all TCP connections (including keep-alive), clears the emulated browser’s cache, resets the HTML frame hierarchy (frame numbering will begin from 1) and clears the user-names and passwords. This option is enabled by default.

- **Clear cache on each iteration.** Clears the browser cache for each iteration in order to simulate a user visiting a Web page for the first time. Clear the check box to disable this option and allow Vusers to use the information stored in the browser's cache, simulating a user who recently visited the page.

Cache URLs Requiring Content - Advanced

The Advanced dialog box lets you specify the URL content types that you want to store in the cache. This dialog box is accessible from the Run-time Settings **- Browser:Browser Emulation** node.

Note that changes to the advanced settings for multiple groups simultaneously, are not supported—edit each group's settings individually.

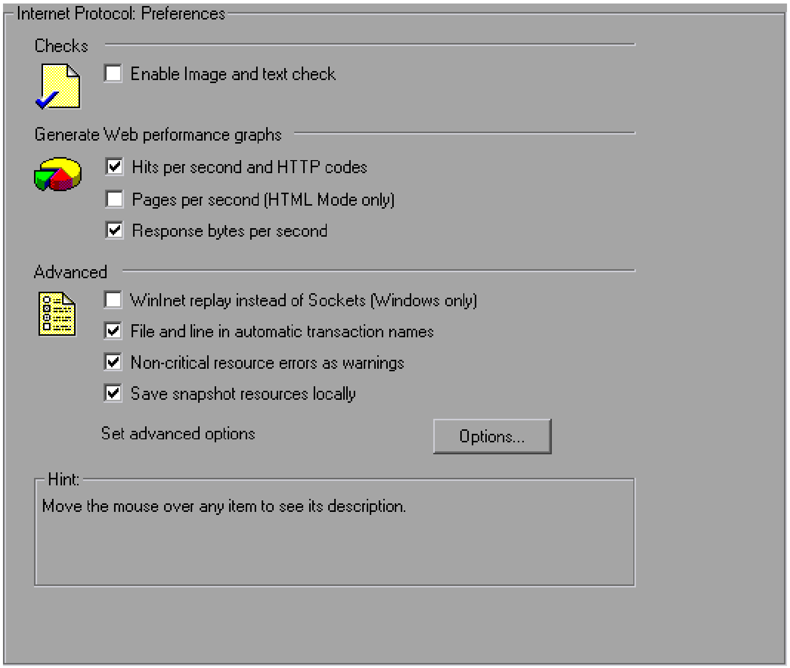
To add a content type:

1. Enable the **Specify URLs requiring content in addition to HTML page** option.
2. Click the plus sign to add additional content types, such as text/plain, text/xml, image/jpeg, and image/gif. Enter the content name in the text box.
3. To remove a content type from the list, select it and click the minus sign.

Setting Internet Preferences

You use the **Internet Protocol:Preferences** node in the Run-Time Settings tree, to set the settings related to the following areas:

- Image and Text Checks
- Generating Web Performance Graphs
- Advanced Web Run-Time Options



Internet Protocol: Preferences

Checks

☐ Enable Image and text check

Generate Web performance graphs

☑ Hits per second and HTTP codes

☐ Pages per second (HTML Mode only)

☑ Response bytes per second

Advanced

☐ WinInet replay instead of Sockets (Windows only)

☑ File and line in automatic transaction names

☑ Non-critical resource errors as warnings

☑ Save snapshot resources locally

Set advanced options

Hint:

Move the mouse over any item to see its description.

Image and Text Checks

The **Enable image and text checks** option allows the Vuser to perform verification checks during replay by executing the verification functions: **web_find** or **web_image_check**. This option only applies to statements recorded in HTML-based mode. Vusers running with verification checks use more memory than Vusers who do not perform checks (disabled by default).

Generating Web Performance Graphs

Instructs a Vuser to collect data used to create Web Performance graphs. You view the **Hits per Second**, **Pages per Second,** and **Response Bytes per Second (Throughput)** graphs during test execution using the online monitors and after test execution using the Analysis. You view the Component Breakdown graph after test execution using the Analysis. Select the types of graph data for the Vuser to collect.

Note: If you do not use the Web performance graphs, disable these options to save memory.

Advanced Web Run-Time Options

- **WinInet Replay.** Instructs VuGen to use the WinInet replay engine instead of the standard Sockets replay. VuGen has two HTTP replay engines: Sockets-based (default) or WinInet based. The WinInet is the engine used by Internet Explorer and it supports all of the features incorporated into the IE browser. The limitations of the WinInet replay engine are that it is not scalable, nor does it support UNIX. In addition, when working with threads, the WinInet engine does not accurately emulate the modem speed and number of connections.

VuGen's proprietary sockets-based replay is a lighter engine that is scalable for load testing. It is also accurate when working with threads. The limitation of the sockets-based engine is that it does not support SOCKS proxy. If you are recording in that type of environment, use the WinInet replay engine.

- **File and line in automatic transaction names.** Creates unique transaction names for automatic transactions by adding file name and line number to the transaction name (enabled by default).

Note: This option places additional information in the log file, and therefore requires more memory.

- **Non-critical item errors as warnings.** This option returns a warning status for a function which failed on an item that is not critical for load testing, such as an image or Java applet that failed to download. This option is enabled by default. If you want a certain warning to be considered an error and fail your test, you can disable this option. You can set a content-type to be critical by adding it to the list of Non-Resources. For more information, see "Specifying Non-Resource Content Types" on page 338.
- **Save snapshot resources locally.** Instructs VuGen to save the snapshot resources to files on the local machine. This feature lets the Run-Time viewer create snapshots more accurately and display them quicker.

Additional Options for Internet Preferences

Click the **Options** button in the Advanced section of the Preferences node to set advanced options in the following areas: DNS caching, HTTP version, Keep-Alive HTTP connections, Accept server-side compression, Accept-Language headers, HTTP-request connect timeout, HTTP-request receive timeout, Network buffer size, and Step download timeout.

HTTP

- **HTTP version.** Specifies which version HTTP to use: version 1.0 or 1.1. This information is included in the HTTP request header whenever a Vuser sends a request to a Web server. The default is HTTP 1.1. HTTP 1.1 supports the following features:
  - Persistent Connections—see "Keep-Alive HTTP connections" below.
  - HTML compression—see "Performing HTML Compression" on page 463.
  - Virtual Hosting—multiple domain names sharing the same IP address.
- **Keep-Alive HTTP connections.** Keep-alive is a term used for an HTTP extension that allows persistent or continuous connections. These long-lived HTTP sessions allow multiple requests to be sent over the same TCP connection. This improves the performance of the Web server and clients.

The keep-alive option works only with Web servers that support keep-alive connections. This setting specifies that all Vusers that run the Vuser script have keep-alive HTTP connections enabled (enabled by default).

➤ **Accept-Language request header.** Provides a comma-separated list of accepted languages. For example, **en-us**, **fr**, and so forth.

➤ **HTTP errors as warnings.** Issues a warning instead of an error upon failing to download resources due to an HTTP error.

➤ **HTTP-request connect timeout (seconds).** The time, in seconds, that a Vuser will wait for the connection of a specific HTTP request within a step before aborting. Timeouts provide an opportunity for the server to stabilize and respond to the user (default value is 120 seconds). Note that this timeout also applies to the time the Vuser will wait for a WAP connection, initiated by the **wap_connect** function.

➤ **HTTP-request receive timeout (seconds).** The time, in seconds, that a Vuser will wait to receive the response of a specific HTTP request within a step before aborting. Timeouts provide an opportunity for the server to stabilize and respond to the user (default value is 120 seconds).

➤ **Request Zlib Headers.** Sends request data to the server with the **zlib** compression library headers. By default, requests sent to the server include the **zlib** headers. This option lets you emulate non-browser applications that do not include **zlib** headers in their requests. To exclude these headers, set this option to **No** (default is Yes).

➤ **Accept Server-Side Compression.** Indicate to the server that the replay can accept compressed data. The available options are: **None** (no compression), **gzip** (accept gzip compression), **gzip, deflate** (accept gzip or deflate compression), and **deflate** (accept deflate compression). Note that by accepting compressed data, you may significantly increase the CPU consumption. The default is to accept **gzip, deflate** compression.

General

➤ **DNS caching.** Instructs the Vuser to save a host's IP addresses to a cache after resolving its value from the Domain Name Server. This saves time in subsequent calls to the same server. In situations where the IP address changes, as with certain load balancing techniques, be sure to disable this option to prevent Vuser from using the value in the cache (enabled by default).

- **Convert from/to UTF-8.** Converts received HTML pages and submitted data from and to UTF-8. You enable UTF-8 support in the recording options. For more information, see "Web, Wireless, and Oracle NCA Recording Options" on page 332 (No by default).
- **Step timeout caused by resources is a warning.** Issues a warning instead of an error when a timeout occurs due to a resource that did not load within the timeout interval. For non-resources, VuGen issues an error (disabled by default).
- **Parse HTML Content-Type.** When expecting HTML, parse the response only when it is the specified content-type: **HTML**, **text\html**, **TEXT** any text, or **ANY**, any content-type. Note that text/xml is not parsed as HTML. The default is **TEXT**.

The timeout settings are primarily for advanced users who have determined that acceptable timeout values should be different for their environment. The default settings should be sufficient in most cases. If the server does not respond in a reasonable amount of time, check for other connection-related issues, rather than setting a very long timeout which could cause the scripts to wait unnecessarily.

- **Step download timeout (sec).** The time that the Vuser will wait before aborting a step in the script. This option can be used to emulate a user behavior of not waiting for more than x seconds for a page.
- **Network buffer size.** Sets the maximum size of the buffer used to receive the HTTP response. If the size of the data is larger than the specified size, the server will send the data in chunks, increasing the overhead of the system. When running multiple Vusers from the Controller, every Vuser uses its own network buffer. This setting is primarily for advanced users who have determined that the network buffer size may affect their script's performance. The default is 12K bytes. The maximum size is 0x7FFF FFFF.
- **Print NTLM information.** Print information about the NTLM handshake to the standard log.
- **Print SSL information.** Print information about the SSL handshake to the standard log.

- **Max number of error matches issued as ERRORS.** Limits the number of error matches issued as ERRORS for content checks using a LB or RB (left boundary or right boundary). This applies to matches where a failure occurs when the string is found (Fail=Found). All subsequent matches are listed as informational messages. The default is 10 matches.
- **Maximum number of META Refresh to the same page.** The maximum number of times that a META refresh can be performed per page. The default is 2.
- **ContentCheck values in UTF-8.** Store the values in the ContentCheck XML file in UTF-8.

Authentication

- **Fixed think time upon authentication retry (msec).** Automatically adds a think time to the Vuser script for emulating a user entering authentication information (username and password). This think time will be included in the transaction time (default is 0).
- **Disable NTLM2 session security.** Use full NTLM 2 handshake security instead of the more basic NTLM 2 session security response (default is No).
- **Use Windows native NTLM implementation.** Use the Microsoft Security API for NTLM authentication instead of the indigenous one (default is No).
- **Enable integrated Authentication.** Enable Kerberos-based authentication. When the server proposes authentication schemes, use **Negotiate** in preference to other schemes (default is No).
- **Induce heavy KDC load.** Do not reuse credentials obtained in previous iterations. Enabling this setting will increase the load on the KDC (Key Distribution Server). To lower the load on the server, set this option to **Yes** in order to reuse the credentials obtained in previous iterations. This option is only relevant when Kerberos authentication is used (default is No).

Log

- **Print buffer line length.** Line length for printing request/response header/body and/or JavaScript source, disabling wrapping.

- **Print buffer escape only binary zeros.**
  - **Yes.** Escape only binary zeros when printing request/response headers/body and/or JavaScript source.
  - **No**. Escape any unprintable/control characters.

Web (Click and Script) Specific

- **General**
  - **Home Page URL.** The URL of the home page that opens with your browser (default is about:blank).
  - **DOM-based snapshots.** Instructs VuGen to generate snapshots from the DOM instead of from the server responses (**Yes** by default).
  - **Charset conversions by HTTP.** Perform charset conversions by the 'Content-Type:....; charset=...' HTTP response header. Overrides 'Convert from /to UTF-8.'
  - **Reparse when META changes charset.** Reparse HTML when a META tag changes the charset. Effective only when **Charset conversions by HTTP** is enabled. **Auto** means reparsing is enabled only if it used in the first iteration.
  - **Fail on JavaScript error.** Fails the Vuser when a JavaScript evaluation error occurs. The default is No, issuing a warning message only after a JavaScript error, but continuing to run the script.
  - **Initialize standard classes for each new window project.** When enabled, the script—the src compiled script, will not be cached.
  - **Ignore acted on element being disabled.** Ignore the element acted on by a Vuser script function being disabled.
- **Timers**
  - **Optimize timers at end of step.** When possible, executes a setTimeout/setInterval/<META refresh> that expires at the end of the step before the expiration time (default is Yes).

  - **Single setTimeout/setInterval threshold (seconds).** Specifies an upper timeout for the window.setTimeout and window.setInterval methods. If the delay exceeds this timeout, these methods will not invoke the functions that are passed to them. This emulates a user waiting a specified time before clicking on the next element (default is 5 seconds).
  - **Accumulative setTimeout/setInterval threshold (seconds).** Specifies a timeout for the window.setTimeout and window.setInterval methods. If the delay exceeds this timeout, additional calls to window.setTimeout and window.setInterval will be ignored. The timeout is accumulative per step (default is 30 seconds).
  - **Reestablish setInterval at end of step.** **0** = No; **1** = Once; **2** = Yes.
- **History**
  - **History support.** Enables support for the window.history object for the test run. The options are Enabled, Disabled, and Auto. The Auto option (default) instructs Vusers to support the window.history object only if it was used in the first iteration. Note that by disabling this option, you improve performance.
  - **Maximum history size.** The maximum number of steps to keep in the history list (default is 100 steps).
- **Navigator Properties**
  - **navigator.browserLanguage.** The browser language set in the navigator DOM object's **browserLanguage** property. The default is the recorded value. Scripts created with older recording engines, use **en-us** by default.
  - **navigator.systemLanguage.** The system language set in the navigator DOM object's **systemLanguage** property. The default is the recorded value. Scripts created with older recording engines, use **en-us** by default.
  - **navigator.userLanguage.** The user language set in the navigator DOM object's **userLanguage** property. The default is the recorded value. Scripts created with older recording engines, use **en-us** by default.
- **Screen Properties**

  - **screen.width** Sets the width property of the screen DOM object in pixels (default is 1024 pixels).
  - **screen.height** Sets the height property of the screen DOM object in pixels (default is 768 pixels).
  - **screen.availWidth** Sets the availWidth property of the screen DOM object in pixels (default is 1024 pixels).
  - **screen.availHeight.** Sets the availHeight property of the screen DOM object in pixels (default is 768 pixels).
- **Memory Management**
  - **Default block size for DOM memory allocations.** Sets the default block size for DOM memory allocations. If the value is too small, it may result in extra calls to malloc, slowing the execution times. Too large a block size, may result in an unnecessarily big footprint (default is 16384 bytes).
  - **Memory Manager for dynamically-created DOM objects. Yes**—Use the Memory Manager for dynamically-created DOM objects. **No—**Do not use the Memory Manager, for example when multiple DOM objects are dynamically created in the same document as under SAP. **Auto**—Use the protocol recommended (default Yes for all protocols except for SAP).
  - **JavaScript Runtime memory size (KB).** Specifies the size of the JavaScript runtime memory in kilobytes (default is 256 KB).
  - **JavaScript Stack memory size (KB).** Specifies the size of the JavaScript stack memory in kilobytes (default is 32 KB).

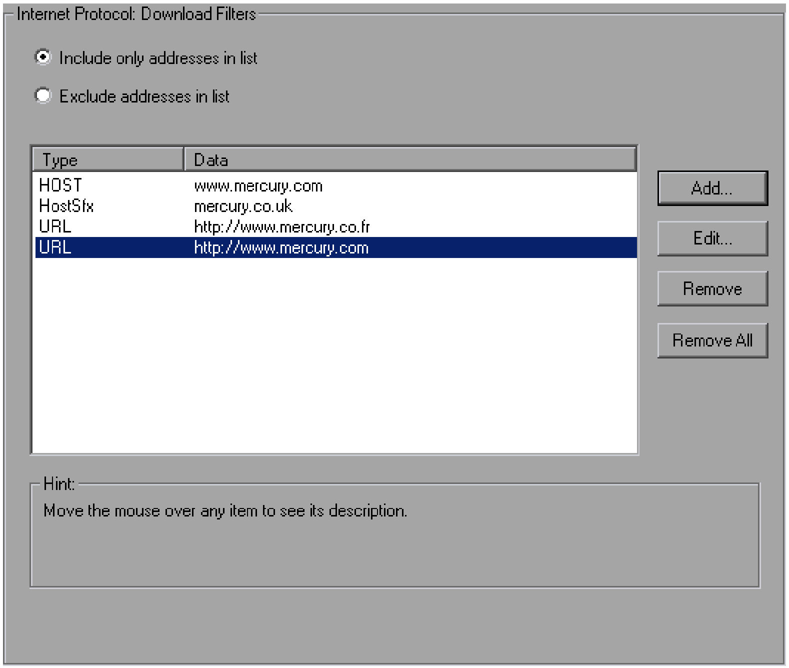
Filtering Web Sites

You can specify the Web sites from which Vusers should download resources during replay. You can indicate either the sites to exclude or the sites to include. You control the allowed or disallowed sources, by specifying a URL, host name, or host suffix name.

A **URL** is the complete URL address of a Web site, beginning with http:// or https://. **Host** is the name of the host machine with its domain, such as www.hp.com.

Host suffix is the common suffix for several host names, such as hp.com. This is useful where you have several Web sites on a common domain.

If you specify the sites to exclude, VuGen downloads resources from all Web sites except for those specified in the list. If you specify the sites to include, VuGen filters out resources from all Web sites except for those in the Include list.



Internet Protocol: Download Filters

● Include only addresses in list

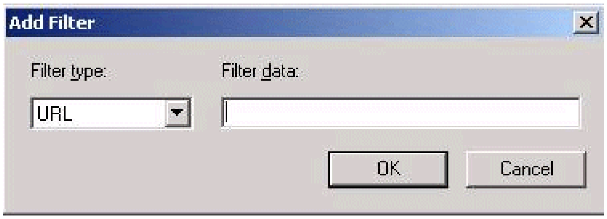
○ Exclude addresses in list

Type	Data
HOST	www.mercury.com
HostSfx	mercury.co.uk
URL	http://www.mercury.co.fr
URL	http://www.mercury.com

Hint:
Move the mouse over any item to see its description.

To create a list of filtered Web sites:

1. Click the **Internet Protocol:Download Filters** node.
2. Select the desired option: **Include only addresses in list** or **Exclude addresses in list**.
3. Add entries to the list. To add an entry, click **Add**. The Add filter dialog box opens.



Add Filter

Filter type: URL

Filter data:

Select a filter type: **URL**, **Host**, or **Host Suffix**, and enter the filter data, such as a URL. When entering a URL, make sure to enter a complete URL beginning with **http://** or **https://**. Click **OK**.

4. To edit an entry, select it and click **Edit**.
5. To delete and entry, select it and click **Remove**. To delete all entries, click **Remove All**.

Obtaining Debug Information

When you run a Vuser script, the execution information is displayed in the Output window or log file. You control the amount of information sent to the Output window and log files, using the **Log** node of the General run-time settings. For more information, see "Configuring the Log Run-Time Settings" on page 424.

Debug information includes:

- log information
- transaction failures
- the connection status with the gateway—connecting, disconnecting, and redirecting. (WAP only)

To obtain more information for debugging, edit the **default.cfg** file. Locate the **WEB** section and set the **LogFileWrite** flag to **1**. The resulting trace file documents all events in the execution of the script.

When performing load testing, make sure to clear the **LogFileWrite** flag to prevent the Vusers from wasting resources by creating a large trace file.

Performing HTML Compression

Browsers that support HTTP 1.1 can decompress HTML files. The server compresses the files for transport, substantially reducing the bandwidth required for the data transfer. You can enable compression automatically or manually.

To automatically enable compression in VuGen, use the **Internet Protocol** > **Preferences** node of the Run-Time settings. Click **Options** to open the Advanced Options and enable the **Accept Server-Side compression** option. Note that this option is enabled by default. For more information, see "Additional Options for Internet Preferences" on page 454.

To manually add compression, enter the following function at the beginning of the script:

```
web_add_auto_header("Accept-Encoding", "gzip");
```

To verify that the server sent compressed data, search for the string **Content -Encoding: gzip** in the section of the server's responses of the Execution log. The log also shows the data size before and after decompression.

Compression has a greater effect on large data transfers—the larger the data, the greater effect the compression will have. When working with larger data, you can also increase the network buffer size (see the Network Buffer Size option) to get the data in single chunks.

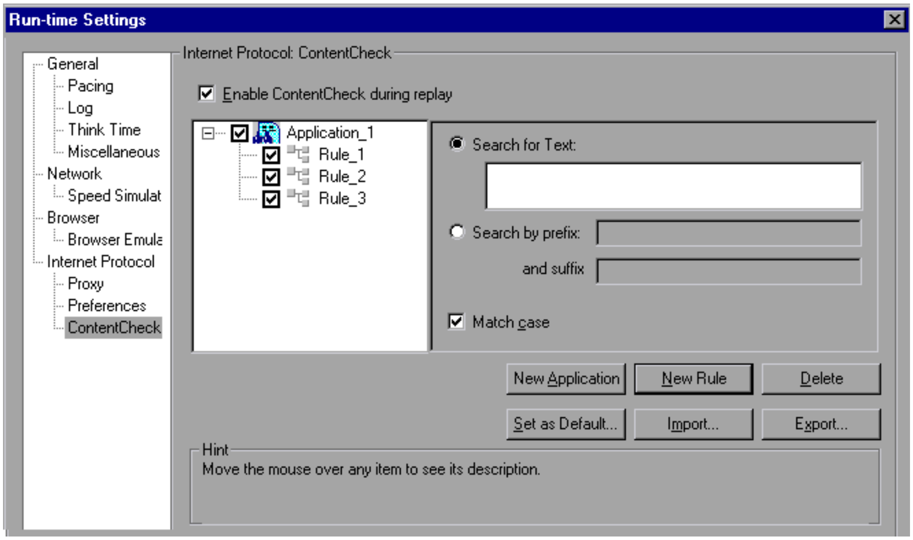
Checking Web Page Content

VuGen's Content Check mechanism allows you to check the contents of a page for a specific string. This is useful for detecting non-standard errors. In normal operations, when your application server fails, the browser displays a generic HTTP error page indicating the nature of the error. The standard error pages are recognized by VuGen and treated as errors, causing the script to fail. Some application servers, however, issue their own error pages that are not detected by VuGen as error pages. The page is sent by the server and it contains a formatted text string, stating that an error occurred.

For example, suppose that your application issues a custom page when an error occurs, containing the text **ASP Error**. You instruct VuGen to look for this text on all returned pages. When VuGen detects this string, it fails the replay. Note that VuGen searches the body of the pages—not the headers.

You use the **Internet Protocol:ContentCheck** Run-Time setting to specify the content for which you want to search. You can define content for several applications with multiple rules. The following sections discuss:

- ➤ Understanding Content Rules
- ➤ Defining ContentCheck Rules



Internet Protocol: ContentCheck

☑ Enable ContentCheck during replay

● Search for Text:

○ Search by prefix:

and suffix

☑ Match case

Hint
Move the mouse over any item to see its description.

Understanding Content Rules

You use the ContentCheck run-time options to check the contents of a page for a specific string. This is useful for detecting non-standard errors. In normal operations, when your application server fails, the browser displays a generic HTTP error page indicating the nature of the error. The standard error pages are recognized by VuGen and treated as errors, causing the script to fail. Some application servers, however, issue their own error pages that are not detected by VuGen as error pages. The page is sent by the server and it contains a formatted text string, stating that an error occurred.

For example, suppose that your application issues a custom page when an error occurs, containing the text **ASP Error**. You instruct VuGen to look for this text on all returned pages. When VuGen detects this string, it fails the replay. Note that VuGen searches the body of the pages—not the headers.

- **Enable ContentCheck during replay.** Enable content checking during replay (enabled by default). Note that even after you define applications, you can disable it for a specific test run, by disabling this option.

Rule Information

This right pane contains the matching criteria for the text you want to find. You can specify either the actual text or a prefix and suffix of the text.

- **Search for Text.** The text of the string for which you want to search.
- **Search by Prefix and Suffix.** The prefix and suffix of the string for which you want to search.
- **Match case.** Perform a case sensitive search.
- **Search JavaScript alert box text.** Only search for text within JavaScript alert boxes (Web (Click and Script), PeopleSoft Enterprise, and Oracle Web Applications 11i Vusers only).

Adding and Removing Applications and Rules

- **New Application.** Automatically adds a new application to the list of applications in the left pane. The default name is Application_**index**, beginning with **Application_1**. After you create a new application, click **New Rule** to add a rule to this application. To modify the name of an application, double-click on it.

- **New Rule.** Displays the rule criteria in the right pane, allowing you to enter a new rule for the currently selected application. The rules are stored with the script in standard xml files. You can export your rule files and share them with other users or import them to other machines.
- **Delete.** Deletes the selected rule or application.

Importing and Exporting Rules

- **Import/Export.** Imports or exports a rule file. The rule file with an **xml** extension, stores the applications and rules. You can export the file to use on other machines. You can also import other rule files. If you import a rule and the selected rule conflicts with an existing rule, VuGen issues a warning indicating that it is a Conflicting Rule. You can then merge the rules you created on a former script with the one you are importing or overwrite the current rules. When you click Export, VuGen opens the Choose Application to Export dialog box.

Setting Rules as Default

- **Set as Default.** There are three types of rules for Content Checks: **Installation**, **Default**, and **per script**. Installation rules are provided automatically during installation of the product. Default rules, apply to all scripts executed on your machine. The per script rules are the ones defined for the current script. When you modify or add rules, these changes only apply to the current script. To instruct VuGen to add a rule to the list of Default rules so that it will apply to all scripts on that machine, click **Set as Default**.

When working on multiple scripts, or when performing a product upgrade, a conflict may arise between the default rules and the script rules. VuGen asks you if you want to merge the rules. When you merge the rules (recommended), the rule is added to the list of rules for the application.

This action only effects applications that are enabled in the Application list (the left pane). If no applications were marked as Enabled in the current script, no application will be marked as Enabled in the Defaults file. Click **Yes** to overwrite the Defaults file. Click **No** to cancel the operation and retain the original Defaults file.

The rules are stored in standard xml files. You can export your rule files and share them with other users or import them to other machines.

When you click Set as Defaults (and confirm the overwriting), VuGen performs the following actions:

1 Marks all applications in the Defaults File as **Disabled**.

2 For applications marked as **Enabled** in the current script, it performs a merge or copy, depending on whether the application exists. If the application exists, it merges the rules of the current script with those of the Defaults file. If the application did not exist in the Defaults file, then VuGen just copies the rules to the Defaults file.

3 Marks the applications that were enabled in the script, as **Enabled** in the Defaults file. If no application is marked as **Enabled** in the current script, no application will be marked as **Enabled** in the Defaults file.

Use Defaults

Imports rules from the Defaults file. When you click this button, VuGen opens a dialog box with a list of the applications and their default settings. You can import these rules or modify them. If this conflicts with one of the existing rules, VuGen issues a warning indicating that it is a Conflicting Rule. You can also merge the rules defined in the Defaults file with the ones currently defined.

To use the default settings for all of your applications, click **Use Defaults** which imports the definitions from the Defaults file. It opens a dialog box with a list of the applications and their default settings. You can import these definitions or modify them. If this conflicts with one of the rules, VuGen issues a warning indicating that it is a Conflicting Rule. You can merge or overwrite the rules defined in the Defaults file with the active ones.

Defining ContentCheck Rules

You use the **Internet Protocol:ContentCheck** node in the Run-Time Setting tree, to define the rules for checking Web page content.

To define a ContentCheck rule:

1 Open the Run-Time settings and select the **Internet Protocol:ContentCheck** node.

2 Select the **Enable ContentCheck during replay** option.

3 Click **New Application** to add a new entry to the list of applications whose content to check.

4 Click **New Rule** to add rules for existing applications. Each application server may have one or more rules. Enable or disable the relevant rules by clearing or selecting the check boxes adjacent to the rule in the left pane.

5 To search for the actual text string, select **Search for Text** and specify the text for which you want to search. To obtain the best results, be as specific as possible. For example, do not use the term **Error**, rather **ASP Error** or text specific to the application.

6 To search for the text preceding and following your string, select **Search by Prefix** and specify the prefix and suffix.

7 To indicate a case sensitive search, select the **Match case** check box.

8 To set a rule as a default, indicating that it should apply to all scripts on that machine, select the rule or application and click **Set as Default**.

9 To export the rule file click **Export** and specify a save location.

10 To import a rule file, click **Import** and locate the file.

11 To remove an application or rule, select it and click **Delete**.

12 To use the default settings for all of your applications, click **Use Defaults.** A dialog box opens with a list of the applications and their default settings. You can overwrite or merge the rules if there are conflicts.

24

Run-Time Settings for Selected Protocols

After you record a Vuser script, you configure the run-time settings for the script. These settings specify how the script behaves when it runs.

This chapter includes:

- RDP Run-Time Settings on page 470
- Citrix Run-Time Settings on page 475
- .NET Environment Run-Time Settings on page 479
- Oracle NCA Run-Time Settings on page 482
- SAPGUI Run-Time Settings on page 485
- Java and EJB Run-Time Settings on page 489
- RTE Run-Time Settings on page 492
- WAP Run-Time Settings on page 494
- MMS Run-Time Settings on page 500
- Flex Run-Time Settings on page 502

RDP Run-Time Settings

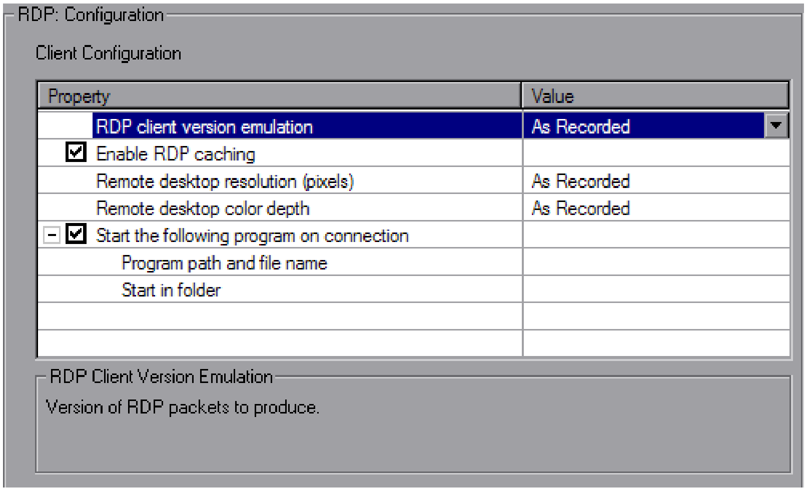
After creating an RDP Vuser script, you set the run-time settings. These settings let you control the behavior of the Vuser when running the script.

You can set the RDP-specific run-time settings in the following areas:

- RDP Configuration Run-Time Settings
- RDP Synchronization Run-Time Settings
- RDP Advanced Run-Time Settings
- RDP Agent Run-Time Settings

RDP Configuration Run-Time Settings

You use the RDP Configuration settings to set the behavior of the (virtual) RDP client.



RDP: Configuration

Client Configuration

Property	Value
RDP client version emulation	As Recorded
☑ Enable RDP caching	
Remote desktop resolution (pixels)	As Recorded
Remote desktop color depth	As Recorded
− ☑ Start the following program on connection	
Program path and file name	
Start in folder	

RDP Client Version Emulation

Version of RDP packets to produce.

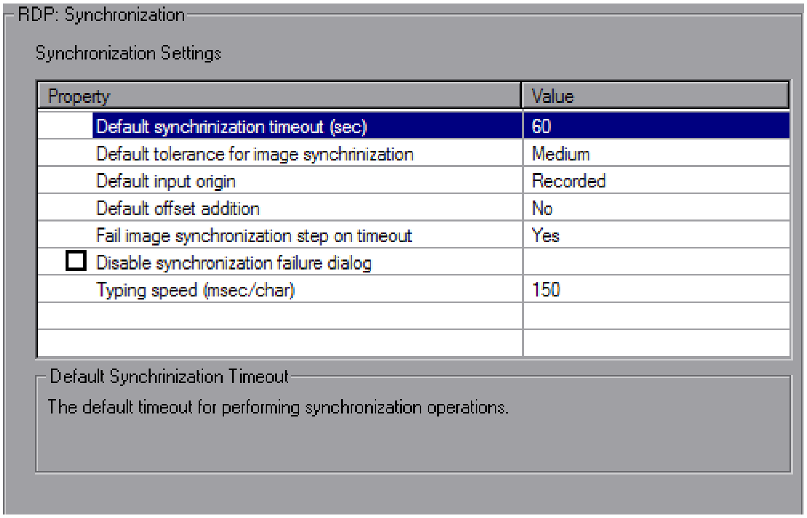
- **RDP Client Version Emulation.** The version of RDP packets to produce during replay: **As Recorded**, or a specific version number.
- **Enable RDP caching.** Support data caching orders in RDP (enabled by default).
- **Remote desktop resolution (pixels).** The size of the window in which the applications are run: **As Recorded**, or a specific size.

- **Remote desktop color depth.** The color depth settings for the replay: **As Recorded**, or a specific depth.

- **Start the following program on connection.** Open RDP connection to invoke the specified application. Specify the following information: **Program path and file name** and optionally, **Start in folder**.

RDP Synchronization Run-Time Settings

RDP Synchronization settings indicate the default timing values for various functions.



RDP: Synchronization

Synchronization Settings

Property	Value
Default synchrinization timeout (sec)	60
Default tolerance for image synchrinization	Medium
Default input origin	Recorded
Default offset addition	No
Fail image synchronization step on timeout	Yes
□ Disable synchronization failure dialog	
Typing speed (msec/char)	150

Default Synchrinization Timeout

The default timeout for performing synchronization operations.

- **Default synchronization timeout (sec)**. The time in seconds to wait for synchronization operations. Enter a value between 0 and 1000. The default value is 60.

- **Default tolerance for image synchronization.** The tolerance level for performing synchronization on images. Select one of the options: **Exact**, **Low**, **Medium** (default), or **High**. **High** has the most tolerance for changes and mismatches. **Low** requires a match of approximately 95 percent, **Medium** requires a match of approximately 85 percent, **High** requires a match of approximately 70 percent, and **Exact** requires an 100 percent match.

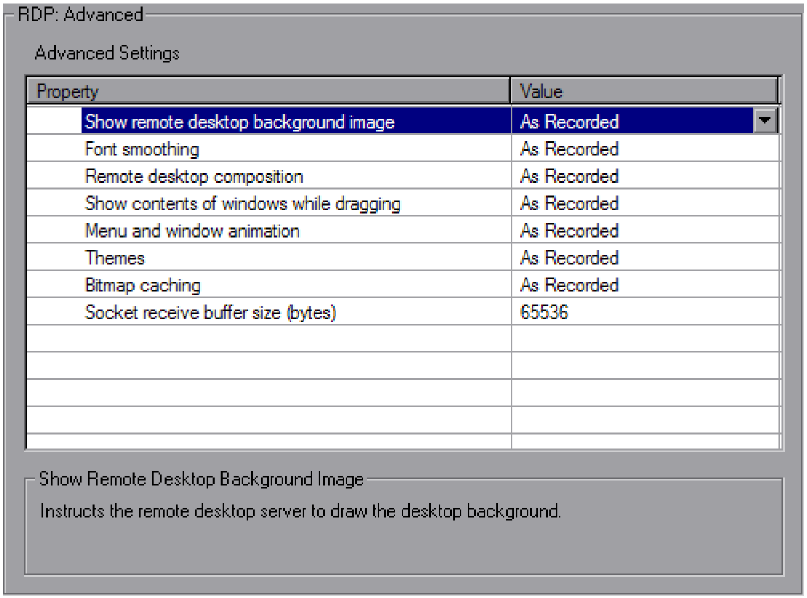
- **Default input origin.** The default origin for input operations:

  - **Recorded.** Uses coordinates for all input operations with a non-specified input origin. This is the default selection.
  - **Synched.** Adds the most recent offsets saved at one of the previous synchronization functions to the recorded coordinates of each input operation with a non-specified input origin.

- **Default offset addition.** Saves the offset of images that moved during synchronization for all subsequent functions (**No** by default).
- **Fail image synchronization step on timeout.** Instructs Vusers how to proceed when images are not found during synchronization. **Yes** (default) sets a Fail status and Vusers follow the Continue on Error setting. **No** returns an LR_NOT_FOUND flag, the step reports a warning and the script continues.
- **Disable synchronization failure dialog.** When selected, it prevents the Synchronization Failure Dialog box from opening (not selected by default).
- **Typing speed (msec/char)**. The time in milliseconds for sending consecutive characters in keyboard commands. Enter a value between 0 and 1000. The default value is 150.

RDP Advanced Run-Time Settings

You can edit advanced RDP Run-Time settings in the Advanced Settings dialog box.



RDP: Advanced

Advanced Settings

Property	Value
Show remote desktop background image	As Recorded
Font smoothing	As Recorded
Remote desktop composition	As Recorded
Show contents of windows while dragging	As Recorded
Menu and window animation	As Recorded
Themes	As Recorded
Bitmap caching	As Recorded
Socket receive buffer size (bytes)	65536

Show Remote Desktop Background Image

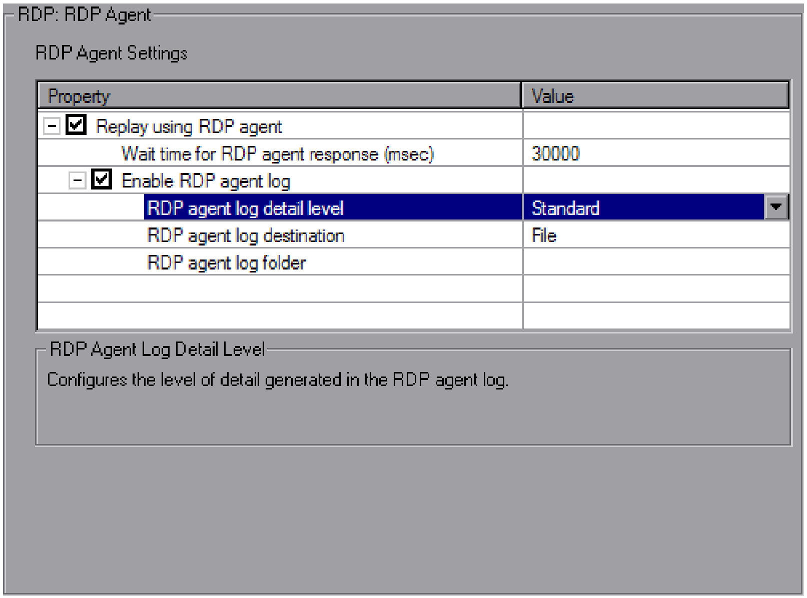
Instructs the remote desktop server to draw the desktop background.

- **Show remote desktop background image.** Allows you to run the remote desktop application without displaying the desktop background image on the remote desktop. Disabling this setting can save system resources on the remote desktop server.
- **Font smoothing.** Allows the remote desktop server to use font smoothing. Disabling this setting can save system resources on the remote desktop server.
- **Remote desktop composition.** Enables remote desktop composition.
- **Show contents of window while dragging.** Shows the contents of windows while they are being dragged. Disabling this setting can save system resources on the remote desktop server.
- **Menu and window animation.** Allows the remote desktop server to animate menus and windows. Disabling this setting can save system resources on the remote desktop server.

- **Themes.** Allows the remote desktop server to use Windows themes. Disabling this setting can save system resources on the remote desktop server.
- **Bitmap caching.** Allows the remote desktop server to use bitmap caching. Enabling this setting can save system resources on the remote desktop server.
- **Socket receive buffer size (bytes).** The number of bytes to allocate for the socket's receive buffer. If the buffer is too small, it can fill up causing the server to disconnect. If the buffer is too large, it uses more local system resources (memory).

RDP Agent Run-Time Settings

The Agent run-time settings control the way the RDP agent for Microsoft Terminal Server functions with VuGen during replay.



RDP: RDP Agent

RDP Agent Settings

Property	Value
☑ Replay using RDP agent	
Wait time for RDP agent response (msec)	30000
☑ Enable RDP agent log	
RDP agent log detail level	Standard
RDP agent log destination	File
RDP agent log folder	

RDP Agent Log Detail Level

Configures the level of detail generated in the RDP agent log.

You can set the following options:

- **Use RDP agent.** Instructs VuGen to use RDP agent during recording, then generates script using information gathered by the RDP agent during the recorded session. LoadRunner RDP agent must be installed on the server.
- **Enable RDP agent log.** Enables the RDP agent log. This feature should be used only for debugging purposes.
  - **RDP agent log detail level.** Configures the level of detail generated in the RDP agent log with **Standard** being the lowest level of detail and **Extended Debug** being the highest level of detail.
  - **RDP agent log destination.** Configures the destination of the RDP agent log data. **File** saves the log messages only on the remote server side. **Stream** sends the log messages to the Vugen machine. **FileAndStream** sends the log messages to both destinations.
  - **RDP agent log folder.** The folder path on the remote server that the RDP agent log file will be generated in. If none is specified and the agent log destination was set to **File**, the log is saved in the temp folder of the user on the server.

Citrix Run-Time Settings

After creating a Citrix Vuser script, you set the run-time settings. These settings let you control the behavior of the Vuser when running the script. Your Citrix run-time settings in the **Configuration** node should correspond to the properties of your Citrix client. These settings will influence the load on the server. To view the connection properties, select the icon representing the ICA connection in the Citrix Program Neighborhood, and select **Properties** from the right-click menu. Select the **Default Options** tab.

Note: Citrix Vusers do not support IP spoofing.

To set the General Run-time settings, see Chapter 22, "Configuring Run-Time Settings." To set the Speed Emulation properties, see Chapter 23, "Configuring Network Run-Time Settings."

You can set the Citrix-specific run-time settings in the following areas:

- Citrix Configuration Run-Time Settings
- Citrix Synchronization Run-Time Settings

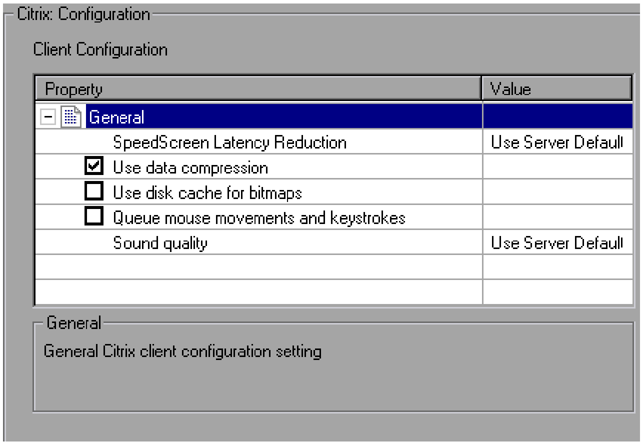
Citrix Configuration Run-Time Settings

The configuration settings relate to the screen latency, data compression, disk cache, and queuing of mouse movements.

To set the Configuration Run-Time Settings:

1 Open the Run-Time settings dialog box. Click the **Run-Time Settings** button on the VuGen toolbar, or select **Vuser** > **Run-Time Settings**.

2 Select the **Citrix:Configuration** node. Specify the **General** properties:



Citrix: Configuration

Client Configuration

Property	Value
⊟ General	
SpeedScreen Latency Reduction	Use Server Default
☑ Use data compression	
☐ Use disk cache for bitmaps	
☐ Queue mouse movements and keystrokes	
Sound quality	Use Server Default

General

General Citrix client configuration setting

Set the desired client configuration options:

- **SpeedScreen Latency Reduction**. The mechanism used to enhance user interaction when the network speed is slow. You can turn this mechanism **on** or **off**, depending on the network speed. The **auto** option turns it on or off based on the current network speed. If you do not know the network speed, set this option to **Use Server Default** to use the machine's default.

- **Use data compression.** Instructs Vusers to compress the transferred data. To enable this option, select the check box to the left of the option; to disable it, clear the check box. You should enable data compression if you have a limited bandwidth (enabled by default).
- **Use disk cache for bitmaps.** Instructs Vusers to use a local cache to store bitmaps and commonly-used graphical objects. To enable this option, select the check box to the left of the option; to disable it, clear the check box. You should enable this option if you have a limited bandwidth (disabled by default).
- **Queue mouse movements and keystrokes.** Instructs Vusers to create a queue of mouse movements and keystrokes, and send them as packets to the server less frequently. This setting reduces network traffic with slow connections. Enabling this option makes the session less responsive to keyboard and mouse movements. To enable this option, select the check box to the left of the option; to disable it, clear the check box (disabled by default).
- **Sound quality.** Specifies the quality of the sound: **Use server default**, **Sound off**, **High sound quality**, **Medium sound quality**, or **Low sound quality**. If the client machine does not have a 16-bit Sound Blaster-compatible sound card, select **Sound Off**. With sound support enabled, you will be able to play sound files from published applications on your client machine.

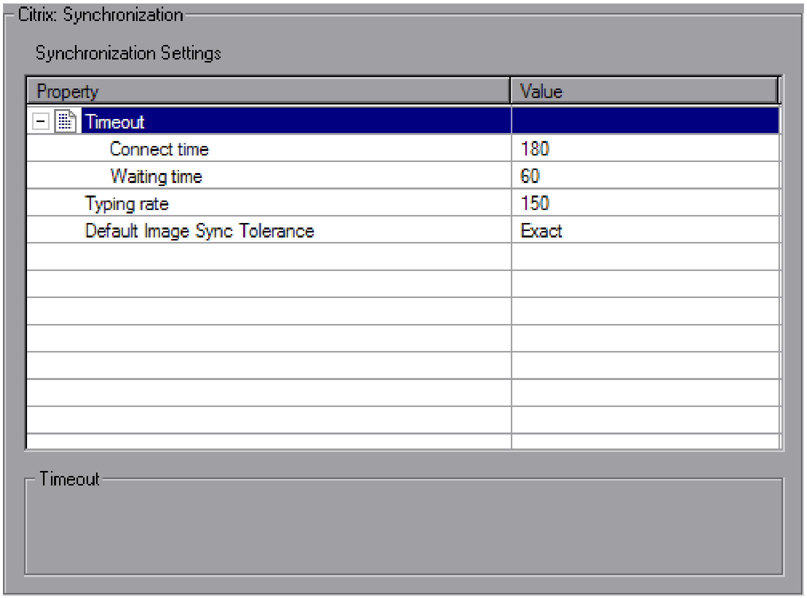
Citrix Synchronization Run-Time Settings

The synchronization settings relate to the connection and waiting times.

To set the synchronization run-time settings:

1 Click the **Run-Time Settings** button on the VuGen toolbar, or select **Vuser** > **Run-Time Settings**.

2 Select the **Citrix:Synchronization** node.



Citrix: Synchronization

Synchronization Settings

Property	Value
Timeout	
Connect time	180
Waiting time	60
Typing rate	150
Default Image Sync Tolerance	Exact

Timeout

3 Indicate the **Connect Time**, the time (in seconds) to wait idly at an established connection before exiting. The default is 180 seconds.

4 Indicate the **Waiting Time**, the time (in seconds) to wait idly at a synchronization point before exiting. The default is 60 seconds.

To set the waiting time for a specific section of the script, click **Insert** > **New Step** and insert a **Set Waiting Time** step. The new waiting time applies from the point of insertion until the end of the script or the next **Set Waiting Time** step.

5 Specify a **Typing rate**, the delay (in milliseconds) between keystrokes.

6 Indicate the **Default Image Sync Tolerance** level. This setting controls the level of equality two images must share to be considered synchronized.

- **Exact.** (default setting) Must have a 100% match.
- **Low.** Must have a 95 % match.
- **Medium.** Must have an 85% match.

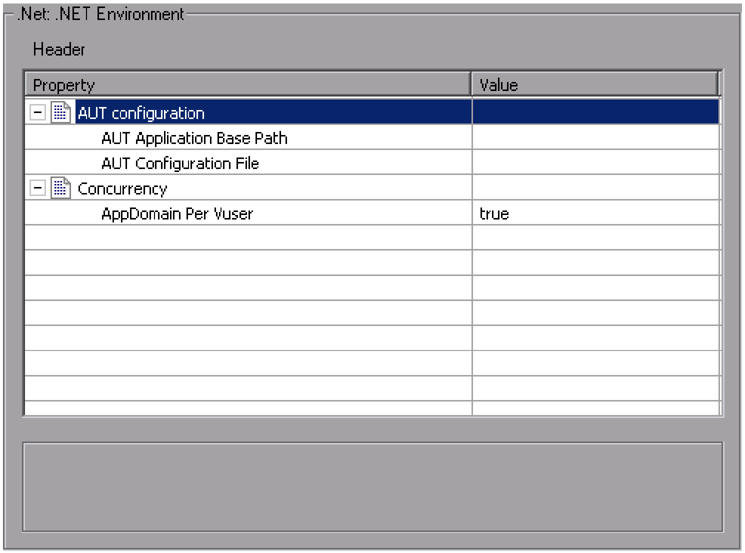
➤ **High**. Must have a 70% match.

For more information on synchronization, see Chapter 18, "Citrix Protocol" in *Volume II-Protocols*.

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

.NET Environment Run-Time Settings

Before running your Microsoft .NET Vuser script, you can specify the .NET environment settings from the Run-Time Settings dialog box.



Property	Value
⊟ AUT configuration	
AUT Application Base Path	
AUT Configuration File	
⊟ Concurrency	
AppDomain Per Vuser	true

You can also set general run-time settings for your Microsoft .NET script for configuring the pacing and iteration options. For more information, see Chapter 22, "Configuring Run-Time Settings."

AUT Configuration

AUT Application Base Path. The AUT (Application Under Test) base directory from which DLLs are loaded during replay. By default, during recording, all of the necessary DLLs are stored in the script's directory. Use this option to specify the location of any missing DLL files for the AUT. This is usually the installation path of the recorded application. Note that the AUT must be installed on the machine running the script. If you leave this box empty, VuGen uses the local script\bin directory as the application base directory during replay.

AUT Configuration File. The file name of the recorded application's configuration file. VuGen copies the AUT configuration file to the script\bin directory and loads the locally saved file. To specify a different location, use a full path. If you only specify a file name, and the file is not in the script\bin folder, VuGen loads it from the App base directory.

Concurrency

- **AppDomain Per Vuser.** Enables execution of each Vuser in a separate app domain (true by default). Running Vusers in separate App Domains enables each Vuser to execute separately without sharing static variables and prevents locking between them.

  ADO.NET providers deploy a feature called **connection pooling** which can significantly influence load test accuracy. Whenever only one app domain is used for all Vusers, connection pooling is turned on—.NET Framework keeps the database connections open and tries to reuse them when a new connection is requested. Since many Vusers are executed in the context of a single application domain, they may interfere with one another. Their behavior will not be linear and that may decrease their accuracy. The default setting, **true**, allocates a separate connection pool for each Vuser. This means that there is connection pooling in the scope of each Vuser, but the Vusers will not interfere with one another. This setting provides more accuracy, but lower scalability.

  If you disable this option, you need to manually disable connection pooling for the database.

The following table describes how to manually disable connection pooling:

Provider	Option
.NET Framework Data Provider for SQL Server	"Pooling=false" or "Pooling=no"
.NET Framework Data Provider for Oracle	"Pooling=false" or "Pooling=no"
.NET Framework Data Provider for ODBC	Connection pooling is managed by an ODBC Driver Manager. To enable or disable connection pooling, use the ODBC Data Source Administrator (found in Control Panel or the Administrative Tools folder). The **Connection Pooling** tab allows you to specify connection pooling parameters for each of the installed ODBC drivers.
.NET Framework Data Provider for OLE DB	"OLE DB Services=-2"
Oracle Data Provider for .NET	"pooling=false"
Adaptive Server Enterprise ADO.NET Data Provider	"Pooling=False"

To specify .NET resources:

1. Open the run-time setting—press F4 or select **Vuser** > **Run-Time Settings**.
2. Click on the **.NET Environment** node in the left pane.
3. Set the base folder of the DLLs in the **AUT Application Base Path** box.
4. Set the path of the recorded application in the **AUT Configuration File** box.
5. The recommended setting for **AppDomain Per Vuser** is true, the default.

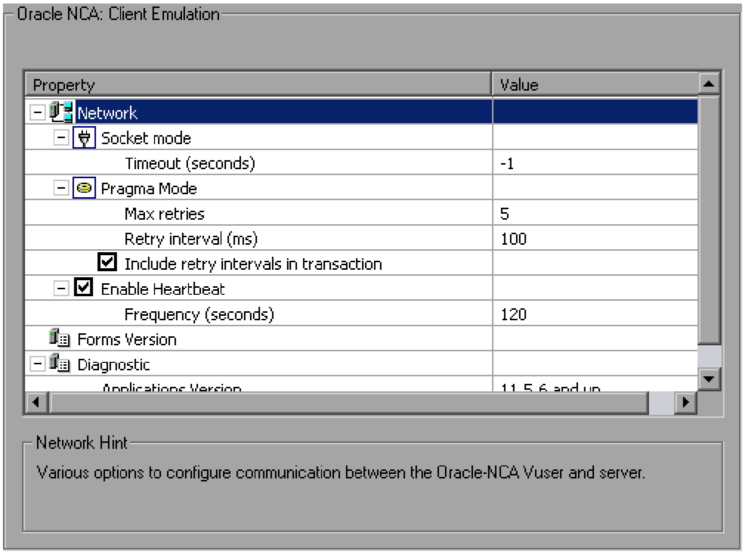
Oracle NCA Run-Time Settings

Before running your script, you can set the run-time settings to allow the script to accurately emulate a real user. For information on the general run-time settings for all protocols, such as think time, pacing, and logging, see Chapter 22, "Configuring Run-Time Settings." For network speed related settings, see Chapter 23, "Configuring Network Run-Time Settings."

The following section describes the run-time settings specific to Oracle NCA Vusers. These run-time setting allow you to indicate the communication parameters.

Configuring Oracle NCA Client Emulation Run-Time Settings

You can configure several network settings to accurately emulate an Oracle NCA client.



Oracle NCA: Client Emulation

Property	Value
− Network	
− Socket mode	
Timeout (seconds)	-1
− Pragma Mode	
Max retries	5
Retry interval (ms)	100
☑ Include retry intervals in transaction	
− ☑ Enable Heartbeat	
Frequency (seconds)	120
Forms Version	
− Diagnostic	
Applications Version	11.5.6 and up

Network Hint

Various options to configure communication between the Oracle-NCA Vuser and server.

You can set the following options:

Socket Mode

The communication to and from the client is performed on a socket level—not on the higher HTTP level.

Timeout (seconds): The time that an Oracle NCA Vuser waits for a response from the server. The default value of -1 disables the timeout and the client waits indefinitely.

Pragma Mode

In Pragma mode, communication is carried out in the Oracle-defined Pragma mode. This communication level, above the HTTP and Servlet levels, is characterized by the periodic sending of messages. In this mode, the client recognizes that the server cannot respond with data immediately. The server sends messages at given intervals until it is able to send the requested data.

- **Max Retries**. Indicates the maximum number of **IfError** messages the client will accept from the server before issuing an error. **IfError** messages are the periodic messages the server sends to the client, indicating that it will respond with the data as soon as it is able.
- **Retry Interval.** Defines the interval between retries in the case of **IfError** messages.
- **Include retry intervals in transaction.** Includes the interval between retry time, as part of the transaction duration time.

For information about recording in Pragma mode, see Chapter 45, "Oracle NCA Protocol" in *Volume II-Protocols*.

Heartbeat

You can enable or disable the heartbeat sent to the Oracle server. The heartbeat verifies that there is proper communication with the server. If you are experiencing a heavy load on the Oracle NCA server, disable the heartbeat. If you enable the heartbeat, you can set the frequency of how often heartbeat messages are sent to the server.

- **Enable Heartbeat.** By default, a heartbeat signal is sent to the server. To disable it, clear the check box.
- **Frequency.** The frequency of the heartbeat signal. The default is 120 seconds.

Forms

You can specify the version of the Oracle Forms server detected during recording.

- **Version.** Modify this setting only if the server was upgraded since the recording.

Diagnostic

This section lets you provide information about diagnostic modules for the database layer of Oracle Applications.

- **Application version.** The version of Oracle Application. This option is relevant when using Oracle Application—not a custom Oracle NCA application. It is only required when using Oracle database breakdown.

To set the Client Emulation settings:

1. Open the Run-Time Settings dialog box. Select **Vuser** > **Run-Time Settings** or click the **Run-Time Settings** button on the VuGen toolbar.
2. Select the **Oracle NCA:Client Emulation** node from the Run-Time settings tree.
3. Set the network timeout value in seconds. To instruct the client to wait indefinitely for a server response, use the default value of -1.
4. When working in Pragma mode, specify the number of retries **Max Retries**, (**IfError** messages) for the client to accept before issuing an error. The default is 5.
5. To enable the sending a a heartbeat to the Oracle NCA server, select the **Enable Heartbeat** option. In the next line, specify a frequency in seconds for the sending of the heartbeat. The default is 120 seconds.
6. Click **OK** to accept the settings and run the script.

SAPGUI Run-Time Settings

After creating and enhancing your SAPGUI Vuser script, you configure its run-time settings and run it from VuGen to check its functionality. Run-Time settings let you control the Vuser behavior during replay. You configure these settings before running the Vuser script. You can set both general and SAPGUI-specific run-time settings.

The general settings include the run logic, pacing, logging, think time, and performance preferences. For information about the general run-time settings, see Chapter 22, "Configuring Run-Time Settings."For SAPGUI-specific settings, see the following sections.

Once you configure the Run-Time settings, you save the Vuser script and run it from VuGen to verify that it runs correctly. For details about running the Vuser script as a standalone test, see Chapter 9, "Running Vuser Scripts in Standalone Mode."

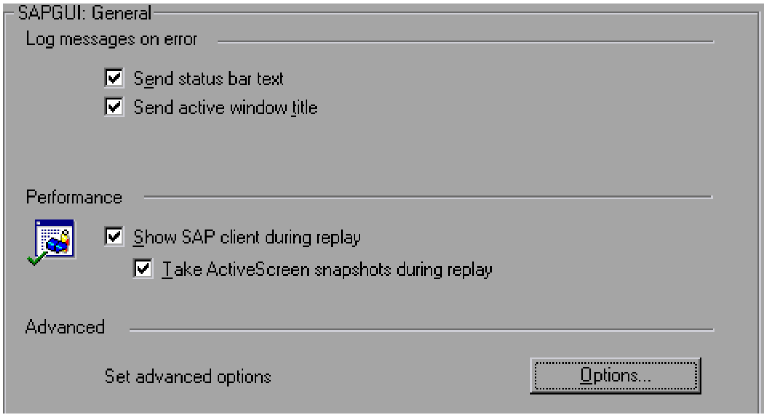
After you create a script, you integrate it into your environment: a LoadRunner scenario, Performance Center load test, or Business Process Monitor profile. For more information, see the *HP LoadRunner Controller*, *Performance Center*, or *HP Business Availability Center* documentation.

You can configure the SAPGUI specific Run-Time settings in the following areas:

- SAPGUI General Run-Time Settings
- SAPGUI Advanced Run-Time Settings

SAPGUI General Run-Time Settings

General run-time settings let you set the general settings for a SAPGUI Vuser script. VuGen uses these settings when running the script.



SAPGUI: General

Log messages on error

☑ Send status bar text

☑ Send active window title

Performance

☑ Show SAP client during replay

☑ Take ActiveScreen snapshots during replay

Advanced

Set advanced options

The Log run-time settings specify the information a Vuser sends to the Execution log whenever an error occurs.

- **Send status bar text.** Send the text from the status bar to the log file.
- **Send active window title.** Send the active window title text to the log file.

The Performance run-time settings allow you to indicate whether or not to display the SAP client during replay.

- **Show SAP Client during replay.** Shows an animation of the actions in the SAP client during replay. The benefit of displaying the user interface (UI) is that you can see how the forms are filled out and closely follow the actions of the Vuser. This option, however, requires additional resources and may affect the performance of your load test.

- **Take ActiveScreen snapshots during replay.** Captures replay snapshots with the Control ID information for all active objects. ActiveScreen snapshots differ from regular ones, in that they allow you to see which objects were recognized by VuGen in the SAPGUI client. As you move your mouse across the snapshot, VuGen highlights the detected objects. You can then add new steps to the script directly from within the snapshot. It also allows you to add steps interactively from within the snapshot for a specific object. For more information, see Chapter 46, "SAPGUI Protocol" in *Volume II-Protocols*.

Advanced options let you set a timeout for the **SAPfewgsvr.exe** process, save a snapshot on error, and configure VuGen to use SAPlogon during replay. For more information, see "SAPGUI Advanced Run-Time Settings" on page 487.

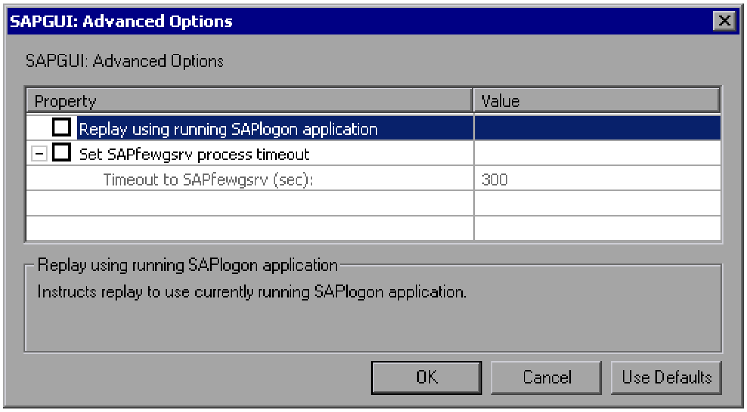
To set the SAPGUI Run-Time Settings:

1. Open the Run-Time settings dialog box. Click the **Run-Time Settings** button on the VuGen toolbar, or select **Vuser** > **Run-Time Settings**.
2. Select the **SAPGUI:General** node.
3. In the **Log messages on error** section, select one or more message sources: **Send status bar text** or **Send active window title**.
4. In the **Performance** section, select the **Show SAP client during replay** check box to show the SAPGUI user interface during replay.
5. Click **Options** to set a timeout for the **SAPfewgsvr.exe** process.

SAPGUI Advanced Run-Time Settings

Each Vuser invokes a separate **SAPfewgsvr.exe** process during test execution. In some instances, the process stays active even after the replay session has ended. You can check the Windows Task Manager to see if the process is still active.

The Advanced SAPGUI settings let you set a timeout for this application. When the timeout is reached, VuGen closes any **SAPfewgsvr** processes not previously terminated.



SAPGUI: Advanced Options

Property	Value
□ Replay using running SAPlogon application	
□ Set SAPfewgsrv process timeout	
Timeout to SAPfewgsrv (sec):	300

Replay using running SAPlogon application

Instructs replay to use currently running SAPlogon application.

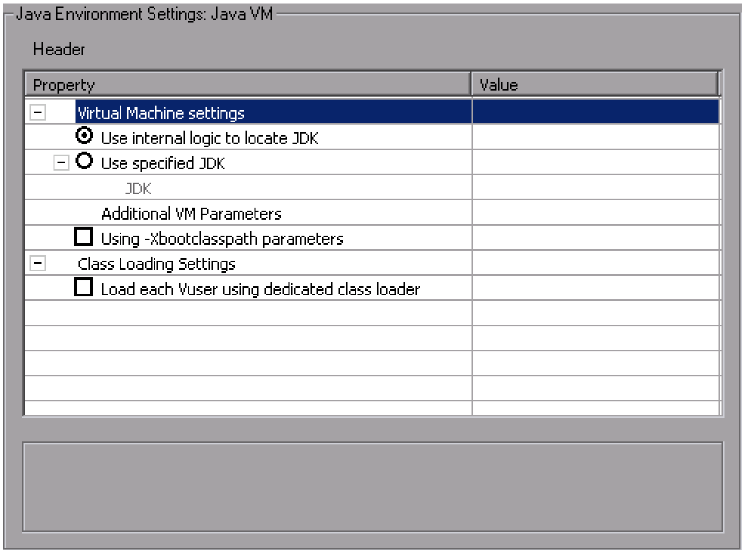
- **Replay using running SAPlogon application**. Instructs the Vusers to use the SAPlogon application that is currently running for replay.
- **Set SAPfewgsvr application timeout**. Allows you to modify the **SAPfewgsvr.exe** process timeout.
- **Timeout to SAPfewgsvr.** The **SAPfewgsvr.exe** process timeout in seconds. The default is 300 seconds.

Java and EJB Run-Time Settings

You set the Java related run-time settings through the **Java VM** options in the Run-Time Settings dialog box.

Specifying the JVM Run-Time Settings

In the Java VM section, you provide information about the Java virtual machine settings.



Java Environment Settings: Java VM

Header

Property	Value
⊟ Virtual Machine settings	
● Use internal logic to locate JDK	
⊟ ○ Use specified JDK	
JDK	
Additional VM Parameters	
□ Using -Xbootclasspath parameters	
⊟ Class Loading Settings	
□ Load each Vuser using dedicated class loader	

The following settings are available:

- **Virtual Machine settings**
  - **Use internal logic to locate JDK.** Search the PATH, registry, and Windows folder for the JDK to use during replay.
  - **Use specified JDK.** Use the JDK specified below during replay.
- **Additional VM Parameters.** Enter any optional parameters used by the virtual machine.
  - **Using Xbootclasspath parameters.** Replays the script with the Xbootclasspath /p option.
- **Class Loading Settings**

- **Load each Vuser using dedicated class loader.** Load each Vuser using a dedicated class loader. This will allow you to use a unique namespace for each Vuser and manage their resources separately.

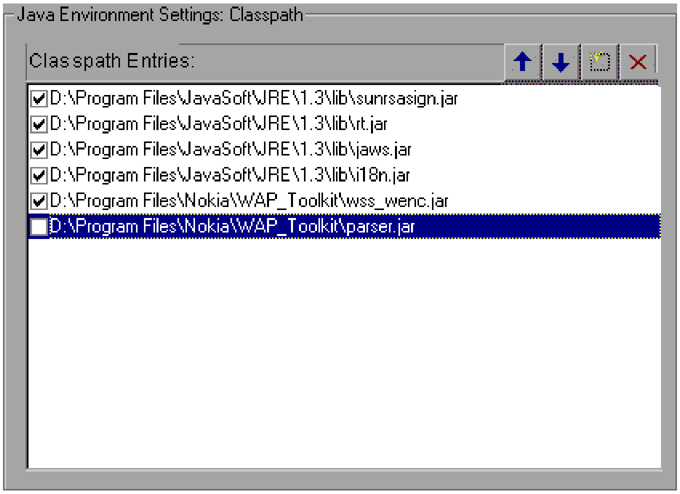
To set the Java VM run-time settings:

1. Select **Vuser** > **Run-Time Settings** and select the **Java Environment Settings:Java VM** node in the Run-Time Settings tree.
2. Select the desired **Virtual Machine settings** indicating the JDK to use for the replay.
3. To replay with the **-Xbootclasspath/p** option, select the **Using Xbootclasspath parameters** option.
4. Click **OK**.

Setting the Run-Time Classpath Options

The **ClassPath** section lets you specify the location of additional classes that were not included in the system's classpath environment variable. You may need these classes to run Java applications and insure proper replay.

You can browse for the required classes on your computer or network and disable them for a specific test. You can also manipulate the classpath entries by changing their order.



☑ D:\Program Files\JavaSoft\JRE\1.3\lib\sunrsasign.jar
☑ D:\Program Files\JavaSoft\JRE\1.3\lib\rt.jar
☑ D:\Program Files\JavaSoft\JRE\1.3\lib\jaws.jar
☑ D:\Program Files\JavaSoft\JRE\1.3\lib\i18n.jar
☑ D:\Program Files\Nokia\WAP_Toolkit\wss_wenc.jar
□ D:\Program Files\Nokia\WAP_Toolkit\parser.jar

To set the Classpath run-time settings:

1 Open the Run-Time settings (F4). Select the **Java Environment Settings:Classpath** node in the Run-Time settings tree.

2 Add a classpath to the list:

Click the **Add Classpath** button. VuGen adds a new line to the classpath list.

Type in the path and name of the **jar**, **zip** or other archive file for your class. Alternatively, click the **Browse** button to the right of the field, and locate the desired file. VuGen adds the new location to the classpath list, with an enabled status.

3 To permanently remove a classpath entry, select it and click the Delete button.

4 To disable a classpath entry for a specific test, clear the check box to the left of the entry.

5 To move a classpath entry down in the list, select it and click the Down arrow.

6 To move a classpath entry up within the list, select it and click the Up arrow.

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

RTE Run-Time Settings

You set the Terminal Emulator related run-time settings through the **RTE** node in the Run-Time Settings dialog box.

Modifying Connection Attempts

The **TE_connect** function is generated by VuGen when you record a connection to a host. When you replay an RTE Vuser script, the **TE_connect** function connects the terminal emulator to the specified host. If the first attempt to connect is not successful, the Vuser retries a number of times to connect successfully. Details of each connection are recorded in the report file **output.txt**.

To set the maximum number of times that a Vuser will try to connect, enter a number in the **Maximum number of connection attempts** box in the RTE Run-Time settings.

By default, a Vuser will try to connect 5 times.

For more information about the **TE_connect** function, see the *Online Function Reference* (**Help** > **Function Reference**).

Specifying an Original Device Name

In certain environments, each session (Vuser) requires a unique device name. The **TE_connect** function generates a unique 8-character device name for each Vuser, and connects using this name. To connect using the device name (that is contained within the com_string parameter of the **TE_connect** function), select the **Use original device name** option in the RTE Run-Time settings.

Note: The original device name setting applies to IBM block-mode terminals only.

By default, Vusers use original device names to connect.

For details about the **TE_connect** function, see the *Online Function Reference* (**Help** > **Function Reference**).

Setting the Typing Delay

The delay setting determines how Vusers execute **TE_type** functions.

To specify the amount of time that a Vuser waits before entering the first character in a string, enter a value in the First key box, in milliseconds.

To specify the amount of time that a Vuser waits between submitting successive characters, enter a value in the Subsequent keys box, in milliseconds.

If you enter zero for both the first key and the subsequent key delays, the Vuser will send characters as a single string, with no delay between characters.

You can use the **TE_typing_style** function to override the Delay settings for a portion of a Vuser script.

For details about the **TE_type** and **TE_typing_style** functions, see the *Online Function Reference* (**Help** > **Function Reference**).

Configuring the X-System Synchronization

RTE Vuser scripts use the **TE_wait_sync** function for synchronization. You can set a timeout value and a stable-time value that VuGen applies to all **TE_wait_sync** functions. For details about the **TE_wait_sync** function, see the *Online Function Reference* (**Help** > **Function Reference**).

Timeout

When you replay a **TE_wait_sync** function, if the system does not stabilize before the synchronization timeout expires, the **TE_wait_sync** function returns an error code. To set the synchronization timeout, enter a value (in seconds) in the Timeout section of the RTE Run-Time settings.

The default timeout value is 60 seconds.

Stable Time

After a Vuser executes a **TE_wait_sync** function, the Vuser waits until the terminal is no longer in the X-SYSTEM mode. After the terminal returns from the X-SYSTEM mode, the Vuser still monitors the system for a short time. This makes sure that the terminal has become stable, that is, that the system has not returned to the X-SYSTEM mode. Only then does the **TE_wait_sync** function terminate.

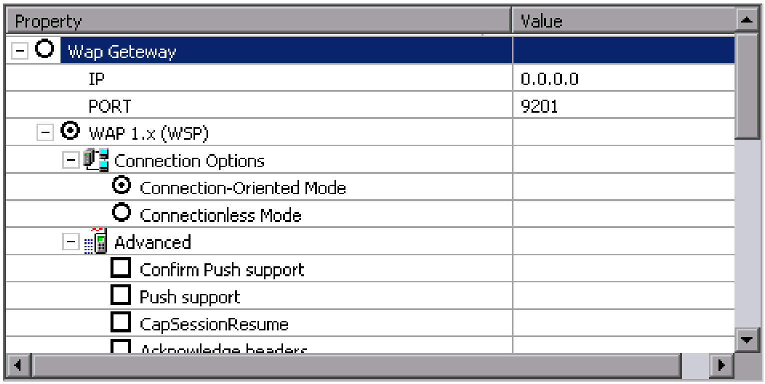
To set the time that a Vuser continues to monitor the system after the system has returned from the X-SYSTEM mode, enter a value (in milliseconds) in the Stable time box of the RTE Run-Time settings.

The default stable time is 1000 milliseconds.

WAP Run-Time Settings

Configuring Gateway Options

You use the **WAP:Gateway** node in the Run-Time Settings tree to set the Gateway settings.



Property	Value
○ Wap Geteway	
IP	0.0.0.0
PORT	9201
● WAP 1.x (WSP)	
Connection Options	
● Connection-Oriented Mode	
○ Connectionless Mode	
Advanced	
□ Confirm Push support	
□ Push support	
□ CapSessionResume	
□ Acknowledge headers	

Connection Options

The connection options specify the method that the Vuser uses to connect to the WAP gateway.

WAP Gateway. Run the Vusers accessing a Web server via a WAP Gateway.

HTTP Direct. Run the Vusers run in HTTP mode, accessing a Web server directly.

Note: If you select the HTTP Direct connection mode, the remaining WAP Gateway options are not applicable.

Gateway Settings

If the Vusers connect through a gateway, the IP, Port, and WAP Versions options specify the Gateway connection.

IP. Specify the IP address of the gateway.

Port. Specify the port of the gateway. When running your Vusers through a WAP gateway, VuGen automatically sets default port numbers, depending on the selected mode. However, you can customize the settings and specify a custom IP address and port for the gateway.

Wap version. Select the appropriate WAP version, **1.x (WSP)** or **2.0 (HTTP proxy)**. If you recorded in WAP 1.x (WSP), you can run the Vuser in either 1.x (WSP), or 2.0 (HTTP proxy) mode. If you recorded in WAP 2.0 (HTTP proxy), then you can only run the Vuser in the same mode.

If you are running the script in WAP 1.x (WSP), you can specify several connection and advanced options.

Gateway Connection Mode

The connection mode settings apply to WAP version 1.x (WSP) connections.

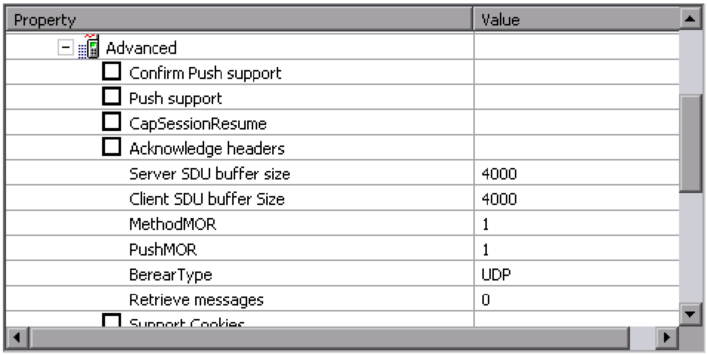
Connection-oriented Mode. Set the connection mode for the WSP session to Connection-Oriented.

Connectionless Mode. Set the connection mode for the WSP session to Connectionless.

Enable security. Enable a secure connection to the WAP gateway.

Advanced Gateway Options

Expand the **Advanced** option in the Gateway node to configure the WAP Capabilities and other advanced gateway options.



Property	Value
⊟ Advanced	
□ Confirm Push support	
□ Push support	
□ CapSessionResume	
□ Acknowledge headers	
Server SDU buffer size	4000
Client SDU buffer Size	4000
MethodMOR	1
PushMOR	1
BerearType	UDP
Retrieve messages	0
□ Support Cookies	

- **Confirm Push support.** In CO mode, if a push message is received, this option instructs the Vuser to confirm the receipt of the message (disabled by default). For more information, see Chapter 58, "Wireless Protocols" in *Volume II-Protocols*.
- **Push support.** Enables push type messages across the gateway (disabled by default).
- **CAPSessionResume.** Enables requests for session suspend or resume.
- **Acknowledge headers.** Returns standard headers that provide information to the gateway (disabled by default).
  - **Server SDU buffer size.** The largest transaction service data unit that may be sent to the server during the session (4000 by default).
  - **Client SDU buffer size.** The largest transaction service data unit that may be sent to the client during the session (4000 by default).
  - **MethodMOR.** The number of outstanding methods that can occur simultaneously.
  - **PushMOR.** The number of outstanding push transactions that can occur simultaneously.
  - **BearerType.** The type of bearer used as the underlying transport.

  - **Retrieve messages.** When a push messages is received, this option instructs the Vuser to retrieve the message data from the URL indicated in the push message (disabled by default).

- **Support Cookies.** Provide support for saving and retrieving cookies (disabled by default).

- **WTP Segmentation and Reassembly.** Enables segmentation and reassembly (SAR) in WTP, Wireless Transport Protocol. (True by default).

  - **WTP Retransmission Time.** The time in seconds that the WTP layer waits before resending the PDU if it did not receive a response. (5000 by default).

- **WTLS Abbreviated Handshake.** Use an abbreviated handshake instead of a full one, when receiving a redirect message. (False by default).

- **WTLS Deffie Hellman.** Use the Deffie Hellman encryption scheme for WTLS (Wireless Transport Layer Security) instead of the default scheme, RSA. (False by default).

  - **WTLS Deffie Hellman identifier.** An identifier for the Deffie Hellman encryption scheme. This identifier is required for the abbreviated handshake with the Operwave gateway that uses the Deffie Hellman encryption scheme.

  - **Network MTU Size.** the maximum size in bytes, of the network packet. (4096 by default).

Setting the Gateway Options

The following section describes the procedure for setting the WAP Gateway options.

To set the WAP gateway options:

1. Click the **Run-Time Settings** button or select **Vuser** > **Run-Time Settings** to display the Run-Time Settings dialog box. Select the **WAP:Gateway** node.

2. To replay the script in WSP mode (not HTTP), select **WAP Gateway**.

3. Specify an IP address and port for the gateway. You can also use the default port indicated by VuGen.

4. Select a WAP version: **WAP 1.x (WSP)** or **WAP 2.0 (HTTP)**.

5 For WAP 1.x (WSP), select a connection mode—**Connection-oriented** or **Connectionless**. To indicate a secure connection mode, select the **Enable Security** option.

6 For WAP 1.x (WSP), expand the **Advanced** node to set the client capabilities and other advanced gateway options. For more information about the Advanced options, see above.

Configuring Radius Connection Data

RADIUS (Remote Authentication Dial-In User Service) is a client/server protocol and software that enables remote access servers to communicate with a central server to authenticate dial-in users and authorize their access to the requested system or service.

RADIUS allows a company to maintain user profiles in a central database that all remote servers can share. It provides better security, allowing a company to set up a policy that can be applied at a single administered network point. Using a central service makes it easier to track usage for billing and store network statistics.

RADIUS has two sub-protocols:

- **Authentication.** Authorizes and controls user access.
- **Accounting.** Tracks usage for billing and for keeping network statistics.

In VuGen, the RADIUS protocol is only supported for WSP replay for both Radius sub-protocols—authentication and accounting.

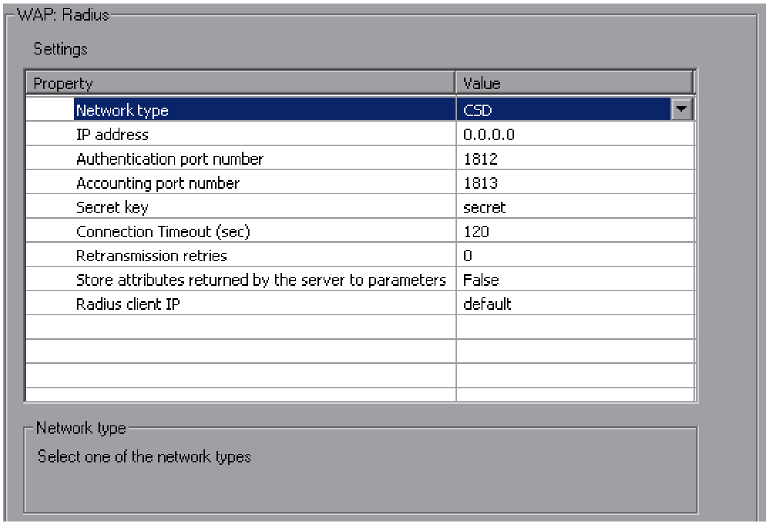
You supply the dial-in information in the Run-Time Settings' Radius node:

Property	Value
Network Type	Accounting network type: GPRS (General Packet Radio Service) or CSD (Circuit-Switched Data).
IP Address	IP address of the Radius server.
Authentication port number	Authentication port of the Radius server.
Accounting port number	Accounting port of the Radius server.

Secret Key	The secret key of the Radius server.
Connection Timeout (sec)	The time in seconds to wait for the Radius server to respond. The default is 120 seconds.
Retransmission retries	The number of times to retry after a failed transmission. The default is 0.
Store attributes returned by the server to parameters	Allow Vusers to save attributes returned by the server as parameters, which can be used at a later time. The default is **False**.
Radius client IP	Radius packets source IP, usually used to differentiate between packets transmitted on different NIC cards on a single Load Generator machine.

To set the WAP Radius options:

1 Click the **Run-Time Settings** button or select **Vuser** > **Run-Time Settings** to display the Run-Time Settings dialog box. Click the **Radius** node.



WAP: Radius

Settings

Property	Value
Network type	CSD
IP address	0.0.0.0
Authentication port number	1812
Accounting port number	1813
Secret key	secret
Connection Timeout (sec)	120
Retransmission retries	0
Store attributes returned by the server to parameters	False
Radius client IP	default

Network type

Select one of the network types

2 Select an accounting **Network type**: GPRS (General Packet Radio Service) or CSD (Circuit-Switched Data).

3 Enter the **IP address** of the Radius server in dot form.

4 Enter the **Authentication Port number** and **Accounting Port number** of the Radius server.

5 Type in the **Secret key** for Radius or Accounting Authentication.

6 Enter a **Connection Timeout** value.

7 Specify the number of **Retransmission retries**.

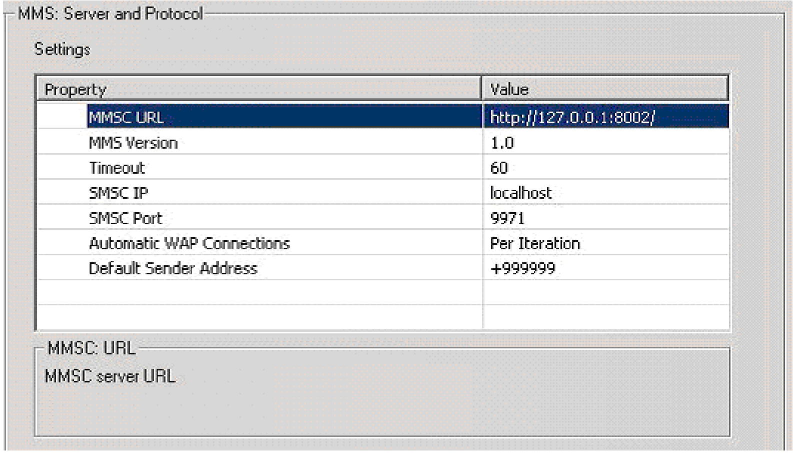
8 Specify whether you want VuGen to **store attributes returned by the server to parameters**.

9 Click **OK** to accept the settings and close the dialog box.

MMS Run-Time Settings

Before running your script, you can set the run-time settings to allow the script to accurately emulate a real user.

The following section describes the run-time settings specific to MMS (Multimedia Messaging Service) Vusers. These run-time setting allow you to configure the server and protocol settings.



MMS: Server and Protocol

Settings

Property	Value
MMSC URL	http://127.0.0.1:8002/
MMS Version	1.0
Timeout	60
SMSC IP	localhost
SMSC Port	9971
Automatic WAP Connections	Per Iteration
Default Sender Address	+999999

MMSC: URL

MMSC server URL

You can set the following options:

- **MMSC URL.** The URL of the MMSC (Multimedia Messaging Center) server.

- **MMS Version.** The version of the MMS protocol used by the script.
- **Timeout (seconds).** The time that the server waits for incoming messages. The default value is 60 seconds.
- **SMSC IP.** The IP address of the SMSC server used for sending MMS notifications over SMPP.
- **SMSC Port.** The IP port of the SMSC server used for sending MMS notifications over SMPP.
- **Automatic WAP Connections.** Defines when to connect and disconnect from a WAP gateway. This setting is only relevant when a WAP gateway is used. The possible values are:
  - **Per Iteration.** Connect at the beginning of each iteration and disconnect at the end of each iteration. (default)
  - **Per Send or Receive.** Connect and disconnect at the beginning and end of each message.
  - **None.** Do not use automatic WAP connections.
- **Default Sender address.** The default address sent in the Sender header. The default is +999999.

To set the MMS Server and Protocol settings:

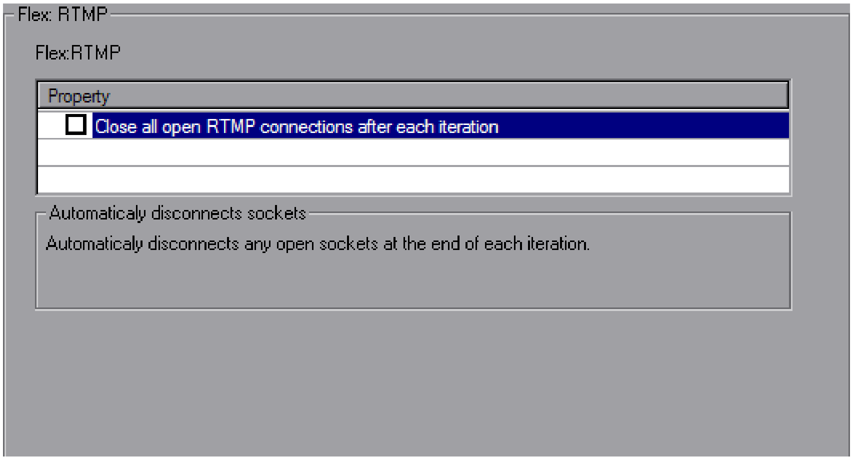
1. Open the Run-Time Settings dialog box. Select **Vuser** > **Run-Time Settings** or click the **Run-Time Settings** button on the VuGen toolbar.
2. Select the **MMS:Server and Protocol** node from the Run-Time settings tree.
3. Select the desired values as explained above.
4. Select **General:Miscellaneous** from the Run-Time settings tree.
5. Under Multithreading, select **Run Vuser as a process.**

6 Click **OK** to accept the settings and run the script.

Flex Run-Time Settings

Before running your script, you can set the run-time settings to allow the script to accurately emulate a real user.

The following section describes the run-time settings specific to Flex Vusers.



Flex: RTMP

Flex:RTMP

Property
□ Close all open RTMP connections after each iteration

Automaticaly disconnects sockets

Automaticaly disconnects any open sockets at the end of each iteration.

You can set the following option:

➤ **Close all open RTMP connections after each iteration.** Automatically disconnects any open RTMP connections at the end of each iteration.

Part 5

Information for Advanced Users

25

Creating Vuser Scripts in Visual Studio

You can create a Vuser script template in Visual Studio using Visual C or Visual Basic. You compile it as you would a regular C or Visual Basic program.

This chapter includes:

- About Creating Vuser Scripts in Visual Studio on page 505
- Creating a Vuser Script with Visual C on page 506
- Creating a Vuser Script with Visual Basic on page 508
- Configuring Runtime Settings and Parameters on page 510

About Creating Vuser Scripts in Visual Studio

There are several ways to create Vuser scripts: through VuGen or a development environment such as Visual Studio.

VuGen You can use VuGen to create Vuser script that run on Windows or UNIX platforms by recording or by manually programming within the VuGen editor. You create the script in a Windows environment and run it in either Windows or UNIX—recording is not supported on UNIX.

Visual Studio For users working with Visual Studio, you can program in Visual Basic, C or C++. The programs must be compiled into a dynamic link library (dll).

This chapter describes how to develop a Vuser script through programming within the Visual C and Visual Basic environments. In these environments, you develop your Vuser script within your development application, while importing the Vuser API libraries.

You can also program a Vuser script from within the VuGen editor, incorporating your application's libraries or classes. Programming within VuGen is available for C, Java, Visual Basic, VBScript, and JavaScript. For more information, see "Programming a Script in the VuGen Editor" in *Volume II-Protocols*.

To create a Vuser script through programming, you can use a VuGen template as a basis for a larger Vuser script. The template provides:

- correct program structure
- Vuser API calls
- source code and makefiles for creating a dynamic library

After creating a basic Vuser script from a template, you can enhance the script to provide run-time information and statistics. For more information, see Chapter 6, "Enhancing Vuser Scripts."

An online C reference of the common functions used in Vuser scripts, are included in the *Online Function Reference* (**Help** > **Function Reference**).

Creating a Vuser Script with Visual C

Please note that you can create Vuser scripts using Visual C version 6.0 or higher.

To create a Vuser script with Visual C:

1 In Visual C, create a new project - dynamic link library (dll). Select **File** > **New** and click the Projects tab.

2 In the Wizard, select *empty dll*.

3 Add the following files to the project:

   - A new *cpp* file with 3 exported function: *init*, *run*, *end* (the names may be customized).

- The library file lrun50.lib (located in the <lr installation dir>/lib).

4 In the project settings change the following:

- Select the C/C++ tab and select **Code generation** (Category) > **Use Run Time library** (List). Change it to: **Multithreaded dll.**
- Select the C/C++ tab and select **Preprocessor** (Category) > **Preprocessor definitions** (edit field) Remove _DEBUG.

5 Add code from your client application, or program as you normally would.

6 Enhance your script with Vuser API functions. For example, **lr_output_message** to issue messages, **lr_start_transaction** to mark transactions, and so forth. For more information, see the General functions in the *Online Function Reference* (**Help** > **Function Reference**).

7 Build the project. The output will be a DLL.

8 Create a directory with the same name as the DLL and copy the DLL to this directory.

9 In the **lrvuser.usr** file in the *Template* directory, Update the USR file key *BinVuser* with the DLL name: BinVuser=<*DLL_name*>.

In the following example, the lr_output_messsage function issues messages indicating which section is being executed. The lr_eval_string function retrieves the name of the user. To use the following sample, verify that the path to the Vuser API include file, lrun.h is correct.

```
#include "c:\lrun_5\include\lrun.h"

extern "C" {
int __declspec(dllexport) Init (void *p)
{
    lr_output_message("in init");
return 0;
}

int __declspec(dllexport) Run (void *p)
{
    const char *str = lr_eval_string("<name>");
    lr_output_message("in run and parameter is %s", str);
return 0;
}

int __declspec(dllexport) End (void *p)
{
    lr_output_message("in end");
return 0;
}
} //extern C end
```

Creating a Vuser Script with Visual Basic

To create a Vuser in Visual Basic:

1 In Microsoft Visual Basic, create a new project. Select **File** > **New Project**.

2 Select **LoadRunner Virtual User**. A new project is created with one class and a template for a Vuser.

3 Save the project before you continue to program. Chose **File** > **Save Project**.

4. Open the Object Browser (View menu). Select the LoadRunner Vuser library and double-click on the Vuser Class module to open the template. The template contains three sections, Vuser_Init, Vuser_Run, and Vuser_End.

```
Option Explicit

Implements Vuser

Private Sub Vuser_Init()
'Implement the Vuser initialization code here
End Sub

Private Sub Vuser_Run()
'Implement the Vuser main Action code here
End Sub

Private Sub Vuser_End()
'Implement the Vuser termination code here
End Sub
```

5. Add code from your client application, or program as you normally would.
6. Use the Object Browser to add the desired VuGen elements to your code, such as transactions, think time, rendezvous, and messages, using the object browser.
7. Enhance your program with run-time settings and parameters. For more information, see "Configuring Runtime Settings and Parameters" on page 510.
8. Build the Vuser script: select **File** > **Make** *project_name*.dll.

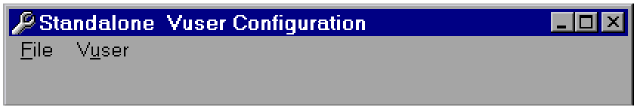
The project is saved in the form of a Vuser script (.usr). The script resides in the same directory as the project.

Configuring Runtime Settings and Parameters

After you create the DLL for your script, you create a script (*.usr*) and configure its settings. The *lrbin.bat* utility provided with VuGen lets you define parameters and configure runtime settings for scripts created with Visual C and Basic. This utility is located in the *bin* directory of the product installation.

To configure runtime settings and parameterize scripts:

1 In the product's *bin* directory, double-click on *lrbin.bat*. The Standalone Vuser Configuration dialog box opens.



2 Select **File** > **New**. Specify a script name for the *usr* file. The script name must be identical to the name of the directory to which you saved the DLL.

3 Select **Vuser** > **Advanced** and enter the DLL name in the Advanced dialog box.

4 Select **Vuser** > **Run-time Settings** to define run-time settings. The Run-time Settings dialog box is identical to that displayed in the VuGen interface. For more information, see Chapter 22, "Configuring Run-Time Settings."

5 Select **Vuser** > **Parameter List** to define parameters for your script. The Parameter dialog boxes are identical to those in VuGen. For more information, see Chapter 13, "Working with VuGen Parameters."

Test the script by running it in standalone mode. Select **Vuser** > **Run Vuser**. The Vuser execution window appears while the script runs.

6 Select **File** > **Exit** to close the configuration utility.

26

Programming with the XML API

You can create Vuser scripts that support the complete XML structure. VuGen provides functions that allow you to query and manipulate the XML data.

This chapter includes:

- About Programming with the XML API on page 512
- Understanding XML Documents on page 513
- Using XML Functions on page 514
- Specifying XML Function Parameters on page 517
- Working with XML Attributes on page 519
- Structuring an XML Script on page 519
- Enhancing a Recorded Session on page 521
- Using Result Parameters on page 526

The following information applies primarily to Web, Web Services, and Wireless Vuser scripts.

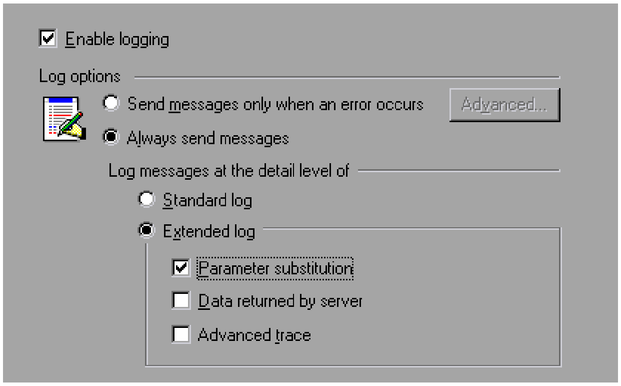
About Programming with the XML API

VuGen's support for XML allows you to dynamically work with XML code and retrieve the values during test execution. Follow these steps in creating an effective XML script:

- Record a script in the desired protocol, usually Web, Web Services, or Wireless.
- Copy the XML structures into your script.
- Add XML functions from the LR API in order to retrieve dynamic data and the XML element values.

The LR API uses XPath, the XML Path language to manipulate the text in an XML document.

You can instruct VuGen to display the output values of XML elements in the Execution log window using the Run-Time settings. VuGen displays the line numbers, the number of matches, and the value. To allow the displaying of values, you need to enable parameter substitution. In the Run-Time settings, open the **General:Log** node, select **Extended log**, and select **Parameter Substitution**. For more information, see Chapter 22, "Configuring Run-Time Settings."



☑ Enable logging

Log options

○ Send messages only when an error occurs

● Always send messages

Log messages at the detail level of

○ Standard log

● Extended log

☑ Parameter substitution

□ Data returned by server

□ Advanced trace

All Vuser API XML functions return the number of matches successfully found, or zero for failure.

Understanding XML Documents

XML, or Extensible Markup Language, is a markup language that you can use to create your own custom tags. Using these tags, you give a meaning to the text between the tags. This stands in contrast to standard HTML tags such as H1, P, DIV, and so on, which cannot be customized and do not indicate the content of the text.

XML documents consist of trees with many nodes and branches. There are three common terms used that describe the parts of an XML document: tags, elements, and attributes. The following example illustrates these terms:

```
<acme_org>
    <accounts_dept>
        <employee type=’PT’>
            <name>John Smith</name>
            <cubicle>227</cubicle>
            <extension>2145</extension>
        </employee>
    </accounts_dept>
    <engineering_dept>
        <employee type=’PT’>
            <name>Sue Jones</name>
            <extension>2375</extension>
        </employee>
    </engineering_dept>
</acme_org>
```

A *tag* is the text between the left and right angle brackets. <acme_org>, <employee> and <name> are examples of tags. There are starting tags, such as <name>, and ending tags, such as </name>. The above XML fragment describes the Acme organization with two employees, John Smith and Sue Jones.

An *element* is the starting tag, ending tag, and everything in between. In the sample above, the <employee> element contains three child elements: <name>, <cubicle>, and <extension>.

An *attribute* is a name-value pair inside the starting tag of an element. In this example, **type=’PT’** is an attribute of the <employee> element;

In the above example, the tag *name* is an element of *employee*. Each element has a value. An example of a *name* element's value is the string "John Smith"

Using XML Functions

The next sections provide examples of how to work with data in an XML tree. Certain functions allow you to retrieve information, and others let you write information to an XML tree. These examples use the following XML tree containing the names and extensions of several employees in the Acme organization.

```
<acme_org>
    <accounting_dept>
        <employee type='PT'>
            <name>John Smith</name>
            <extension>2145</extension>
        </employee>
    </accounting_dept>
    <engineering_dept>
        <employee type='PT'>
            <name>Sue Jones</name>
            <extension>2375</extension>
        </employee>
    </engineering_dept>
</acme_org>
```

Reading Information from an XML Tree

The functions which read information from an XML tree are:

lr_xml_extract	Extracts XML string fragments from an XML string.
lr_xml_find	Performs a query on an XML string.
lr_xml_get_values	Retrieves values of XML elements found by a query.

To retrieve a specific value through a query, you specify the tags of the parent and child nodes in a path format.

For example, to retrieve an employee name in the Accounting department, use the following string:

```
lr_xml_get_values("XML={XML_Input_Param}",
"ValueParam=OutputParam",
"Query=/acme_org/accounting_dept/employee/name",
LAST);
```

The Execution log window (with Extended logging enabled) shows the output of this function:

```
Output:
Action.c(20): "lr_xml_get_values" was successful, 1 match processed
Action.c(25): Query result = John Smith
```

Writing to an XML Structure

The functions which write values to an XML tree are:

lr_xml_delete	Deletes fragments from an XML string.
lr_xml_insert	Inserts a new XML fragment into an XML string.
lr_xml_replace	Replaces fragments of an XML string.
lr_xml_set_values	Sets the values of XML elements found by a query.
lr_xml_transform	Applies Extensible Stylesheet Language (XSL) transformation to XML data.

The most common *writing* function is **lr_xml_set_values** which sets the values of specified elements in an XML string. The following example uses **lr_xml_set_values** to change the phone extensions of two *employee* elements in an XML string.

First, we save the XML string to a parameter called *XML_Input_Param*. We want two values to be matched and substituted, so we prepare two new parameters, *ExtensionParam_1* and *ExtensionParam_2*, and set their values to two new phone extensions, 1111 and 2222.

lr_xml_set_values contains the argument "ValueName=ExtensionParam", which picks up the values of ExtensionParam_1 and ExtensionParam_2. The current extensions of the two employees are substituted with the values of these parameters, 1111 and 2222. The value of OutputParam is then evaluated proving that the new phone extensions were in fact substituted.

```
Action() {

    int i, NumOfValues;
    char buf[64];

    lr_save_string(xml_input, "XML_Input_Param"); // Save input as parameter
    lr_save_string("1111", "ExtensionParam_1");
    lr_save_string("2222", "ExtensionParam_2");

    lr_xml_set_values("XML={XML_Input_Param}",
        "ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
        "SelectAll=yes", "Query=//extension", LAST);

    NumOfValues= lr_xml_get_values("XML={NewXmlParam}",
        "ValueParam=OutputParam", "Query=//extension",
        "SelectAll=yes", LAST);

    for (i = 0; i < NumOfValues; i++) {/* Print the multiple values of MultiParam */

        sprintf(buf, "Retrieved value %d : {OutputParam_%d}", i+1, i+1);
        lr_output_message(lr_eval_string(buf));
    }

    return 0;
}
Output:
Action.c(40): Retrieved value 1: 1111
Action.c(40): Retrieved value 2: 2222
```

Specifying XML Function Parameters

Most XML API functions require that you specify the **XML element** and a **query**. You can also indicate if you want to retrieve all results or a single one.

Defining the XML Element

For defining the XML element to query, you can specify a literal string of the XML element, or a parameter that contains the XML. The following example shows the XML input string defined as a literal string:

```
"XML=<employee>JohnSmith</employee>"
```

Alternatively, the **XML** string can be a parameter containing the XML data. For example:

```
"XML={EmployeeNameParam}"
```

Querying an XML Tree

Suppose you want to find a value within an XML tag, for example, an employee's extension. You formulate a query for the desired value. The query indicates the location of the element and which element you want to retrieve or set. The path that you specify limits the scope of the search to a specific tag. You can also search for all elements of a specific type under all nodes below the root.

For a specific path, use "Query=/*full_xml_path_name*/*element_name*"

For the same element name under all nodes, use "Query=//*element_name*"

In the VuGen implementation of XML functions, the scope of a query is the entire XML tree. The tree information is sent to the Vuser API functions as the value of the *xml* argument.

Multiple Query Matching

When you perform a query on an XML element, by default VuGen returns only the first match. To retrieve multiple values from a query, you specify the "SelectAll=yes" attribute within your functions. VuGen adds a suffix of *_index* to indicate multiple parameters. For example, if you defined a parameter by the name *EmployeeName*, VuGen creates *EmployeeName_1*, *EmployeeName_2*, *EmployeeName_3*, and so on.

```
lr_xml_set_values("XML={XML_Input_Param}",
"ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
"SelectAll=yes", "Query=//extension", LAST);
```

With functions that *write* to a parameter, the values written to the parameter can then be evaluated. For example, the following code retrieves and prints multiple matches of a query:

```
NumOfValues = lr_xml_get_values("Xml={XmlParam}", "Query=//name",
        "SelectAll=yes", "ValueParam=EmployeeName", LAST);
```

For functions that *read* from parameters, the values of the parameters must be pre-defined. The parameter must also use the convention *ParamName_IndexNumber*, for example *Param*_1, *Param*_2, *Param*_3, and so on. This collection of parameters is also known as a parameter set.

In the following example, lr_xml_set_values reads values from the parameter set and then uses those values in the XPath query. The parameter set that represents the employee extensions, is called ExtensionParam. It has two members: ExtensionParam_1 and ExtensionParam_2. The **lr_xml_set_values** function queries the XML input string and sets the value of the first match to 1111 and the second match to 2222.

```
lr_save_string("1111", "ExtensionParam_1");
lr_save_string("2222", "ExtensionParam_2");

lr_xml_set_values("XML={XML_Input_Param}",
    "ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
    "SelectAll=yes", "Query=//extension", LAST);
```

Working with XML Attributes

VuGen contains support for attributes. You can use a simple expression to manipulate attributes of XML elements and nodes, just as you can manipulate the elements themselves. You can modify the desired attribute or only attributes with specific values.

In the following example, **lr_xml_delete** deletes the first cubicle element with the name attribute.

```
lr_xml_delete("Xml={ParamXml}",
                    "Query=//cubicle/@name",
                        "ResultParam=Result",
                    LAST
    );
```

In the next example, **lr_xml_delete** deletes the first cubicle element with a name attribute that is equal to Paul.

```
lr_xml_delete("Xml={ParamXml}",
        "Query=//cubicle/@name="Paul",
        "ResultParam=Result",
        LAST
    );
```

Structuring an XML Script

Initially, you create a new script in your preferred protocol. You can record a session in that protocol, or you may program the entire script without recording. Structure the Actions section of the script as follows:

- XML input declaration
- The Actions section

The XML input section contains the XML tree that you want to use as an input variable. You define the XML tree as a char type variable. For example:

```
char *xml_input=
"<acme_org>"
    "<employee>"
        " <name>John Smith</name>"
        "<cubicle>227</cubicle>"
        "<extension>2145</extension>"
    "</employee>"
    "<employee>"
        "<name>Sue Jones</name>"
        "<cubicle>227</cubicle>"
        "<extension>2375</extension>"
    "</employee>"
"</acme_org>";
```

The Action section contains the evaluation of the variables and queries for the element values. In the following example, the XML input string is evaluated using **lr_save_string**. The input variable is queried for employee names and extensions.

```
Action() {

    /* Save the input as a parameter.*/
    lr_save_string(xml_input, "XML_Input_Param");

    /* Query 1 - Retrieve an employee name from the specified element.*/
    lr_xml_get_values("XML={XML_Input_Param}",
        "ValueParam=OutputParam",
        "Query=/acme_org/employee/name", LAST);

    /* Query 2 - Retrieve an extension under any path below the root.*/
    lr_xml_get_values("XML={XML_Input_Param}",
        "ValueParam=OutputParam",
        "Query=//extension", LAST);

    return 0;
}
```

Enhancing a Recorded Session

You can prepare an XML script by recording a session and then manually adding the relevant XML and Vuser API functions.

The following example illustrates how a recorded session was enhanced with Vuser API functions. Note that the only function that was recorded was **web_submit_data**, which appears in bold.

The first section contains the XML input declaration of the variable SOAPTemplate, for a SOAP message:

```
#include "as_web.h"

// SOAP message
const char*pSoapTemplate=
          "<soap:Envelope xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">"
               "<soap:Body>"
                    "<SendMail xmlns=\"urn:EmailIPortTypeInft-IEmailService\"/>"
               "</soap:Body>"
          "</soap:Envelope>";
```

The following section represents the actions of the user:

```
Action1()
{
    // get response body
    web_reg_save_param("ParamXml", "LB=", "RB=", "Search=body", LAST);

    // fetch weather by HTTP GET
    web_submit_data("GetWeather",
                    "Action=http://glkev.net.innerhost.com/glkev_ws/
                          WeatherFetcher.asmx/GetWeather",
                        "Method=GET",
                        "EncType=",
                        "RecContentType=text/xml",
                        "Referer=http://glkev.net.innerhost.com
                           /glkev_ws/WeatherFetcher.asmx?op=GetWeather",
        "Snapshot=t2.inf",
        "Mode=HTTP",
        ITEMDATA,
        "Name=zipCode", "Value=10010", ENDITEM,
        LAST);

    // Get City value
    lr_xml_get_values("Xml={ParamXml}",
                    "Query=City",
                    "ValueParam=ParamCity",
                    LAST
    );

    lr_output_message(lr_eval_string("***** City = {ParamCity} *****"));

    // Get State value
    lr_xml_get_values("Xml={ParamXml}",
                    "Query=State",
                    "ValueParam=ParamState",
                    LAST
    );

    lr_output_message(lr_eval_string("***** State = {ParamState} *****"));
```

```
// Get several values at once by using template
lr_xml_get_values_ex("Xml={ParamXml}",
                        "Template="
                              "<Weather>"
                                  "<Time>{ParamTime}</Time>"
                                  "<Temperature>{ParamTemp}</Temperature>"
                                  "<Humidity>{ParamHumid}</Humidity>"
                                  "<Conditions>{ParamCond}</Conditions>"
                              "</Weather>",
                       LAST
    );

    lr_output_message(lr_eval_string("***** Time = {ParamTime}, Temperature =
                                                {ParamTemp}, "
                                     "Humidity = {ParamHumid}, Conditions =
                                          {ParamCond} *****"));

    // Generate readable forecast
    lr_save_string(lr_eval_string("\r\n\r\n*** Weather Forecast for {ParamCity}, {ParamState} ***\r\n"
                               "\tTime: {ParamTime}\r\n"
                              "\tTemperature: {ParamTemp} deg. Fahrenheit\r\n"
                              "\tHumidity: {ParamHumid}\r\n"
                              "\t{ParamCond} conditions expected\r\n"
                             "\r\n"),
                              "ParamForecast"
    );

    // Save soap template into parameter
    lr_save_string(pSoapTemplate, "ParamSoap");
```

```
// Insert request body into SOAP template
    lr_xml_insert("Xml={ParamSoap}",
                  "ResultParam=ParamRequest",
                  "Query=Body/SendMail",
                  "position=child",
                  "XmlFragment="
                            "<FromAddress>taurus@merc-int.com</FromAddress>"
                            "<ToAddress>support@merc-int.com</ToAddress>"
                            "<ASubject>Weather Forecast</ASubject>"
                            "<MsgBody/>",
                  LAST
    );

//
//      "<soap:Envelope xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">"
//          "<soap:Body>"
//              "<SendMail xmlns=\"urn:EmailIPortTypeInft-IEmailService\"/>"
//                          "<FromAddress>taurus@merc-int.com</FromAddress>"
//                          "<ToAddress>support@merc-int.com</ToAddress>"
//                          "<ASubject>Weather Forecast</ASubject>"
//                          "<MsgBody/>"
//              "</SendMail>"
//          "</soap:Body>"
//      "</soap:Envelope>";
//

    // Insert actual forecast text
    lr_xml_set_values("Xml={ParamRequest}",
                      "ResultParam=ParamRequest",
                      "Query=Body/SendMail/MsgBody",
                      "ValueParam=ParamForecast",
                      LAST);
    // Add header for SOAP
    web_add_header("SOAPAction", "urn:EmailIPortTypeInft-IEmailService");

    // Get response body
    web_reg_save_param("ParamXml", "LB=", "RB=", "Search=body", LAST);

    // Send forecast to recipient, using SOAP request
    web_custom_request("web_custom_request",
        "URL=http://webservices.matlus.com/scripts/emailwebservice.dll/soap/IEmailservice",
        "Method=POST",
        "TargetFrame=",
        "Resource=0",
        "Referer=",
        "Body={ParamRequest}",
        LAST);

    // Verify that mail was sent
    lr_xml_find("Xml={ParamXml}",
                    "Query=Body/SendMailResponse/return",
                    "Value=0",
                    LAST
    );

    return 0;
}
```

Using Result Parameters

Some of the **lr_xml** functions return a result parameter, such as **ResultParam**. This parameter contains the resulting XML data after the function is executed. The result parameters will be available from the parameter list in the Select or Create Parameter dialog box.

For example, for **lr_xml_insert**, ResultParam contains the complete XML data resulting from the insertion of the new XML fragment

You can use the result parameters as input to other XML related functions such as Web Service calls. During replay, VuGen captures the value of the result parameter. In a later step, you can use that value as an input argument.

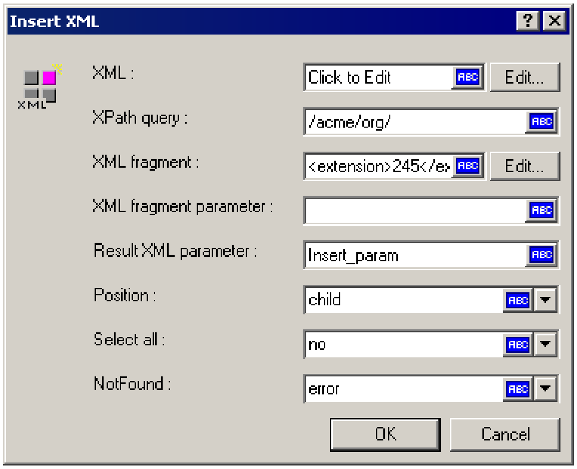
The functions that support result parameters are **lr_xml_insert**, **lr_xml_transform**, **lr_xml_replace**, **lr_xml_delete**, and **lr_xml_set_values**.

The following functions save values to a parameter other than the resultParam: **lr_xml_get_values** saves values to ValueParam and **lr_xml_extract** saves values to XMLFragmentParam. These values are also available for parameter substitution.

To use the result parameter as input:

1 In Tree view, double-click on an XML step to view its Properties.

2 In the Result XML Parameter box, specify a name for the **Result XML parameter** (or ValueParam and XMLFragmentParam).



Insert XML

XML : Click to Edit

XPath query : /acme/org/

XML fragment : <extension>245</e

XML fragment parameter :

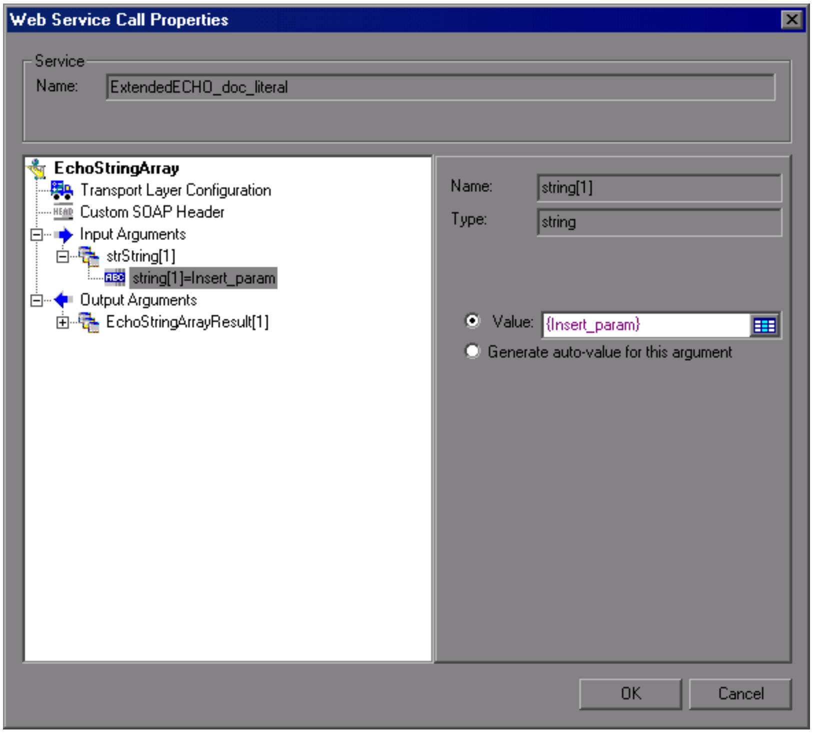
Result XML parameter : Insert_param

Position : child

Select all : no

NotFound : error

3 Reference the parameter name as in input argument.



Web Service Call Properties

Service

Name: ExtendedECHO_doc_literal

- EchoStringArray
  - Transport Layer Configuration
  - Custom SOAP Header
  - Input Arguments
    - strString[1]
      - string[1]=Insert_param
  - Output Arguments
    - EchoStringArrayResult[1]

Name: string[1]

Type: string

● Value: {Insert_param}

○ Generate auto-value for this argument

For more information, see "Input Argument Values" on page 97.

27

VuGen Debugging Tips

You can use the following integration and configuration tips to help you produce an error-free Vuser script.

This chapter includes:

- General Debugging Tip on page 530
- Using C Functions for Tracing on page 530
- Adding Additional C Language Keywords on page 530
- Examining Replay Output on page 531
- Debugging Database Applications on page 531
- Working with Oracle Applications on page 533
- Solving Common Problems with Oracle 2-Tier Vusers on page 534
- Two-tier Database Scripting Tips on page 539
- Running PeopleSoft-Tuxedo Scripts on page 548

General Debugging Tip

VuGen can be used as a regular text editor. You can open any text file in it and edit it. When an error message is displayed during replay in the output window below, you can double click on it and VuGen jumps the cursor to the line of the test that caused the problem. You can also place the cursor on the error code and press F1 to view the online help explanation for the error code.

Using C Functions for Tracing

You can use the C interpreter trace option (in version 230 or higher) to debug your Vuser scripts. The ci_set_debug statement allows trace and debug to be turned on and off at specific points in the script.

```
ci_set_debug(ci_this_context, int debug, int trace);
```

For example, you could add the following statements to your script:

```
ci_set_debug(ci_this_context, 1, 1) /* turn ON trace & debug */
ci_set_debug(ci_this_context, 0, 0) /* turn OFF trace & debug */
```

Adding Additional C Language Keywords

When you run a C script in VuGen, its parser uses the built-in C interpreter to parse the functions in the script. You can add keywords that are not part of the standard parser's library. By default, several common C++ keywords are added during installation, such as *size_t* and *DWORD*. You can edit the list and add additional keywords for your environment.

To add additional keywords:

1 Open the vugen_extra_keywords.ini file, located in your machine's <Windows> or <Windows>/System directory.

2 In the EXTRA_KEYWORDS_C section, add the desired keywords for the C interpreter.

The file has the following format:

```
[EXTRA_KEYWORDS_C]
FILE=
size_t=
WORD=
DWORD=
LPCSTR=
```

Examining Replay Output

Look at the replay output (either from within VuGen, or the file **output.txt** representing the output of the VuGen driver). You may also change the run-time settings options in VuGen to select more extensive logging in order to obtain a more detailed log output of the replayed test.

Debugging Database Applications

The following tips apply to database applications only (Oracle, ODBC, Ctlib):

- Generating Debugging Information
- Examining Compiler Information
- Code Generation Information
- Preprocessing and Compilation Information

Generating Debugging Information

Note: You can now set options to view most of the information described in this section using VuGen's user interface.

VuGen contains an inspector "engine." You can force VuGen recorder to create "inspector" output by editing *WINDOWS_DIR**vugen.ini* as follows:

```
[LogMode]
EnableAscii=ASCII_LOG_ON
```

When this option is enabled, VuGen creates a file, **vuser.asc** in the Data directory at the end of the recording. Note that this option should be used for debugging purposes only. This output file can become very large (several MB) and have serious effects on machine performance and disk space.

For cases like ODBC-based applications, it is possible to configure the ODBC Administrator (located in the Windows Control Panel) to provide a similar trace output. Open the ODBC options, and select 'Trace ODBC calls' to ON. Similarly the ODBC Developer Kit provides a Spy utility for call tracing.

To enable further debug information, add the following section to the \WINDOWS_DIR\vugen.ini file:

```
[INSPECTOR]
TRACE_LEVEL=3
TRACE_FILENAME=c:\tmp\sqltrace.txt
```

The file (sqltrace.txt) will include useful internal information regarding the hooking calls made during recording. The trace_level is between 1 and 3, with 3 representing the most detailed debug level. Note that in VuGen versions 5.02 and higher, you can set the trace level from the user interface.

Examining Compiler Information

You can view information about each stage of code generation, preprocessing and compilation to determine the source of any errors.

Code Generation Information

Look at the **vuser.log** file under the Data directory. This file, which contains a log of the code generation phase, is automatically created at the end of every lrd recording (i.e. all database protocols).

The following is an example of a log file:

```
lrd_init: OK
lrd_option: OK
lrd_option: OK
lrd_option: OK
```

```
Code generation successful
lrd_option: OK
lrd_end: OK
```

If any of the messages are not OK or successful, then a problem occurred during the code generation.

Preprocessing and Compilation Information

During runtime, VuGen displays information about both the preprocessing and compilation processes.

Working with Oracle Applications

Oracle Applications is a two-tier ("fat" client) packaged application, made up of 35 different modules (Oracle Human Resources, Oracle Financials, and so forth).

There are a number of issues that you should be aware of while recording and replaying Vusers for Oracle Applications:

- A typical script contains thousands of events, binds and assigns.
- A typical script has many db connections per user session.
- scripts almost always require correlated queries.
- Oracle Applications' clients are 16-bit only (developed with Oracle Developer 2000). This means that for debugging, if you don't have the Oracle 32bit client, you need to use VuGen's Force 16-bit options.

When a new window is created, the application retrieves an .xpf file from the file system for display. Currently, VuGen does not take this into consideration since it records at the client/server level. Therefore, there is a fairly significant inaccuracy in performance measurements since in most cases performance problems are related to the network bottleneck between clients and file server. We are currently thinking about this problem and how, if at all, to solve it.

Solving Common Problems with Oracle 2-Tier Vusers

This section contains a list of common problems that you may encounter while working with Oracle Vusers, and suggested solutions.

ORA-20001 and ORA-06512

Errors ORA-20001 and ORA-06512 appear during replay when the lrd_stmt contains the pl/sql block: fnd_signon.audit_responsibility(...)

This statement fails during replay because the sign-on number is unique for each new connection.

Solution

In order to solve this problem you need to use the new correlation tool for the sign-on number. This is second assigned value in the statement.

After you scan for possible values to correlate, highlight the value of the second lrd_assign_bind() for the failed statement. Note that the values in the "correlated query" window may not appear in the same order as the actual recorded statements.

The grid containing the substitution value should appear after the lrd_stmt which contains the pl/sql block: fnd_signon.audit_user(...).

Note: Since the sign-on number is unique for every connection, you need to use correlation for each new connection that you record.

Example of Solution

The following statement failed in replay because the second value, "1498224" is the unique sign-on number for every new connection.

```
lrd_stmt(Csr6, "begin fnd_signon.audit_responsibility(:s,:l,:f,:a,:r,:t,:p)"
       "; end;", -1, 1, 1, 0);
   lrd_assign_bind(Csr6, "s", "D", &s_D216, 0, 0, 0);
   lrd_assign_bind(Csr6, "l", "1498224", &l_D217, 0, 0, 0);
   lrd_assign_bind(Csr6, "f", "1", &f_D218, 0, 0, 0);
   lrd_assign_bind(Csr6, "a", "810", &a_D219, 0, 0, 0);
   lrd_assign_bind(Csr6, "r", "20675", &r_D220, 0, 0, 0);
   lrd_assign_bind(Csr6, "t", "Windows PC", &t_D221, 0, 0, 0);
   lrd_assign_bind(Csr6, "p", "", &p_D222, 0, 0, 0);
   lrd_exec(Csr6, 1, 0, 0, 0, 0);
```

The sign-on number can be found in the lrd_stmt with "fnd_signon.audit_user". The value of the first placeholder "a" should be saved. The input of "a" is always "0" but the output is the requested value.

Modified code:

```
 lrd_stmt(Csr4, "begin fnd_signon.audit_user(:a,:l,:u,:t,:n,:p,:s); end;", -1, 1, 1, 0);
   lrd_assign_bind(Csr4, "a", "0", &a_D46, 0, 0, 0);
   lrd_assign_bind(Csr4, "l", "D", &l_D47, 0, 0, 0);
   lrd_assign_bind(Csr4, "u", "1001", &u_D48, 0, 0, 0);
   lrd_assign_bind(Csr4, "t", "Windows PC", &t_D49, 0, 0, 0);
   lrd_assign_bind(Csr4, "n", "OraUser", &n_D50, 0, 0, 0);
   lrd_assign_bind(Csr4, "p", "", &p_D51, 0, 0, 0);
   lrd_assign_bind(Csr4, "s", "14157", &s_D52, 0, 0, 0);
   lrd_exec(Csr4, 1, 0, 0, 0, 0);
lrd_save_value(&a_D46, 0, 0, "saved_a_D46");
   Grid0(17);
lrd_stmt(Csr6, "begin fnd_signon.audit_responsibility(:s,:l,:f,:a,:r,:t,:p)"
      "; end;", -1, 1, 1, 0);
   lrd_assign_bind(Csr6, "s", "D", &s_D216, 0, 0, 0);
   lrd_assign_bind(Csr6, "l", "<saved_a_D46>", &l_D217, 0, 0, 0);
   lrd_assign_bind(Csr6, "f", "1", &f_D218, 0, 0, 0);
   lrd_assign_bind(Csr6, "a", "810", &a_D219, 0, 0, 0);
   lrd_assign_bind(Csr6, "r", "20675", &r_D220, 0, 0, 0);
   lrd_assign_bind(Csr6, "t", "Windows PC", &t_D221, 0, 0, 0);
   lrd_assign_bind(Csr6, "p", "", &p_D222, 0, 0, 0);
   lrd_exec(Csr6, 1, 0, 0, 0, 0);
```

Working with large numbers

Large numbers (NUMBER data type) sometimes appear in different format in the GRID and in the ASCII file. This difference makes it more difficult to identify numbers while searching for values to save for correlation.

For example, you could have a value appear as 1000003 in the grid, but as 1e+0006 in the Recording Log (ASCII file).

Workaround

If you have an error during replay and the correlation tool cannot locate the value in previous results, look for this value in the other format in grid.

ORA-00960

This error may occur with non-unique column names. For example:

```
lrd_stmt(Csr9, "SELECT UOM_CODE, UOM_CODE, DESCRIPTION FROM "
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

In this case you receive the following error:

```
"lrdo.c/fjParse: "oparse" ERROR return-code=960, oerhms=ORA-00960:
ambiguous column naming in select list".
```

Workaround

Change the statement by adding an alias to at least one of the non-unique columns, thus mapping it to a new unique name. For example:

```
lrd_stmt(Csr9,"SELECT UOM_CODE,UOM_CODE second, DESCRIPTION
FROM"
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

Alternate Workaround: remove ORDER BY from the lrd statement.

ORA-2002

Error 2002 appears when you try to use an unopened cursor. It occurs when you replay a user more than one iteration and you recorded into more than one section of the script.

Specifically, if a cursor is opened in the vuser_init section and closed in the Actions section, then you will encounter this error on the second iteration if you try to use the cursor. This is because it was closed but not re-opened.

For example: You have *lrd_open_cursor* in the vuser_init section and *lrd_close_cursor* in the Actions section. If you replay this user more than one iteration, you are going to get an error in the second iteration because you try using an unopened cursor (it was closed in first iteration, but not re-opened in the second).

Workaround

The easiest way to solve this is to move the **lrd_close_cursor** or/and **lrd_close_connection** of the problem cursor to the *vuser_end* section.

Database Protocols (lrd)

Replay of recorded asynchronous operations is not supported.

Wrong Client Version

You may receive an error message when running the wrong Oracle client version:

```
"Error: lrdo_open_connection: "olog" LDA/CDA return-code_019: unable to
allocate memory in the user side"
```

Workaround

You need to modify the library information in the *lrd.ini* file, located in the your product's bin directory. This file contains the settings that indicate which version of database support is loaded during recording or replay. The file contains a section for each type of host. For example, the following section of the *lrd.ini* file is for Oracle on HP/UX:

```
[ORACLE_HPUX]
;816=liblrdhpo816.sl
;81=liblrdhpo81.sl
;80=liblrdhpo80.sl
73=liblrdhpo73.sl
72=liblrdhpo72.sl
```

These settings indicate that Vusers should use the LoadRunner library liblrdhpo816.sl if the client uses Oracle 8.1.6, liblrdhpo81.sl for Oracle 8.1.5, and so on.

During replay on UNIX, the settings in the lrd *ini* file must indicate the correct version of the database to use. Suppose it is necessary to replay a Vuser for HP/UX using Oracle 8.1.5. In that case the previous lines for other versions of Oracle should be commented out with a ";" at the beginning of the line.

This section of the lrd.ini file will now look like:

```
[ORACLE_HPUX]
;816=liblrdhpo816.sl
81=liblrdhpo81.sl
;80=liblrdhpo80.sl
73=liblrdhpo73.sl
72=liblrdhpo72.sl
```

You also may need to make a change for Win32 if the application does not use the DLL mentioned in the lrd.ini file. For example, PowerBuilder 6.5 uses Oracle 8.0.5, but it uses the ora803.dll, not the ora805.dll. In that case, either comment out the 805 and 804 sections of the ORACLE_WINNT section, or change the 805 section from:

```
805=lrdo32.dll+ora805.dll
```

to

```
805=lrdo32.dll+ora803.dll
```

Two-tier Database Scripting Tips

The following section offers solutions for two-tier database scripts. For Siebel specific solutions, see "Siebel-specific Scripting Tips" on page 544.

Question 1: Why does the script fail when it is data driven, while the same values work with the application itself?

Answer: The failure may be a result of trailing spaces in your data values. Even though the data values that you type directly into the GUI are probably truncated, you should manually eliminate them from your data file. Tab-delimited files can hide trailing spaces and therefore obscure problems. In general, comma-delimited files are recommended. You can view the files in Excel to see if things are correct.

Question 2: Why does an SQL error of an invalid cursor state occur on the second iteration?

Answer: The **lrd_close_cursor** function may not have been generated or it may be in the *end* section instead of the *action* section. You will need to add a cursor close function or move it from the *end* section to make the script iterate successfully.

Opening a new cursor may be costly in terms of resources. Therefore, we recommend that you only open a cursor once in the *actions* section during the first iteration. You can then add a new parameter that contains the iteration number as a string by using the Iteration Number type. Call this parameter *IterationNum*. Then, inside the *actions* section replace a call to open a new cursor like

```
lrd_open_cursor(&Csr1, Con1, 0);
```

with

```
if (!strcmp(lr_eval_string("<IterationNum>"), "1"))
   lrd_open_cursor(&Csr1, Con1, 0);
```

Question 3: How can I fix code produced by VuGen that will not compile because of data declarations in the *vdf.h* file?

Answer: The problem, most likely, is an SQL data type that is not supported by VuGen. For Microsoft SQL, you can often work around this issue by replacing the undefined error message in *vdf.h* with "DT_SZ" (null terminated string). Although this is not the actual datatype, VuGen can compile the script correctly. Please report the problem and send the original script to customer support.

Question 4: What is the meaning of LRD Error 2048?

Answer: VuGen is failing because it is trying to bind a variable with a longer length than what was allocated during recording. You can correct this by enlarging the variable definition in *vdf.h* to receive a longer string back from the database. Search this file for the unique numeric identifier. You will see its definition and length. The length is the third element in the structure. Increase this length as required and the script will replay successfully.

For example, in the following script, we have:

```
lrd_assign(&_2_D354, "<ROW_ID>", 0, 0, 0);
```

In *vdf.h*, we search for *_2_D354* and find

```
static LRD_VAR_DESC _2_D354 = {
            LRD_VAR_DESC_EYECAT, 1, 10, LRD_BYTYPE_ODBC,
            {0 ,0, 0}, DT_SZ, 0, 0, 15, 12};
```

We change it to:

```
static LRD_VAR_DESC _2_D354 = {
            LRD_VAR_DESC_EYECAT, 1, 12, LRD_BYTYPE_ODBC,
            {0,0, 0}, DT_SZ, 0, 0, 15, 12};
```

The complete definition of LRD_VAR_DESC appears in *lrd.h*. You can find it by searching for typedef struct LRD_VAR_DESC.

Question 5: How can I obtain the number of rows affected by an UPDATE, INSERT or DELETE when using ODBC and Oracle?

Answer: You can use **lrd** functions to obtain this information. For ODBC, use **lrd_row_count**. The syntax is:

```
 int rowcount;
    .
    .
    .
lrd_row_count(Csr33, &rowcount, 0);
```

Note that **lrd_row_count** must immediately follow the pertinent statement execution.

For Oracle you can use the fourth argument of **lrd_exec**.

```
lrd_exec(Csr19, 1, 0, &rowcount, 0, 0);
```

If you are using Oracle's OCI 8, you can use the fifth argument of **lrd_ora8_exec**.

```
lrd_ora8_exec(OraSvc1, OraStm3, 1, 0, &uliRowsProcessed, 0, 0, 0, 0, 0);
```

Question 6: How can I avoid duplicate key violations?

Answer: Occasionally, you will see a duplicate key violation when performing an Insert. You should be able to find the primary key by comparing two recordings to determine the problem. Check whether this or earlier UPDATE or INSERT statement should use correlated queries. You can use the data dictionary in order to find the columns that are used in the violated unique constraint.

In Oracle you will see the following message when a unique constraint is violated:

```
ORA-00001: unique constraint (SCOTT.PK_EMP) violated
```

In this example SCOTT is the owner of the related unique index, and PK_EMP is the name of this index. Use SQL*Plus to query the data dictionary to find the columns. The pattern for this query is:

```
select column_name from all_ind_columns where index_name = '<IndexName>
and index_owner = '<IndexOwner>';
```

```
select column_name from all_ind_columns where index_name = 'PK_EMP' and
index_owner = 'SCOTT';
```

Since the values inserted into the database are new, they might not appear in earlier queries, but they could be related to the results of earlier queries, such as one more than the value returned in an earlier query.

For Microsoft SQL Server you will see one of these messages:

```
Cannot insert duplicate key row in object 'newtab' with unique index 'IX_newtab'.
Violation of UNIQUE KEY constraint 'IX_Mark_Table'. Cannot insert duplicate
key in object 'Mark_Table'.
Violation of PRIMARY KEY constraint 'PK_NewTab'. Cannot insert duplicate key
in object 'NewTab'.
```

You can use the Query Analyzer to find out which columns used by the key or index. The pattern for this query is:

```
select C.name
    from sysindexes A, sysindexkeys B, syscolumns C
    where C.colid = B.colid and C.id = B.id and
    A.id = B.id and A.indid = B.indid
    and A.name = '<IndexName>' and A.id = object_id('<TableName>')
select C.name
    from sysindexes A, sysindexkeys B, syscolumns C
    where C.colid = B.colid and C.id = B.id and
    A.id = B.id and A.indid = B.indid
    and A.name = 'IX_newtab' and A.id = object_id('newtab')
```

For DB2 you might see the following message:

```
SQL0803N One or more values in the INSERT statement, UPDATE statement,
or foreign key update caused by a DELETE statement are not valid because
they would produce duplicate rows for a table with a primary key, unique
constraint, or unique index. SQLSTATE=23505
```

If you still encounter problems, be sure to check the number of rows changed for Updates and Inserts for both recording and replay. Very often, an UPDATE fails to change any rows during replay, because the WHERE clause was not satisfied. This does not directly result in an error, but it causes a table not to be properly updated, and can cause a later SELECT to select the wrong value when correlating the query.

Also verify that there are no problems during multi-user replay. In certain instances, only one user will successfully perform an UPDATE. This occurs with Siebel, where it is necessary to manually write a loop to overcome the problem.

Question 7: The database does not appear to be modified after replaying a script which should have modified the database.

Answer: Through the user application's UI, check if the updated values appear when trying to see the current data accessible to the application. If the values have not been updated, you need to determine they were not changed. Possibly, an UPDATE statement changed one or more rows when the application was recorded, and did not change any during replay.

Check these items:

- **Verify statement.** If there is a WHERE clause in the UPDATE statement, verify that it is correct.
- **Check for correlations.** Record the application twice and compare the UPDATE statements from each of the recordings to make sure that the necessary correlations were performed.
- **Check the total number of rows.** Check the number of rows that were changed after the UPDATE. For Oracle, this information is stored in the fourth parameter of **lrd_exec**. For ODBC, use **lrd_row_count** to determine the number of rows updated. You can also add code to your script that prints the number of rows that were updated. If this value is 0, the UPDATE failed to modify the database.
- **Check the SET clause.** Check the SET clause of the UPDATE statement. Make sure that you correlated any necessary values here instead of hard-coding them. You can see this by comparing two recordings of the UPDATE.

It certain cases, the UPDATE works when replaying one Vuser, but not for multiple Vusers. The UPDATE of one Vuser might interfere with that of another. Parameterize each Vuser so that each one uses different values during the UPDATE, unless you want each vuser to update with the same values. In this case try adding retry logic to perform the UPDATE a second time.

Question 8: How do I avoid the unique column name error when replaying a statement recorded with an Oracle Application. For example:

```
lrd_stmt(Csr9, "SELECT UOM_CODE, UOM_CODE, DESCRIPTION FROM "
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

The following error message was issued:

```
"lrdo.c/fjParse:  "oparse" ERROR return-code=960, oerhms=ORA-00960:
ambiguous column naming in select list".
```

Answer: Change the statement by adding an alias to at least one of the non-unique columns, thereby mapping it to a new unique name. For example:

```
lrd_stmt(Csr9,"SELECT UOM_CODE,UOM_CODE second, DESCRIPTION
FROM"
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

Siebel-specific Scripting Tips

This section offers solutions for Siebel database users. You should also see the previous section which discusses some general database scripting tips.

Question 9: Virtual users run fine in VuGen but fail in the Controller with duplicate key violations.

Answer: The Siebel client stores a key in the NEXT_SUFFIX column of the S_SSA_ID table. This client has code that detects and recovers from situations in which it fails to successfully get a block of suffix values.

VuGen automatically correlates the NEXT_SUFFIX and MODIFICATION_NUM fields of the S_SSA_ID table. During an UPDATE the MODIFICATION_NUM field is incremented by 1 and the NEXT_SUFFIX field is increased by 100 in base 36. However, VuGen does not add code in instances where a client could not obtain a new block of suffix values. As a result, the replay fails with a unique constraint error, when you attempt to insert new values into the database.

You must manually add code to each location in the script where a block of suffixes is obtained, in order to perform a retry if the first attempt fails. You can locate these places by searching for SiebelPreSave in the script. You must also add a *while* loop with code similar to the example below. This example only works for Oracle. For ODBC use **lrd_row_count** instead of using the fourth argument of **lrd_exec**.

```
unsigned long lRowUpdated;
int nAttempt;

…

// This loops until we successfully obtain a "next_suffix"
lRowUpdated = 0;
nAttempt=0;

while (lRowUpdated != 1) {

   nAttempt++;
   if (nAttempt > 1)
      lr_output_message (".......Next suffix retry %d", nAttempt);
   else
   {
      lrd_open_cursor(&Csr13, Con1, 0);
      lrd_stmt(Csr13, "SELECT\n T1.LAST_UPD,\n T1.CREATED_BY,\n "
         "T1.CONFLICT_ID,\n T1.CREATED,\n T1.NEXT_SUFFIX,\n "
         "T1.ROW_ID,\n T1.NEXT_PREFIX,\n T1.CORPORATE_PREFIX,\n "
         "T1.MODIFICATION_NUM,\n T1.NEXT_FILE_SUFFIX,\n      "
         "T1.LAST_UPD_BY\n   FROM \n SIEBEL.S_SSA_ID T1", -1, 1, 1, 0);
   }
   lrd_bind_cols(Csr13, BCInfo_D375, 0);
   lrd_exec(Csr13, 0, 0, 0, 0, 0);

   SiebelPreSave_1();
   lrd_fetch(Csr13, -1, 4, 0, PrintRow26, 0);
   GRID(26);
   SiebelPostSave_1();

   if (nAttempt > 1)
   {
    lrd_open_cursor(&Csr14, Con1, 0);
    lrd_stmt(Csr14,"\nUPDATE SIEBEL.S_SSA_ID SET\n LAST_UPD_BY=:1,\n "
    "NEXT_SUFFIX = :2,\n      MODIFICATION_NUM = :3,\n      LAST_UPD = "
    ":4\n WHERE\n ROW_ID = :5 AND MODIFICATION_NUM = :6\n", -1, 1,
    1, 0);
   }
   lrd_assign_bind(Csr14, "6", "<modification_num>", &_6_D376, 0,
               LRD_BIND_BY_NUMBER, 0);
   lrd_assign_bind(Csr14,"5", "0-11",&_5_D377,0,LRD_BIND_BY_NUMBER, 0);
   strcpy (szTimeAtNewButton, lr_eval_string("<Now>"));
   sprintf (szTimeStamp, "%s %s", lr_eval_string("<Today>"),
           szTimeAtNewButton);
   lr_save_string (szTimeStamp, "DateTimeStamp");
lrd_assign_bind(Csr14, "4", "<DateTimeStamp>", &_4_D378, 0,
      LRD_BIND_BY_NUMBER, 0);
lrd_assign_bind(Csr14, "3", "<next_modnum>", &_3_D379, 0,
      LRD_BIND_BY_NUMBER, 0);
lrd_assign_bind(Csr14, "2", "<next_suffix_x100>", &_2_D380, 0,
      LRD_BIND_BY_NUMBER, 0);
lrd_assign_bind(Csr14, "1", "1-1E1",&_1_D381,0,LRD_BIND_BY_NUMBER, 0);

// this update won't update any rows unless we successfully got our suffix
lrd_exec(Csr14, 1, 0, &lRowUpdated, 0, 0);
lrd_commit(0, Con1, 0);

}//while
     lr_output_message ("…Rows updated %ld", lRowUpdated);
```

Question 10: How can I find the correct value to correlate for a primary key?

Answer: Siebel tends to generate key values based on base 36 mathematical manipulations of <*next_suffix*>. Try comparing several recordings and try to determine the relationships. You can ignore date fields when correlating Siebel, since they do not seem to effect script replay.

Question 11: How can I solve an INSERT into S_SRV_REQ failure with a duplicate key violation?

Answer: The primary key is SR_NUM. Newer versions of VuGen automatically correlate insertions into this table, by using the function lrd_siebel_str2num, which converts the NEXT_SUFFIX value of the S_SSA_ID table from base 36 to the base 10 equivalent. Older versions of VuGen might not handle this correlation correctly.

Question 12: VuGen does not automatically perform all the correlations I need in order to replay my script correctly. How can I add the missing correlations?

Answer: Currently VuGen only saves the values of the NEXT_SUFFIX and MODIFICATION_NUM columns from the S_SSA_ID table and replaces them with parameters when they are used later in the script. You may need to add some additional correlations manually. The correlation code in the **SiebelPreSave** and **SiebelPostSave** functions in the *print.inl* file can serve as an example of how to correlate specific values once you determine what needs to be correlated.

- Sometimes the NEXT_FILE_SUFFIX and MODIFICATION_NUM columns are chosen from the S_SSA_ID table. In this case, an UPDATE statement updates the NEXT_FILE_SUFFIX by adding one to this string in base 36, and one to the MODIFICATION_NUM. The value of the NEXT_FILE_SUFFIX will often be inserted in the FILE_REV_NUM field of a table. Often the name of this table ends with the *_ATT* suffix, to indicate that it is an attachment.

- Whenever Siebel performs an UPDATE statement, there is a MODIFICATION_NUM column that is incremented by one. VuGen only generates this correlation automatically for the S_SSA_ID table. You have to do it manually for other cases.

- Siebel refers to records according to their ID number. Siebel usually finds all records of a particular type (such as an agreement), and then later uses the ID number for a record when trying to update or delete an existing record of this type. You need to replace the ID number by a parameter during replay in order to generate a meaningful load test. The ID number has the form of one or more digits, a hyphen, followed by one or more alphanumeric characters, such as 1-QPF9. VuGen does not do this parameterization automatically, so you have to do it manually.

- If you find any other missing correlations or parameterizations, please notify customer support in order that HP can improve VuGen's support for Siebel.

Running PeopleSoft-Tuxedo Scripts

To run PeopleSoft-Tuxedo Vusers with Tuxedo 7.x, you must change the library extension in the *mdrv.dat* file:

```
[PeopleSoft-Tuxedo]
WINNT_EXT_LIBS=lrt7.dll
```

28

Advanced Topics

The following advanced information can assist you in determining the replay issues and debugging your Vuser script.

This chapter includes:

➤ Files Generated During Recording on page 549

➤ Files Generated During Replay on page 552

➤ Running a Vuser from the Unix Command Line on page 553

➤ Specifying the Vuser Behavior on page 555

➤ Command Line Parameters on page 556

➤ Recording OLE Servers on page 556

➤ Examining the .dat Files on page 559

➤ Adding a New Vuser Type on page 560

Files Generated During Recording

Assume that the recorded test has been given the name ‘vuser’ and is stored under c:\tmp. Following is a list of the more important files that are generated after recording:

vuser.usr Contains information about the virtual user: type, AUT, action files, and so forth.

vuser.bak A copy of Vuser.usr before the last save operation.

default.cfg Contains a listing of all run-time settings as defined in the VuGen application (think time, iterations, log, web).

vuser.asc The original recorded API calls.

vuser.grd Contains the column headers for grids in database scripts.

default.usp Contains the script's run logic, including how the actions sections run.

init.c Exact copy of the Vuser_init function as seen in the VuGen main window.

run.c Exact copy of the Action function as seen in the VuGen main window.

end.c Exact copy of the Vuser_end function as seen in the VuGen main window.

vdf.h A header file of C variable definitions used in the script.

\Data The Data directory stores all of the recorded data used primarily as a backup. Once the data is in this directory, it is not touched or used. For example, **Vuser.c** is a copy of **run.c**.

Example of Vuser.usr File

```
[General]
Type=Oracle_NCA
DefaultCfg=default.cfg
AppName=C:\PROGRA~1\Netscape\COMMUN~1\Program\netscape.exe
BuildTarget=
ParamRightBrace=>
ParamLeftBrace=<
NewFunctionHeader=0
MajorVersion=5
MinorVersion=0
ParameterFile=nca_test3.prm
GlobalParameterFile=
[Transactions]
Connect=
[Actions]
vuser_init=init.c
Actions=run.c
vuser_end=end.c
```

Example of default.cfg File

```
[General]
XlBridgeTimeout=120

[ThinkTime]
Options=NOTHINK
Factor=1
LimitFlag=0
Limit=1

[Iterations]
NumOfIterations=1
IterationPace=IterationASAP
StartEvery=60
RandomMin=60
RandomMax=90

[Log]
LogOptions=LogBrief
MsgClassData=0
MsgClassParameters=0
MsgClassFull=0
```

Files Generated During Replay

This section describes what occurs when the Vuser is replayed.

1 The **options.txt** file is created which includes command line parameters to the preprocessor.

2 The file **Vuser.c** is created which contains 'includes' to all the relevant .c and .h files.

3 The c preprocessor **cpp.exe** is invoked in order to 'fill in' any macro definitions, precompiler directives, and so on, from the development files.

The following command line is used:

```
cpp  -foptions.txt
```

4 The file **pre_cci.c** is created which is also a C file (**pre_cci.c** is defined in the **options.txt** file). The file **logfile.log** (also defined in **options.txt**) is created containing any output of this process. This file should be empty if there are no problems with the preprocessing stage. If the file is not empty then its almost certain that the next stage of compilation will fail due to a fatal error.

5 The **cci.exe** C compiler is now invoked to create a platform-dependent pseudo-binary file (.ci) to be used by the virtual user driver program that will interpret it at run-time. The cci takes the **pre_cci.c** file as input.

6 The file **pre_cci.ci** is created as follows:

```
cci    -errout  c:\tmp\Vuser\logfile.log    -c  pre_cci.c
```

7 The file **logfile.log** is the log file containing output of the compilation.

8 The file **pre_cci.ci** is now renamed to **Vuser.ci**.

Since the compilation can contain both warnings and errors, and since the driver does not know the results of this process, the driver first checks if there are entries in the **logfile.log** file. If there are, it then checks if the file **Vuser.ci** has been built. If the file size is not zero, it means that the cci has succeeded to compile - if not then compilation has failed and an error message will be given.

9 The relevant driver is now run taking both the **.usr** file and the **Vuser.ci** file as input. For example:

```
mdrv.exe  -usr c:\tmp\Vuser\Vuser.usr    -out c:\tmp\Vuser    -file
c:\tmp\Vuser\Vuser.ci
```

The **.usr** file is needed since it tells the driver program which database is being used. From here it can then know which libraries need to be loaded for the run.

10. The **output.txt** file is created (in the path defined by the 'out' variable) containing all the output messages of the run. This is the same output as seen in both the VuGen runtime output window and the VuGen main lower window.

Example of options.txt file

```
-DCCI
-D_IDA_XL
-DWINNT
-Ic:\tmp\Vuser(name and location of Vuser include files)
-IE:\LRUN45B2\include(name and location of include files)
-ec:\tmp\Vuser\logfile.log (name and location of output logfile)
 c:\tmp\Vuser\VUSER.c(name and location of file to be processed)
```

Example of Vuser.c file

```
#include "E:\LRUN45B2\include\lrun.h"
#include "c:\tmp\web\init.c"
#include "c:\tmp\web\run.c"
#include "c:\tmp\web\end.c"
```

Running a Vuser from the Unix Command Line

VuGen includes a Unix shell script utility, *run_db_Vuser.sh*, that automatically performs the same operations as the virtual user but from the command line. It can perform each of the replay steps optionally and independently. This is a useful tool for debugging tests to be replayed on Unix.

Place the file *run_db_Vuser.sh* in the $M_LROOT/bin directory. To replay a Vuser type:

```
run_db_Vuser.sh   Vuser.usr
```

You can also use the following command line options:

-cpp_only This option will start the prepocessing phase. The output of this process is the file *'Vuser.c'*.

-cci_only This option runs the compilation phase. The *'Vuser.c'* file is used as input, and the output produced is the *'Vuser.ci'* file.

-exec_only This option runs the Vuser, by taking as input the *'Vuser.ci'* file and running it via the replay driver.

-ci ci_file This option allows you to specify the name and location of a .ci file to be run. The second parameter contains the location of the .ci file.

-out output_directory This option allows you to determine the location of any output files created throughout the various processes. The second parameter is the directory name and location.

-driver driver_path This option allows you to specify the actual driver executable to be used for running the Vuser. By default the driver executable is taken from the settings in the VuGen.dat file.

Note that only one of the first three options can be used at a time for running the run_db_vuser.

Specifying the Vuser Behavior

Since VuGen creates the Vuser script and the Vuser behavior as two independent sources, you can configure user behavior without directly referencing the Vuser script, for example, wait times, pacing times, looping iterations, logging, and so forth. This feature lets you make configuration changes to a Vuser, as well as store several 'profiles' for the same Vuser script.

The *'Vuser.cfg'* file, by default, is responsible for defining this behavior - as specified in VuGen's Runtime settings dialog box. You can save several versions of this file for different user behavior and then run the Vuser script referencing the relevant *.cfg* file.

You can run the Vuser script with the relevant configuration file from a server machine. To do this, add the following to the Vuser command line:

```
-cfg c:\tmp\profile2.cfg
```

For information on command line parameters, see "Command Line Parameters" on page 556.

Note that you cannot control the behavior file from VuGen. VuGen automatically uses the *.cfg* file with the same name as the Vuser. (You can, of course, rename the file to be *'Vuser.cfg'*). However, you can do this manually from the command line by adding the -cfg parameter mentioned above to the end of the driver command line.

Note: The Unix utility, *run_db_vuser*, does not yet support this option.

Command Line Parameters

The Vusers can accept command line parameters when invoked. There are several Vuser API functions available to reference them (**lr_get_attrib_double**, and so on). In your environment, you can send command line parameters to the Vuser by adding them to the command line entry of the script window.

When running the Vuser from VuGen, you cannot control the command line parameters. You can do this manually, however, from the Windows command line by adding the parameters at the end of the line, after all the other driver parameters, for example:

```
mdrv.exe -usr c:\tmp\Vuser\Vuser.usr    -out c:\tmp\vuser
vuser_command_line_params
```

Note: The Unix utility, *run_db_vuser*, does not yet support this option.

Recording OLE Servers

VuGen currently does not support recording for OLE applications. These are applications where the actual process is not launched by the standard process creation routines, but by the OLE Automation system. However, you can create a Vuser script for OLE applications based on the following guidelines.

There are two types of OLE servers: executables, and DLLs.

DLL Servers

If the server is the DLL, it will eventually be loaded into the application process space, and VuGen will record the call to LoadLibrary. In this case, you may not even realize that it was an OLE application.

Executable Servers

If the server is the executable, you must invoke the executable in the VuGen in a special way:

- First, determine which process actually needs to be recorded. In most cases, the customer knows the name of the application's executable. If the customer doesn't know the name of the application, invoke it and determine its name from the NT Task Manager.
- After you identify the required process, click **Start Recording** in VuGen. When prompted for the Application name, enter the OLE application followed by the flag "/Automation". Next, launch the user process in the usual way (not via VuGen). VuGen records the running OLE server and does not invoke another copy of it. In most cases, these steps are sufficient to enable VuGen to record the actions of an OLE server.
- If you still are experiencing difficulties with recording, you can use the *CmdLine* program to determine the full command line of a process which is not directly launched. (The program is available in a knowledgebase article on the Customer Support Web site, http://support.hp.com)

Using CmdLine

In the following example, *CmdLine.exe* is used to determine the full command line for the process MyOleSrv.exe, which is launched by some other process.

To determine its full command line:

1. Rename *MyOleSrv.exe* to *MyOleSrv.orig.exe*.
2. Place *CmdLine.exe* in the same directory as the application, and rename it to *MyOleSrv.exe*.
3. Launch *MyOleSrv.exe*. It issues a popup with a message containing the complete command line of the original application, (including additional information), and writes the information into *c:\temp\CmdLine.txt*.
4. Restore the old names, and launch the OLE server, *MyOleSrv.exe*, from VuGen with the correct command line parameters. Launch the user application in a regular way - not through VuGen. In most cases, VuGen will record properly.

If you still are experiencing difficulties with recording, proceed with the following steps:

1. Rename the OLE server to MyOleSrv.1.exe, and CmdLine to MyOleSrv.exe.
2. Set the environment variables "CmdStartNotepad" and "CmdNoPopup" to 1. See "CmdLine Environment Variables" on page 558 for a list of the CmdLine environment variables.
3. Start the application (not from VuGen). Notepad opens with the full command line. Check the command line arguments. Start the application several times and compare the command line arguments. If the arguments are the same each time you invoke the application, then you can reset the CmdStartNotepad environment variable. Otherwise, leave it set to "1".
4. In VuGen, invoke the program, MyOleSrv.1.exe with the command line parameters (use Copy/Paste from the Notepad window).
5. Start the application (not from within VuGen).

CmdLine Environment Variables

You can control the execution of CmdLine through the following environment variables:

CmdNoPopup If set, the popup window will not appear.

CmdOutFileName If set, and non-empty, CmdLine will attempt to create this file instead of c:\temp\CmdLine.txt.

CmdStartNotepad If set, the output file will be displayed in the notepad (Best used with CmdNoPopup).

Examining the .dat Files

There are two .dat files used by VuGen: vugen.dat and mdrv.dat.

vugen.dat

This vugen.dat file resides in the M_LROOT\dat directory and contains general information about VuGen, to be used by both the VuGen and the Controller.

```
[Templates]
RelativeDirectory=template
```

The **Templates** section indicates where the templates are for the VuGen protocols. The default entry indicates that they are in the relative *template* directory. Each protocol has a subdirectory under *template*, which contains the template files for that protocol.

The next section is the GlobalFiles section.

```
[GlobalFiles]
main.c=main.c
@@TestName@@.usr=test.usr
default.cfg=test.cfg
default.usp=test.usp
```

The **GlobalFiles** section contains a list of files that VuGen copies to the test directory whenever you create a new test. For example, if you have a test called "user1", then VuGen will copy *main.c*, *user1.usr* and *user1.cfg* to the test directory.

The **ActionFiles** section contains the name of the file containing the Actions to be performed by the Vuser and upon which to perform iterations.

```
[ActionFiles]
@@actionFile@@=action.c
```

In addition to the settings shown above, *vugen.dat* contains settings that indicate the operating system and other compilation related settings.

mdrv.dat

The mdrv.dat file contains a separate section for each protocol defining the location of the library files and driver executables. The next section describes what you need to add to this file in order to define a new protocol.

Adding a New Vuser Type

To add a new Vuser type/protocol to VuGen, you need to:

- Edit the *mdrv.dat file* with the new protocol's settings.
- Add a *.cfg* file.
- Insert an *.lrp* file.
- Create a template directory.

Editing the mdrv.dat File

First, you edit the mdrv.dat file which resides in the M_LROOT\dat directory. You add a section for the new Vuser type with all of the applicable parameters from the following list.

```
[<extension_name>]
ExtPriorityType=< {internal, protocol}>
WINNT_EXT_LIBS=<dll name for NT>
WIN95_EXT_LIBS=<dll name for 95>
SOLARIS_EXT_LIBS=<dll name for Solaris>
LINUX_EXT_LIBS=<dll name for Linux>
HPUX_EXT_LIBS=<dll name for HP>
AIX_EXT_LIBS=<dll name for IBM>
LibCfgFunc=<configuration function name>
UtilityExt=<other extensions list>
WINNT_DLLS=<dlls to load to the interpreter context, for NT>
WIN95_DLLS=<dlls to load to the interpreter context, for 95>
SOLARIS_DLLS=<dlls to load to the interpreter context, for Solaris>
LINUX_DLLS=<dlls to load to the interpreter context, for Linux>
HPUX_DLLS=<dlls to load to the interpreter context, for HP>
AIX_DLLS=<dlls to load to the interpreter context, for IBM>
ExtIncludeFiles=<extra include files. several files can be seperated by a comma>
ExtCmdLineConc=<extra command line (if the attr exists concatenate value)>
ExtCmdLineOverwrite=<extra command line (if the attr exists overwrite value)>
CallActionByNameFunc=<interpreter exec_action function>
GetFuncAddress=<interpreter get_location function>
RunLogicInitFunc=<action_logic init function>
RunLogicRunFunc=<action_logic run function>
RunLogicEndFunc=<action_logic end function>
```

For example, an Oracle NCA Vuser type is represented by:

```
[Oracle_NCA]
ExtPriorityType=protocol
WINNT_EXT_LIBS=ncarp11i.dll
WIN95_EXT_LIBS=ncarp11i.dll
LINUX_EXT_LIBS=liboranca11i.so
SOLARIS_EXT_LIBS=liboranca11i.so
HPUX_EXT_LIBS=liboranca11i.sl
AIX_EXT_LIBS=liboranca11i.so
LibCfgFunc=oracle_gui_configure
UtilityExt=lrun_api,HttpEngine
ExtCmdLineOverwrite=-WinInet No
ExtCmdLineConc=-UsingWinInet No
SecurityRequirementsFiles=oracle_nca.asl
SecurityMode=On
```

VuGen was designed to be able to handle a new Vuser type with no code modifications. You may, however, need to add a special View.

There is no generic driver supplied with VuGen, but you can customize one of the existing drivers. To use a customized driver, modify *mdrv.dat*. Add a line with the platform and existing driver, then add a new line with your customized driver name, in the format *<platform>_DLLS=<my_replay.dll name>*. For example, if your SAP replay dll is called SAPPLAY32.DLL, add the following two lines to the [sap] section of *mdrv.dat*:

```
WINNT=sapdrv32.exe
WINNT_DLLS=sapplay32.dll
```

Adding a CFG file

You can optionally specify a configuration file to set the default Run-Time Settings for your protocol. You define it in the LibCfgFunc variable in the mdrv.dat file, or place one called default.cfg in the new protocols subdirectory under templates. A sample default.cfg follows.

```
[ThinkTime]
Options=NOTHINK
Factor=1
LimitFlag=0
Limit=1

[Iterations]
NumOfIterations=1
IterationPace=IterationASAP
StartEvery=60
RandomMin=60
RandomMax=90

[Log]
LogOptions=LogExtended
MsgClassData=0
MsgClassParameters=0
MsgClassFull=1
```

Inserting an LRP file

In the dat/protocols directory, insert an *lrp* file which defines the protocol. This file contains the configuration information for the protocol in the Protocol, Template, VuGen, and API sections. Certain protocols may have additional sections, corresponding to the additional run-time setting options.

The Protocol section contains the name, category, description, and bitmap location for the protocol.

```
 [Protocol]
Name=WAP
CommonName=WAP
Category=Wireless
Description=Wireless Application Protocol - used for Web-based, wireless
communication between mobile devices and content providers.
Icon=bitmaps\wap.bmp
Hidden=0
Single=1
Multi=0
```

The Template section indicates the name of the various sections of the script and the default test name.

```
[Template]
vuser_init.c=init.c
vuser_end.c=end.c
Action1.c=action.c
Default.usp=test.usp
@@TestName@@.usr=wap.usr
default.cfg=default.cfg
```

The **VuGen** section has information about the record and replay engines, along with the necessary DLLs and run-time files.

The **API** section contains information about the protocol’s script API functions.

You can use one of the existing lrp files in the protocols directory as a base for your new protocol.

Specifying a Template

After adding an *lrp* file, insert a subdirectory to *M_LROOT/*template with a name corresponding to the protocol name defined in the *lrp* file. In this subdirectory, insert a *default.cfg* file which defines the default settings for the general and run-time settings.

If you want to use a global header file for all of your protocol's scripts, add a file named *globals.h*. This file should contain an include statement which points to a header file for the new protocol. For example, the template/http subdirectory contains a file called *globals.h* which directs VuGen to the *as_web.h* file in the include directory:

```
#include #as_web.h"
```

Part 6

Appendixes

A

Calling External Functions

When working with VuGen, you can call functions that are defined in external DLLs. By calling external functions from your script, you can reduce the memory footprint of your script and the overall run-time.

To call the external function, you load the DLL in which the function is defined.

You can load a DLL:

- locally—for one script, using the **lr_load_dll** function
- globally—for all scripts, by adding statements to the vugen.dat file

This appendix includes:

- Loading a DLL Locally on page 569
- Loading a DLL Globally on page 571

Loading a DLL Locally

You use the **lr_load_dll** function to load the DLL in your Vuser script. Once the DLL is loaded, you can call any function defined within the DLL, without having to declare it in your script.

To call a function defined in a DLL:

1 Use the **lr_load_dll** function to load the DLL at the beginning of your script. Place the statement at the beginning of the *vuser_init* section. **lr_load_dll** replaces the **ci_load_dll** function.

Use the following syntax:

lr_load_dll(*library_name*);

Note that for UNIX platforms, DLLs are known as shared libraries. The extension of the libraries is platform dependent.

2 Call the function defined in the DLL in the appropriate place within your script.

In the following example, the insert_vals function, defined in orac1.dll, is called, after the creation of the Test_1 table.

```
int LR_FUNC Actions(LR_PARAM p)
{
lr_load_dll("orac1.dll");

lrd_stmt(Csr1, "create table Test_1 (name char(15), id integer)\n", -1,
        1 /*Deferred*/, 1 /*Dflt Ora Ver*/, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);

/* Call the insert_vals function to insert values into the table. */
insert_vals();

lrd_stmt(Csr1, "select * from Test_1\n", -1, 1 /*Deferred*/, 1 /*Dflt Ora Ver*/, 0);
lrd_bind_col(Csr1, 1, &NAME_D11, 0, 0);
lrd_bind_col(Csr1, 2, &ID_D12, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_fetch(Csr1, -4, 15, 0, PrintRow14, 0);
…
```

Note: You can specify a full path for the DLL. If you do not specify a path, lr_load_library searches for the DLL using the standard sequence used by the C++ function, LoadLibrary on Windows platforms. On UNIX platforms you can set the LD_LIBRARY_PATH environment variable (or the platform equivalent). The **lr_load_dll** function uses the same search rules as *dlopen*. For more information, see the main pages for dlopen or its equivalent.

Loading a DLL Globally

You can load a DLL globally, to make its functions available to all your Vuser scripts. Once the DLL is loaded, you can call any function defined within the DLL, without having to declare it in your script.

To call a function defined in a DLL:

1 Add a list of the DLLs you want to load to the appropriate section of the *mdrv.dat* file, located in your application's *dat* directory.

Use the following syntax,

```
PLATFORM_DLLS=my_dll1.dll, my_dll2.dll, …
```

replacing the word *PLATFORM* with your specific platform. For a list of platforms, see the beginning section of the *mdrv.dat* file.

For example, to load DLLs for Winsocket Vusers on an NT platform, add the following statement to the mdrv.dat file:

```
[WinSock]
ExtPriorityType=protocol
WINNT_EXT_LIBS=wsrun32.dll
WIN95_EXT_LIBS=wsrun32.dll
LINUX_EXT_LIBS=liblrs.so
SOLARIS_EXT_LIBS=liblrs.so
HPUX_EXT_LIBS=liblrs.sl
AIX_EXT_LIBS=liblrs.so
LibCfgFunc=winsock_exten_conf
UtilityExt=lrun_api
ExtMessageQueue=0
ExtCmdLineOverwrite=-WinInet No
ExtCmdLineConc=-UsingWinInet No
WINNT_DLLS=user_dll1.dll, user_dll2.dll, …
```

2 Call the function defined in the DLL in the appropriate place within your script.

B

Working with Foreign Languages

VuGen supports multilingual environments, allowing you to use languages other than English on native language machines when creating and running scripts.

This appendix includes:

- ➤ About Working with Foreign Languages on page 573
- ➤ Manually Converting String Encoding on page 574
- ➤ Converting String Encoding In Parameter Files on page 575
- ➤ Setting the String Encoding for Web Record and Replay on page 577
- ➤ Specifying a Language for the Accept-Language Header on page 580
- ➤ Protocol Limitations on page 581
- ➤ Quality Center Integration on page 582

About Working with Foreign Languages

When working with languages other than English, the primary issue is ensuring that VuGen recognizes the encoding of the text during record and replay. The encoding applies to all texts used by the script. This includes texts in HTTP headers and HTML pages for Web Vusers, data in parameter files, and others.

Windows 2000 and higher lets you save text files with a specific encoding directly from Notepad: ANSI, Unicode, Unicode big endian, or UTF-8.

By default, VuGen works with the local machine encoding (ANSI). Some servers working with foreign languages, require you to work with UTF-8 encoding. To work against this server, you must indicate in the Advanced recording options, that your script requires UTF-8 encoding.

Manually Converting String Encoding

You can manually convert a string from one encoding to another (UTF-8, Unicode, or locale machine encoding) using the **lr_convert_string_encoding** function. The syntax of the function is:

```
lr_convert_string_encoding(char * sourceString, char * fromEncoding, char *
toEncoding, char * paramName)
```

The function saves the result string (including its terminating NULL) in the third argument, *paramName*. It returns a 0 on success and -1 on failure.

The format for the fromEncoding and toEncoding arguments are:

LR_ENC_SYSTEM_LOCALE	NULL
LR_ENC_UTF8	"utf-8"
LR_ENC_UNICODE	"ucs-2"

In the following example, lr_convert_string_encoding converts "Hello world" from the system locale to Unicode.

```
Action()
{
   int rc = 0;
   unsigned long converted_buffer_size_unicode = 0;
   char              *converted_buffer_unicode = NULL;

   rc = lr_convert_string_encoding("Hello world", NULL, LR_ENC_UNICODE,
"stringInUnicode");
   if(rc < 0)
   {
      // error
   }
return 0;
}
```

In the Execution log, the output window shows the following information:

```
Output:
Starting action Action.
Action.c(7): Notify: Saving Parameter "stringInUnicode = H\x00e\x00l\x00l\x00o\x00
\x00w\x00o\x00r\x00l\x00d\x00\x00\x00"
Ending action Action.
```

The result of the conversion is saved to the *paramName* argument.

Converting String Encoding In Parameter Files

The parameter file contains the data for parameters that were defined in the script. This file, stored in the script's directory, has a **.dat* extension. When running a script, Vusers use the data to execute actions with varying values.

By default, VuGen saves the parameter file with your machine's encoding. When working with languages other than English, however, in cases where the server expects to receive the string in UTF-8, you may need to convert the parameter file to UTF-8. You can do this directly from Notepad, provided that you are working with Windows 2000 or higher.

To apply UTF-8 encoding to a parameter file:

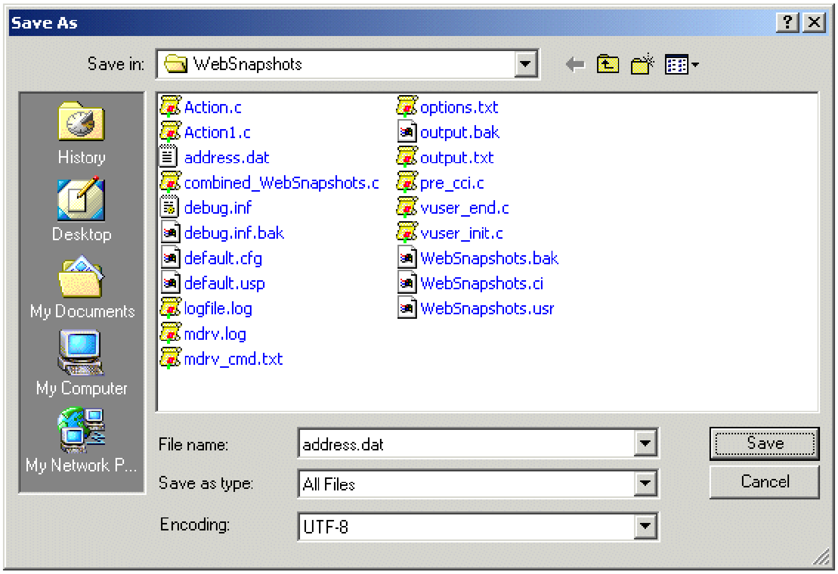
1 Select **Vuser** > **Parameter List** and view the parameter properties.

2 In the right pane, locate the parameter file in the **File path** box.

3 With the parameter table in view, click **Edit in Notepad**. Notepad opens with the parameter file in csv format.

4 In the **Save as type** box, select *All Files*.

In the **Encoding** box, select *UTF-8* type encoding.



File name: address.dat

Save as type: All Files

Encoding: UTF-8

5 Click **Save**. Notepad asks you to confirm the overwriting of the existing parameter file. Click **Yes**.

VuGen now recognizes the parameter file as UTF-8 text, although it still displays it in regular characters.

Setting the String Encoding for Web Record and Replay

When working with Web or other Internet protocols, you can indicate the encoding of the Web page text for recording. The recorded site's language must match the operating system language. You cannot mix encodings in a single recording—for example, UTF-8 together with ISO-8859-1 or shift_jis.

This section discusses:

- Encoding Recording Option
- Manually Enabling Encoding
- Browser Configuration

Encoding Recording Option

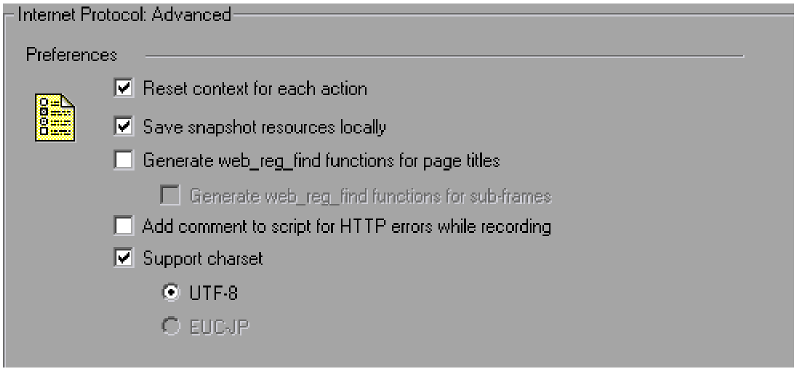
In order to be recognized as a non-English Web page, the page must indicate the charset in the HTTP header or in the HTML meta tag. Otherwise, VuGen will not detect the EUC-JP encoding and the Web site will not be recorded properly. To instruct VuGen to record non-English requests as **EUC-JP** or **UT-8**, select the appropriate option in the Recording Options dialog box, **HTPP Properties: Advanced** node.

- **UTF-8.** This option enables support for UTF-8 encoding. This instructs VuGen to convert non-ASCII UTF-8 characters to the encoding of your locale’s machine in order to display them properly in VuGen.
- **EUC-JP.** For users of Japanese Windows, select this option to enable support for Web sites that use EUC-JP character encoding. This instructs VuGen to convert EUC-JP strings to the encoding of your locale’s machine in order to display them properly in VuGen. VuGen converts all EUC-JP (Japanese UNIX) strings to the SJIS (Japanese Windows) encoding of your locale machine, and adds a **web_sjis_to_euc_param** function to the script (Kanji only).

Note that by selecting the **EUC-JP** or **UTF-8** option in the Recording Options, you are forcing VuGen to record a Web page with the selected encoding, even when it uses different encoding. If, for example, a non-EUC encoded Web page is recorded as EUC-JP, the script will not replay properly.

To enable the charset Encoding:

1 Open the Recording Options (Ctrl+F7) and select the **Advanced** node.



Internet Protocol: Advanced

Preferences

☑ Reset context for each action

☑ Save snapshot resources locally

□ Generate web_reg_find functions for page titles

□ Generate web_reg_find functions for sub-frames

□ Add comment to script for HTTP errors while recording

☑ Support charset

● UTF-8

○ EUC-JP

2 Select **Support charset**. Select the desired character encoding—UTF-8 or EUC-JP (only enabled for the Kanji operating system).

3 Click **OK**.

For more information about these settings, see "Recording Options for Selected Protocols" in *Volume II-Protocols* .

Manually Enabling Encoding

You can manually add full support for recording and replaying of HTML pages encoded in EUC-JP using the **web_sjis_to_euc_param** function. This also allows VuGen to display Japanese EUC-encoded characters correctly in Vuser scripts.

When you use **web_sjis_to_euc_param**, VuGen shows the value of the parameter in the Execution Log using EUC-JP encoding. For example, when you replay the **web_find** function, VuGen displays the encoded values. These include string values that were converted into EUC by the **web_sjis_to_euc_param** function, or parameter substitution when enabled in the **Run-Time Setting** > **Log** > **Extended Log**.

Browser Configuration

If, during recording, non-English characters in the script are displayed as escaped hexadecimal numbers (For example, the string "Ü&" becomes "%DC%26"), you can correct this by configuring your browser not to send URLs in UTF-8 encoding. In Internet Explorer, select **Tools** > **Internet Options and** click the **Advanced** tab. Clear the **Always Send URLs as a UTF-8** option in the Browsing section.

For more information about **web_sjis_to_euc_param**, see the *Online Function Reference*.

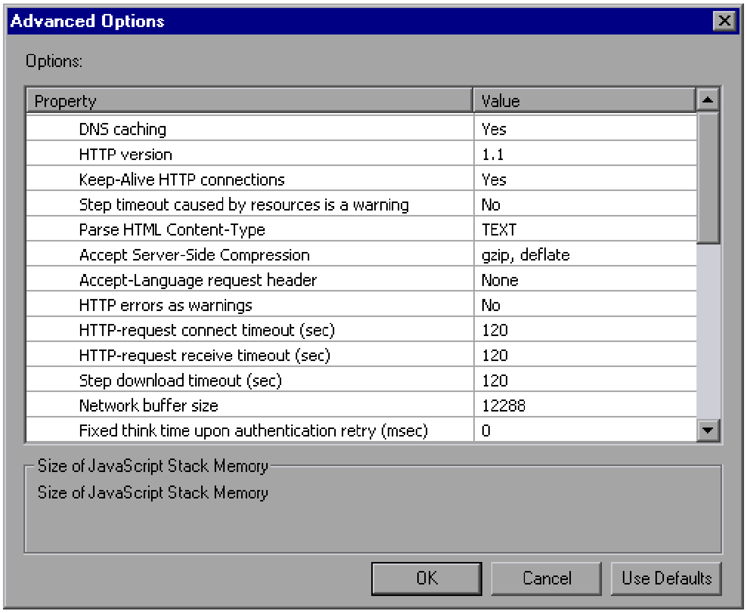
Specifying a Language for the Accept-Language Header

Before running a Web script, you can set the page's request header to match your current language. In the Internet Protocol Run-Time settings, you set the language of the *Accept-Language* request header. This header provides the server with a list of all of the accepted languages.

To set the Accept-Language header:

1 Open the Run-Time settings (F4) and select the **Internet Protocols:Preferences** node.

2 In the Advanced section, click the **Options** button to open the Advanced Options dialog box.



Advanced Options

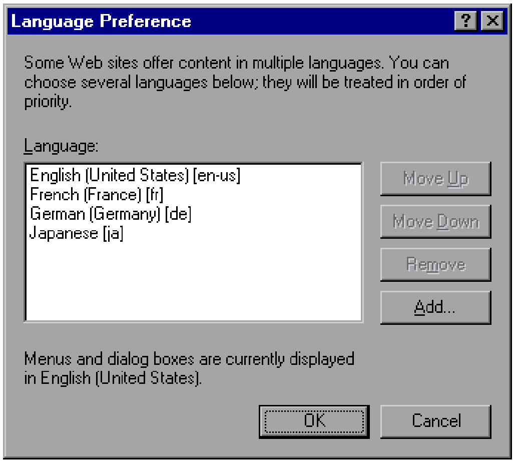
Options:

Property	Value
DNS caching	Yes
HTTP version	1.1
Keep-Alive HTTP connections	Yes
Step timeout caused by resources is a warning	No
Parse HTML Content-Type	TEXT
Accept Server-Side Compression	gzip, deflate
Accept-Language request header	None
HTTP errors as warnings	No
HTTP-request connect timeout (sec)	120
HTTP-request receive timeout (sec)	120
Step download timeout (sec)	120
Network buffer size	12288
Fixed think time upon authentication retry (msec)	0

Size of JavaScript Stack Memory

Size of JavaScript Stack Memory

3 Locate the **Accept-Language request header** option. In the **Value** column, select the desired language from the list. This list is derived from the Internet Options Language settings in your browser.



Language Preference

Some Web sites offer content in multiple languages. You can choose several languages below; they will be treated in order of priority.

Language:

English (United States) [en-us]
French (France) [fr]
German (Germany) [de]
Japanese [ja]

Menus and dialog boxes are currently displayed in English (United States).

For more information about these settings, see "Run-Time Settings for Selected Protocols" in *Volume II-Protocols*.

Protocol Limitations

SMTP Protocol

If you work with SMTP protocol through MS Outlook or MS Outlook Express, the Japanese text recorded in a Vuser script is not displayed correctly. However, the script records and replays correctly.

Script Name Length

When recording in COM, FTP, IMAP, SMTP, POP3, REAL or VB in VBA mode, the length of the script name is limited to 10 multi-byte characters (21 bytes).

Quality Center Integration

To open a script saved in a Quality Center project from VuGen, or a scenario saved in a Quality Center project from Controller, add a new Test Set named "Default" (in English) to the Quality Center project.

C

Programming Scripts on UNIX Platforms

Vusers on UNIX platforms can create scripts through programming. To create a script through programming, you use a template.

This appendix includes:

- About Programming Vuser Scripts to Run on UNIX Platforms on page 583
- Generating Templates on page 584
- Programming Vuser Actions on page 585
- Configuring Vuser Run-Time Settings on page 587
- Defining Transactions and Rendezvous Points on page 592
- Compiling Scripts on page 592

About Programming Vuser Scripts to Run on UNIX Platforms

There are two ways to create Vuser scripts that run on UNIX platforms: by using VuGen, or by programming.

VuGen You can use VuGen to create Vuser scripts that run on UNIX platforms. You record your application in a Windows environment and run it in UNIX—recording is not supported on UNIX.

programming Users working in UNIX-only environments can program Vuser scripts. Scripts can be programmed in C or C++ and they must be compiled into a dynamic library.

This appendix describes how to develop a Vuser script by programming.

To create a script through programming, you can use a Vuser template as a basis for a larger Vuser scrips. The template provides:

- correct program structure
- Vuser API calls
- source code and makefiles for creating a dynamic library

After creating a basic script from a template, you can enhance the script to provide run-time Vuser information and statistics. For more information, see Chapter 6, "Enhancing Vuser Scripts."

Generating Templates

VuGen includes a utility that copies a template into your working directory. The utility is called mkdbtest, and is located in $M_LROOT/bin. You run the utility by typing:

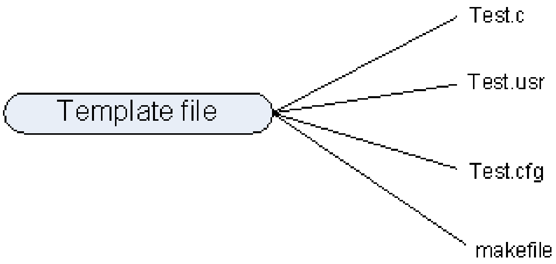
```
mkdbtest name
```

When you run mkdbtest, it creates a directory called name, which contains the template file, name.c. For example, if you type:

```
mkdbtest test1
```

mkdbtest creates a directory called *test1*, which contains the template script, *test1.c*.

When you run the *mkdbtest* utility, a directory is created containing four files *test.c*, *test.usr*, *test.cfg* and *Makefile*, where *test* is the test name you specified for mkdbtest.



Programming Vuser Actions

The Vuser script files, *test*.c, *test*.usr, and *test*.cfg, can be customized for your Vuser.

You program the actual Vuser actions into the *test.c* file. This file has the required structure for a programmed Vuser script. The Vuser script contains three sections: vuser_init, Actions, and vuser_end.

Note that the template defines extern C for users of C++. This definition is required for all C++ users, to make sure that none of the exported functions are modified inadvertently.

```
#include "lrun.h"
#if defined(__cplusplus) || defined(cplusplus) extern "C"
{
#endif
int LR_FUNC vuser_init(LR_PARAM p)
{
                lr_message("vuser_init done\n");
                return 0;
}
 int Actions(LR_PARAM p)
{
                lr_message("Actions done\n");
                return 0;
}
 int vuser_end(LR_PARAM p)
{
                lr_message("vuser_end done\n");
                return 0;
}
#if defined(__cplusplus) || defined(cplusplus)}
#endif
```

You program Vuser actions directly into the empty script, before the **lr_message** function of each section.

The *vuser_init* section is executed first, during initialization. In this section, include the connection information and the logon procedure. The *vuser_init* section is only performed once each time you run the script.

The *Actions* section is executed after the initialization. In this section, include the actual operations performed by the Vuser. You can set up the Vuser to repeat the Actions section (in the *test*.cfg file).

The *vuser_end* section is executed last, after the all of the Vuser's actions. In this section, include the clean-up and logoff procedures. The *vuser_end* section is only performed once each time you run the script.

Note: LoadRunner controls Vusers by sending SIGHUP, SIGUSR1, and SIGUSR2 UNIX signals. Do not use these signals in your Vuser programs.

Configuring Vuser Run-Time Settings

To configure Vuser run-time settings, you modify the *default.cfg* and *default.usp* files created with the script. These run-time settings correspond to VuGen's run-time settings. (See Chapter 22, "Configuring Run-Time Settings".) The *default.cfg* file contains the setting for the General, Think Time, and Log options. The *default.usp* file contains the setting for the Run Logic and Pacing.

General Options

There is one General options for Unix Vuser scripts:

ContinueOnError instructs the Vuser to continue when an error occurs. To activate the option, specify 1. To disable the option, specify 0.

In the following example, the Vuser will continue on an error.

```
[General]
ContinueOnError=1
```

Think Time Options

You can set the think time options to control how the Vuser uses think time during script execution. You set the parameters Options, Factor, LimitFlag, and Limit parameters according to the following chart.

Option	Options	Factor	LimitFlag	Limit
Ignore think time	NOTHINK	N/A	N/A	N/A
Use recorded think time	RECORDED	1.000	N/A	N/A
Multiply the recorded think time by...	MULTIPLY	number	N/A	N/A
Use random percentage of recorded think time	RANDOM	range	lowest percentage	upper percentage
Limit the recorded think time to...	RECORDED / MULTIPLY	number (for MULTIPLY)	1	value in seconds

To limit the think time used during execution, set the LimitFlag variable to 1 and specify the think time Limit, in seconds.

In the following example, the settings tell the Vuser to multiply the recorded think time by a random percentage, ranging from 50 to 150.

```
[ThinkTime]
Options=RANDOM
Factor=1
LimitFlag=0
Limit=0
ThinkTimeRandomLow=50
ThinkTimeRandomHigh=150
```

Log Options

You can set the log options to create a brief or detailed log file for the script's execution.

```
[Log]
LogOptions=LogBrief
MsgClassData=0
MsgClassParameters=0
MsgClassFull=0
```

You set the parameters LogOptions, MsGClassData, MsgClassParameters, and MsgClassFull variables according to the following chart:

Logging Type	LogOptions	MsgClassData	MsgClassParameters	MsgClassFull
Disable Logging	LogDisabled	N/A	N/A	N/A
Standard Log	LogBrief	N/A	N/A	N/A
Parameter Substitution (only)	LogExtended	0	1	0
Data Returned by Server (only)	LogExtended	1	0	0
Advanced Trace (only)	LogExtended	0	0	1
All	LogExtended	1	1	1

In the following example, the settings tell the Vuser to log all data returned by the server and the parameters used for substitution.

```
[Log]
LogOptions=LogExtended
MsgClassData=1
MsgClassParameters=1
MsgClassFull=0
```

Iterations and Run Logic

You can set the Iteration options to perform multiple iterations and control the pacing between the iterations. You can also manually set the order of the actions and their weight. To modify the run logic and iteration properties of a script, you must edit the *default.usp* file.

To instruct the Vuser to perform multiple iterations of the Actions section, set RunLogicNumOfIterations to the appropriate value.

To control the pacing between the iterations, set the RunLogicPaceType variable and its related values, according to the following chart:

Pacing	RunLogicPaceType	Related Variables
As soon as possible	Asap	N/A
Wait between Iterations for a set time	Const	RunLogicPaceConstTime
Wait between iterations a random time	Random	RunLogicRandomPaceMin, RunLogicRandomPaceMax
Wait after each iteration a set time	ConstAfter	RunLogicPaceConstAfterTime
Wait after each iteration a random time	After	RunLogicAfterPaceMin, RunLogicAfterPaceMax

In the following example, the settings tell the Vuser to perform four iterations, while waiting a random number of seconds between iterations. The range of the random number is from 60 to 90 seconds.

```
[RunLogicRunRoot]
MercIniTreeFather=""
MercIniTreeSectionName="RunLogicRunRoot"
RunLogicRunMode="Random"
RunLogicActionOrder="Action,Action2,Action3"
RunLogicPaceType="Random"
RunLogicRandomPaceMax="90.000"
RunLogicPaceConstTime="40.000"
RunLogicObjectKind="Group"
RunLogicAfterPaceMin="50.000"
Name="Run"
RunLogicNumOfIterations="4"
RunLogicActionType="VuserRun"
RunLogicAfterPaceMax="70.000"
RunLogicRandomPaceMin="60.000"
MercIniTreeSons="Action,Action2,Action3"
RunLogicPaceConstAfterTime="30.000"
```

Defining Transactions and Rendezvous Points

When programming a Vuser script without VuGen, you must manually configure the Vuser file in order to enable transactions and rendezvous. The configuration settings are listed in the test.usr file.

```
[General]
Type=any
DefaultCfg=Test.cfg
BinVuser=libtest.libsuffix
RunType=Binary

[Actions]
vuser_init=
Actions=
vuser_end=

[Transactions]
transaction1=

[Rendezvous]
Meeting=
```

Each transaction and rendezvous must be defined in the *usr* file. Add the transaction name to the Transactions section (followed by an "="). Add each rendezvous name to the Rendezvous section (followed by an "="). If the sections are not present, add them to the *usr* file as shown above.

Compiling Scripts

After you modify the template, you compile it with the appropriate *Makefile* in the script's directory. Note that for C++ compiling, you must use the native compiler (not gnu). The compiler creates a dynamic library called:

- libtest.so (solaris)
- libtest.a (AIX)
- libtest.sl (HP)

You can modify the *Makefile* and assign additional compiler flags and libraries by modifying the appropriate sections.

If you are working with a general template, you must include your application's libraries and header files. For example, if your application uses a library called testlib, include it in the LIBS section.

```
LIBS        = \
     -testlib \
     -lLrun50 \
     -lm
```

After you modify the *makefile*, type Make from the command line in the working directory to create the dynamic library files for the Vuser script.

After you create a script, you check it's functionality from the command line.

To run a Vuser script from the UNIX command line, type:

```
mdrv -usr ‘pwd’ test.usr
```

where *pwd* is the full path to the directory containing the Vuser script and *test*.usr is the name of the Vuser file. Check that your script communicates with the server and performs all the required tasks.

After you verify that your script is functional, you integrate it into your environment: a LoadRunner scenario, Performance Center load test, or Business Process Monitor profile. For more information, see the *HP LoadRunner Controller*, *Performance Center*, or *HP Business Availability Center* documentation.

D

Using Keyboard Shortcuts

The following list describes the keyboard shortcuts available in the Virtual User Generator.

ALT+F8	Compares the Current Snapshots (Web Vusers only)
ALT+INS	Create New Step
CTRL+A	Select All
CTRL+C	Copy
CTRL+F	Find
CTRL+G	Go To Line
CTRL+H	Replace
CTRL+N	New
CTRL+O	Open
CTRL+P	Print
CTRL+S	Save
CTRL+V	Paste
CTRL+X	Cut
CTRL+Y	Redo
CTRL+Z	Undo
CTRL+F7	Recording Options
CTRL+F8	Scan for Correlations

CTRL+SHIFT+SPACE	Show Function Syntax (Intellisense)
CTRL+SPACE	Complete Wizard (completes the function name)
F1	Help
F3	FIND Next Downward
SHIFT+F3	Find Next Upward
F4	Run-Time Settings
F5	Run Vuser
F6	Move Between Panes
F7	Show EBCDIC Translation Dialog (for WinSocket data)
F9	Toggle Breakpoint
F10	Run Vuser Step by Step

Index

A

ABC icon 221
accept server-side compression 455
Accept-Language request header 455, 580
Action
 section 82
actions
 importing 101
 recording multiple 95
 reordering 102
 set opening mode 47
Add new column dialog box 239
Additional Attributes run-time setting 431
Adobe Reader 15
advanced code generation options
 RDP 290
Advanced GUI dialog box 345
Advanced recording options 332
Advanced run-time settings (RDP) 473
Agent run-time settings (RDP) 474
allocating Vuser values
 data files 245, 261
 data tables 248
AMF Vuser scripts
 setting recording mode 304
analyzing run results. See run results
animated run
 defined 140
 enabling 142
ANSI 320
arrays
 duplicate in XML 258
AUT Configuration 480
authentication
 connecting to Quality Center 181
authentication during recording 102
authentication retry think time 457
auto recovery 36
auto-detect protocol 403
automatic proxy configuration script 444
automatic transactions
 enabling 437
 transaction names 453

B

basic code generation options
 RDP 288
Basic event recording configuration level 349
behavior, DHTML 355
block size, allocating Vuser values 245, 261
bookmarks
 in Vuser script 151
braces, using in parameterization 233
Breakpoint Manager 149
breakpoints 148
brief log run-time setting 427
browser cache (Web, Wireless) 450
Browser Emulation settings, Web 447
bubbling, in Web events 356
buffer size on network (Internet) 456

C

C functions
 additional keywords 530
 for debugging 530
 using in Vuser scripts 54
C language support
 interpreter 55
cache
 check for newer versions 450
 clear each iteration 450
character encoding 577

character set, RTE 320
Check In command 198
checking-in scripts for version control 198
check-out operation, cancelling 203
Citrix Run Time Settings
  Synchronization settings 477
Citrix Vuser scripts
  recording options 299
  run-time settings 475
Citrix, *See Volume II - Protocols*
Classpath
  run-time settings 490
client emulation run-time setting 482
Close All command 153
Code Generation options
  AMF 308
  Citrix 295
  EJB 286
  Flex 309
  SAPGUI 326
COM, *See Volume II - Protocols*
command line arguments
  reading in C Vuser scripts 123
  UNIX Vuser scripts 156
command prompt 156
Commands tab (Customize dialog box) 33
Comment button 114
comments
  inserting into Vuser scripts 114
  screen header comments (RTE) 323
comparing
  Vusers 136
comparison tool, configuring 37
compiling
  Vuser scripts in UNIX 592
components
  run results. See run results
compression for HTML (gzip) 463
compression headers, requesting 455
concurrency settings, Web Services 480
configuration levels
  customizing 351–359
  standard 349–351
Configuration run-time settings (RDP) 470
connecting
  VuGen to Quality Center 188
connecting to Quality Center 181
connection attempts, modifying (RTE) 492
connection pooling 480
content check
  limit errors 457
  settings (Web) 464
content type filtering (Web) 337
Content type filters dialog box 337
continuing on error
  globally 433
  locally 120
Controller
  scenario 160
CORBA Recording Options dialog box 373
correlating
  debugging tips for Siebel 544
  functions (C) 133
  functions (Java) 135
  modifying existing parameters 138
  overview 131
  recording options-Java 370
  scripting language options 379
Correlation options, for Script recording options 379
Correlation tab, recording options 330
CtLib
  logging server messages 428
  options 301
CtLib, *See Also Volume II - Protocols*
custom event-recording configuration 351–359
  adding listening events 354
  procedure 353
  specifying listening criteria 355
custom headers, for Web and Wireless 334
custom requests 394
Custom Web Event Recording Configuration dialog box 353
Customize dialog box
  Commands tab 33
  Keyboard tab 34
  Options tab 35
  Toolbars tab 33
  Tools tab 34

D

data assignment methods, in parameterization 248
data file parameters
  adding rows and columns 239
  creating a data source 238
  editing 240
  importing data from database 240
  importing data source using data wizard 239
  selecting data source 238
data files
  used for parameterization 224
data grids
  enabling 153
data table parameters
  adding rows and columns 239
  creating a data source 238
  editing 240
  importing data from database 240
  importing data source using data wizard 239
  selecting data source 238
Data Wizard, SQL statement 242
Database Query Wizard dialog box 241
Database recording options 300
Database Vuser scripts
  tips 539
Database Vuser scripts, *See Also Volume II-Protocols*
date/time, parameter values 265
DBCS 320
Debug Message dialog box 118
Debug recording settings (Java) 371
debugging
  database applications 531
  during replay 148
  enabling debugging features 143
  enabling for Web Vusers 143
  obtaining information (WAP) 462
  Oracle applications 533
  setting debug level 427
decrypting text 124
defining parameter properties
  data files 244
  general 226
  tables 246
delimiter of columns
  in data files 244
  in data tables 246
device name (RTE) 492
diagnostics
  enabling in VuGen 437
directory of script, opening 98
disable logging log option 425
disconnecting from Quality Center 191
Display tab, General options 144
DLLs, calling from a Vuser script 569
DNS caching
  Web 455
DNS, *See Volume II - Protocols*
documentation updates 18
documentation, online 15
DOM memory allocation 460
download filter 461
downloading
  from Performance Center to VuGen 211
DSL 441
duplicate key violations
  Oracle, MSSQL 541
  Siebel 544

E

editor, setting font for 36
EJB
  code generation options 286
encoding
  EUC 394
  passwords 125
encrypting text 124
engine, recording 334
Environment run-time settings, .NET 479
Environment tab 36
error handling
  global or local setting 120
  run-time setting 433
error matches, limiting 457
Error Message dialog box 119
errors, generate snapshot on 433
EUC encoded pages 394

EUC encoding 396
EUC-JP encoding
- recording option 333
- setting 577

event-recording configuration 348–359
- customizing levels 351
- resetting 358
- standard levels 349

Export to HTML File dialog box 179
exporting script
- to Word file 127
- to zip file 97

extended log option 426
extended result set 302
external functions 569
externalizable objects 308, 309

F

files, adding to script 126
filtering
- .NET Vusers 311
- content type (Web, Wireless) 337
- downloaded resources 461

filters in .NET
- benefits 311

find
- find in files 154

Find dialog box
- Test Results 175

font in editor 36
format
- for parameterization 275

FTP, *See Volume II - Protocols*
full run-time trace 427
functions
- automatic word completion 57
- getting help 56
- lr (C functions) 54
- syntax 58

G

Gateway settings (WAP) 494
General options
- all Vusers 234
- dialog box 235
- Display tab (Web only) 144
- Environment tab 36
- Parameterization tab 233
- Replay tab 142

generate
- automatic sync points (RDP) 289
- mouse movement calls 289

Generate snapshot on error 433
Generation Log tab 99
global directory 234
go to command 152
graphs
- enabling for Web 453

grids
- hiding 153

group name, parameter values 267
GUI Vuser scripts
- tools for 27

gzip 463

H

handler 355
header files 58
headers
- custom 334
- risky 334

High event recording configuration level 350
history object, support for 459
host suffix, filtering by 461
HP Software Support Web site 17
HP Software Web site 18
HTML tag object 351
HTML view (Web snapshots) 44
HTML-based mode 382
HTTP
- buffer size (Web) 456

HTTP recording mode, WAP 495

I

If-Modified-Since header
- Web 450

ignore assemblies 317
IMAP, *See Volume II - Protocols*

i-mode, *See Volume II - Protocols*
importing
- actions 101
- data from a database 240

Insert Comment dialog box 114
intellisense 57
internal data, parameterization 225, 264
ISDN 441
iteration number, parameter values 268
iterations
- run-time settings 421
- updating parameters for each 276

J

Jacada, *See Volume II - Protocols*
Java virtual machine
- recording options 363
- run-time settings 489

Java Vuser scripts
- CORBA recording options 373
- correlation options 370
- debug options 371
- Recorder options 366
- serialization options 368

Java Vusers
- Classpath run-time settings 490
- Java VM run-time settings 489
- recording options, correlation 370
- recording options, Java VM 363
- recording options, serialization 368

Java Vusers (all)
- run-time settings 489–491

Java Vusers, *See Volume II-Protocols*
Java, *See Volume II-Protocols*
Jscript 377

K

keep-alive connections, Web 454
Kerebros
- authentication 457

keyboard shortcut
- recording options 380
- run-time settings 414
- shortcuts list 595

Keyboard tab 34
keywords, adding additional 530
Knowledge Base 17

L

language encoding 577
language for script generation 375
language headers 580
LDAP, *See Volume II - Protocols*
level of script generation, RDP 288, 290
libraries, for scripting 438
license information 26
load generator name, parameter values 269
loading DLLs
- globally 571
- locally 569
- overview 569

LoadRunner Analysis User's Guide 16
LoadRunner Controller User's Guide 16
LoadRunner Installation Guide 16
LoadRunner Monitor Reference 16
log
- setting detail level - PC 426
- setting detail level - UNIX 589

log cache size 425
Log Message dialog box 117
Log run-time settings 424
lrbin.bat utility 510

M

Mailing services, *See Volume II - Protocols*
managing
- script versions in Quality Center 195

MAPI, *See Volume II - Protocols*
Media Player, *See Volume II - Protocols*
Medium event recording cfg level 350
memory allocation for DOM 460
memory management 460
messages
- sending to output 116

META refresh 457
Microsoft .NET Vuser scripts
- recording options 311
- run-time settings 479

Miscellaneous run-time settings 432
mkdbtmpl script (UNIX) 584
MMS Vuser scripts
  run-time settings 500
modifier keys 288
MS Query 241
multi-action 84
multilingual support 573–582
  parameter files 575
multi-protocol 84
multi-threading 436

N

network settings 441
Network Speed, run-time setting 441
non-resources 338
NTLM
  authentication 102
  security 457

O

online browser 152
online documentation 15
online resources 17
Opening Scripts from a Quality Center Project 192
Opening Tests from the Recent Files List 193
optional elements
  excluding 256
options
  CtLib 301
  lrd log 301
  recording (RTE) 321
Options tab 35
Oracle application debugging 533
Oracle NCA Vuser scripts
  run-time settings 482
Oracle Web Applications 11i Vuser scripts
  advanced GUI-based options 345
Oracle, *See Also Volume II-Protocols*
Output Message dialog box 119
output parameters
  selecting 230
  XML 526
Output window
  hiding 145
  Replay tab 145
  RunTime Data tab 146
  show/hide 153

P

Pacing settings 421
page view (Web snapshots) 44
parameter formats
  adding 275
  deleting 276
  restoring original 276
Parameter Properties dialog box 226
parameter types
  data files 224, 264
  data tables 264
  date/time 265
  group name 267
  internal data 263, 264
  internal data types 225
  iteration number 268
  load generator name 269
  output 230
  random number 270
  tables 224
  understanding 223
  unique number 271
  user-defined functions, format 225
  user-defined functions, overview 264
  user-defined functions, properties 274
  Vuser ID 273
  xml 225
parameterization
  assigning values from files and tables 248
  brace style 223
  braces in script 233
  COM, .NET, VB 220
  creating a new parameter 220
  data files 224
  defining properties 226
  global directory 234
  internal data properties 264
  internal data type formats 275

- Java 221
- limitations 219
- modifying existing parameters 138
- naming a parameter 221
- overview 218
- parameter list 230
- random sequence with seed 249
- restoring original value 230
- setting properties for data files 244
- setting properties for tables 246
- simulating 277
- tables 224
- undoing (Web) 230
- updating parameter values 276
- updating with unique values 249
- user-defined functions 225
- using internal data 225
- using user-defined functions 274
- UTF-8 encoded 575
- xml 225

Parameterization Options 233
parameterization, replacing long string 378
parameters
- creating in Script view 220
- creating in Tree view 221
- creating using Parameter List 231
- deleting 232
- modifying 231

Password Encoder dialog box 125
password, encoding 125
pausing a Vuser 145
PeopleSoft Enterprise Vuser scripts
- advanced GUI-based options 345

PeopleSoft, *See Volume II - Protocols*
PeopleSoft-Tuxedo Vusers, running 548
Performance Center
- connecting to 206
- managing Vuser scripts 205

persistent connections, Web 454
pooling of connections 480
POP3, *See Volume II-Protocols*
Port Mapping settings 398
Pragma mode 482
Preferences run-time settings 452
Print dialog box, Test Results window 176
Print Preview dialog box 177
programming
- in Visual Studio 505
- using templates 584
- using Visual Basic templates 508
- using Visual C templates 506
- Vuser actions 585

properties of parameters
- defining 226
- defining for data files 244
- defining for tables 246

protocol
- advisor 61

protocol advisor 61
protocol detection 61
protocols *See Volume II - Protocols*
protocols, list of supported 88
proxy server
- run-time settings (Internet) 442

Q

Quality Center
- connecting to 188
- connecting to a project 181
- disconnecting from 191
- managing scripts with 187
- managing versions 195
- managing Vuser scripts 187
- opening a Vuser script 192
- saving scripts to 194
- version control for 195

R

Radius
- run-time settings (WAP) 498
- support 498

radius, for synchronization 289, 290
random number, parameter values 270
random parameter assignment 249
raw keyboard and mouse calls (RDP) 289
RDP agent
- recording options 291

RDP Vuser scripts
- recording options 287
- run-time settings 470

RDP, *See Volume II - Protocols*
Record button 93
recording
  status, options 356
recording engine 334
Recording Log tab 99
recording options
  See Also Volume II - Protocols
  .NET Vusers 311
  Advanced (Web, Wireless) 332
  advanced code generation, RDP 290
  basic code generation, RDP 288
  Code Generation, AMF 308
  Code Generation, Citrix 295
  Code Generation, Flex 309
  CORBA Options 373
  Database 300
  Debug (Java) 371
  Java language 361–372
  keyboard shortcut 380
  Login (Citrix) 296
  Port Mapping 398
  RDP 287
  RDP agent 291
  RDP Login 287
  Recorder (Java) 366
  Recording (Web) 395
  Recording .NET 312
  RTE 321
  RTE Configuration 320
  SAPGUI 326
  Script (FTP, COM, Mail) 376
  Web 285, 381
recording Vuser scripts
  overview 81
recovery of lost scripts 36
regenerating Vusers
  all protocols 104
rendezvous
  Rendezvous dialog box 113
rendezvous points, inserting 113
Replace More Occurrences command 229
Replay tab, General Options dialog box 142
report tree, Results Summary (Web) 165
resources
  excluding in Web recordings 338
restoring original value of parameter 230
Result Details tab, Test Results window 167
result parameters, saving XML 526
Results Summary report
  overview 163
  sending custom messages 185
  tree branches 165
  Web Services Vusers 182
results. See run results
row count, obtaining 541
RTE Vuser scripts
  run-time settings 492
RTE, *See Volume II-Protocols*
Run command 144
Run Logic run-time settings 415
run results 163
  customizing display 182
  exporting to HTML 179
  filtering 173
  finding 174, 175
  previewing before printing 177
  printing 176
  schema 182
  Test Results window 165
  viewing for a selected run 169
run sessions
  printing results 176
run_db_vuser shell script 157
running Vuser scripts
  animated mode 140
  step by step 147
  using VuGen 139
run-time settings
  See Also Volume II - Protocols
  .NET environment 479
  Additional Attributes 431
  Advanced (RDP) 473
  Agent (RDP) 474
  all protocols 413, 469
  Browser Emulation node 447
  Client Emulation (Oracle NCA) 482
  Configuration (RDP) 470
  configuring manually 587
  ContentCheck node (Web) 464
  debug information (WAP) 462
  dialog box 414

Gateway node (WAP) 494
Java 489–491
keyboard shortcut 414
Log node 424
Miscellaneous 432
MMS 500
network 439
Oracle NCA 482
Pacing node 421
Preferences - Advanced 453
Preferences (Internet prtcls) 452
Proxy (Internet prtcls) 442
Radius (WAP) 498
RDP 470
Run Logic 415
screen properties 459
shortcuts 414
Speed Simulation 440
Synchronization (RDP) 471
Think Time 429
VBA (Visual Basic Apps) 438
run-time viewer
display options 143
enabling in VuGen 152

S

S_SSA_ID table 546
SAP, *See Volume II-Protocols*
SAPGUI Vuser scripts
auto logon recording options 327
code generation recording options 326
general recording options 325
setting recording options 325
SAP-Web Vuser scripts
recording options 329
scenario
create from VuGen 160
integrating Vuser scripts into 159
parameter simulation 280
schema, for run results 182
screen properties, run-time setting 459
script folder, opening 98
script level, RDP 288, 290
script mode 47
script view
displaying in 38
opening in 47
Search and Replace dialog box 229
sections of a Vuser script 82
Select or Create Parameter dialog box 220
Select Results Directory dialog box 142
sequential parameter assignment 248
Serialization options 369
Service Test 29
Shift Japan Industry Standard (SIJS) 394
show function 57
show function syntax 58
Siebel
See Also Volume II-Protocols
base 36 key values 546
scripting tips for 2-tier 544
SMTP, *See Volume II-Protocols*
snapshots
generate on error 433
save replay snapshot locally 454
saving in Citrix recording 294
Test Results window 164
Web page 41
Speed Simulation settings 440
split actions 377
SQL statement 242
standard event-recording configuration 349–351
standard log option 426
Start Recording dialog box 93
Start Transaction dialog box 111
Step button 147
stopping a Vuser 145
strings, replacing long with parameter 378
suffix values in Siebel 544
synchronization functions
generating for Citrix text 295
Synchronization run-time settings (RDP) 471
synchronizing Vuser scripts
rendezvous 121
syntax, show for function 58

T

table icon 223
tables
  used for parameterization 224
Tasks pane 68
template
  creating new 91
  opening in Visual C 506
  programming in C, UNIX 584
  programming using Visual Basic 508
  user-defined 89
test results
  viewing 168
  Web Services Vusers 182
Test Results report
  sending custom messages 185
Test Results toolbar, Test Results window 167
Test Results tree 166
Test Results window 165
  look and feel 168
  Result Details tab 167
  run results toolbar 167
  run results tree 166
  theme 168
Test Results, Web Services 182
TestDirector, see Quality Center 187
tests
  See also run results
text synchronization
  Citrix 295
think time
  defined 429
  inserting 122
  run-time settings 429
  threshold, Database 300
Think Time dialog box 122
threshold for setTimeout, setInterval 459
thumbnails
  annotating 49
  in workflow wizard 48
  renaming 49
  viewing 46
tiling windows 153
timeouts
  Citrix connection 477
  HTTP request 455
  WAP connection 455
timestamp (Database) 302
tips
  Database related 539
  Siebel specific 544
Toolbars tab 33
Tools tab 34
traffic forwarding 403
Transaction Editor 48, 72
transactions
  automatic, for Web Vuser scripts 437
  breakdown limitation for Oracle DB 83
  editor 72
  in output log 145
  inserting 110
  nested 76, 112
  Web Vusers 437
  wizard workflow 72
treeview
  all Vusers 40
  inserting steps 41
troubleshooting
  2-tier database 539
  Oracle applications 533
  Siebel Vusers 544
  VuGen 529
Troubleshooting and Knowledge Base 17

U

Undo Parameter command 230
unique column name 544
unique number, parameter values 271
unique value parameter assignment 249
UNIX
  command line 156
update methods
  parameter assignment 250
  parameter usage 276
updates, documentation 18
uploading
  scripts to Performance Center 208
URL-based mode 382
Use Existing Parameter command 228
user-agent browser emulation 448

user-defined function parameters
- format 225
- properties 274

Using the Version History Dialog Box 200
UTF-8 conversion
- in run-time settings 456
- recording option 333
- setting 577

V

value sets
- creating 255
- optional elements 256

VBA references 438
VBA run-time setting 438
verification checks
- Web 452

verify_generator 157
version control 195
- checking tests in to 198

version history 200
viewer, XML 52
Virtual User Generator, *See* VuGen
virtual users, defined 22
Visual Basic
- scripting option 377
- Vuser scripts 505

Visual C, using Visual Studio 505
Visual Log options (Web) 143
Visual Studio 505
VM (virtual machine) 363
VuGen
- environment options 36
- introducing 31
- recording Vuser scripts 81
- running Vuser scripts 53
- starting 32
- toolbar 96

Vuser functions
- automatic word completion 57
- external, user defined 569
- general (C) 54
- getting help for 56
- lr (C functions) 54
- syntax 58
- *See Also* Online Function Reference

Vuser ID, parameter values 273
Vuser information, obtaining 115
Vuser scripts
- adding functions 107
- comments, inserting 114
- creating on UNIX 583–593
- debugging features 147
- enhancing 107
- importing from zip 92
- opening 92
- parameterizing 217
- Performance Center upload/download 205
- programming 583–593
- Quality Center integration 187
- regenerating 104
- rendezvous points 113
- running 139
- running from command prompt 156
- run-time settings 413, 469
- run-time settings-Java 489–491
- scenario integration 159
- sections 82
- selecting generation language 375
- transactions 110
- types *See* Vuser types
- UNIX, compiling on 592
- UNIX, running on 156
- uploading to Performance Center 208
- version history 200
- viewing 38
- working from zip 92

Vuser types
- *See Also Volume II-Protocols*
- list of 25

vuser_init, vuser_end sections 82
Vusers
- introducing 22

W

WAP Vuser scripts
- debug information 462

WAP, *See Volume II-Protocols*
Wdiff 136

Web (Click and Script)
- general options 458
- history options 459
- memory management 460
- Navigator properties 459
- timers 458

Web (Click and Script) Vuser scripts
- advanced GUI-based options 345
- checking Web page content 464
- content filtering 337
- custom headers 334
- debugging features, enabling 143
- debugging tools 152
- Results Summary report 163
- run-time viewer 152
- selecting a recording level 382
- setting recording options 344
- viewing recorded functions 343
- Visual Log options 143

Web Event Recording Configuration dialog box 350

Web performance graphs
- generating for Web Vusers 453

Web Services Vuser scripts
- reporting tool 182

Web Services, *See Volume II-Protocols*

Web Vuser scripts
- *See Also Volume II-Protocols*
- content filtering 337
- custom headers 334
- debugging features, enabling 143
- debugging tools 152
- recording options 285, 381
- Results Summary report 163
- run-time settings 440
- run-time viewer 152
- setting Visual Log options 152
- Visual Log options 143

web_set_user function generation 102

Web-event-recording configuration 348–359
- customizing 351–359
- standard 349–351

window names, Citrix 293

WinInet engine (Internet protocols) 453

WinSock, *See Volume II-Protocols*

Wireless Vuser scripts
- custom headers 334

Wireless, *See Volume II-Protocols*

wizard, workflow 67

word completion 57

workflow wizard 67

workflow wizard, restoring 79

X

XML
- attributes, working with 519

XML API
- programming with 511

XML parameters
- creating 253

XML viewer 52

Z

zip file
- exporting to 97
- using 100
- working from 92

zlib headers 455