WebLOAD Recorder User's Guide

Version 12.0

WebLOAD Recorder User's Guide

November, 2018, RadView Publication Number WL-OSSPRO-0913-IUG10

For product assistance or information, contact:	
Toll free in the US: Fax: World Wide Web:	1-888-RadView +1-908-864-8099 www.RadView.com
North American Headquarters:	**International Headquarters:**
RadView Software Inc. 991 Highway 22 West, Suite 200 Bridgewater, NJ 08807 Email: info@RadView.com Phone: 908-526-7756 Fax: 908-864-8099 Toll Free: 1-888-RadView	RadView Software Ltd. 13 Haamal Street, Park Afek Rosh Haayin, 4809249, Israel Email: info@RadView.com Phone: +972-3-915-7060 Fax: +972-3-915-7011

Table of Contents

Chapter 1

Introduction

Welcome to WebLOAD, the premier performance, scalability, and reliability testing solution for internet applications.

WebLOAD is easy to use and delivers maximum testing performance and value. WebLOAD verifies the scalability and integrity of internet applications by generating a load composed of Virtual Clients that simulate real-world traffic. Probing Clients let you refine the testing process by acting as a single user that measures the performance of targeted activities, and provides individual performance statistics of the internet application under load.

This section provides a brief introduction to WebLOAD technical support, including both documentation and online support.

IMPORTANT NOTE: In previous WebLOAD versions, a WebLOAD script was called an “Agenda”. From version 12.0, it is referred to simply as a script. Wherever “Agenda” is still displayed, we are referring to the WebLOAD script

WebLOAD Recorder used to be called WebLOAD IDE.

WebLOAD Documentation

WebLOAD is supplied with the following documentation:

WebLOAD™ Installation Guide

Instructions for installing WebLOAD and its add-ons.

WebLOAD™ Recorder User’s Guide

Instructions for recording, editing, and debugging load test Scripts to be executed by WebLOAD to test your Web-based applications.

WebLOAD™ Console User's Guide

A guide to using WebLOAD console, RadView's load/scalability testing tool to easily and efficiently test your Web-based applications. This guide also includes a quick start section containing instructions for getting started quickly with WebLOAD using the RadView Software test site.

WebLOAD™ Analytics User's Guide

Instructions on how to use WebLOAD Analytics to analyze data and create custom, informative reports after running a WebLOAD test session.

WebRM™ User's Guide

Instructions for managing testing resources with the WebLOAD Resource Manager.

WebLOAD™ Scripting Guide

Complete information on scripting and editing JavaScript scripts for use in WebLOAD and WebLOAD Recorder.

WebLOAD™ JavaScript Reference Guide

Complete reference information on all JavaScript objects, variables, and functions used in WebLOAD and WebLOAD Recorder test scripts.

WebLOAD™ Extensibility SDK

Instructions on how to develop extensions to tailor WebLOAD to specific working environments.

WebLOAD™ Automation Guide

Instructions for automatically running WebLOAD tests and reports from the command line, or by using the WebLOAD plugin for Jenkins

WebLOAD™ Cloud User Guide

Instructions for using RadView's WebLOAD Cloud to view, analyze and compare load sessions in a web browser, with full control and customization of the display.

The guides are distributed with the WebLOAD software in online help format. The guides are also supplied as Adobe Acrobat files. View and print these files using the Adobe Acrobat Reader. Install the Reader from the Adobe website http://www.adobe.com.

Icons and Typographical Conventions

Before you start using this guide, it is important to understand the terms, icons, and typographical conventions used in the documentation.

For more information on specialized terms used in the documentation, see *Glossary* (on page 397).

The following icons appear next to the text to identify special information.

Table 1: Icon Conventions

Icon	Type of Information
✎	Indicates a note.
🧩	Indicates a feature that is available only as part of a WebLOAD Add-on.

The following kinds of formatting in the text identify special information.

Table 2: Typographical Conventions

Formatting convention	Type of Information
Special Bold	Items you must select, such as ribbon options, command buttons, or items in a list.
Emphasis	Use to emphasize the importance of a point or for variable expressions such as parameters.
CAPITALS	Names of keys on the keyboard. for example, SHIFT, CTRL, or ALT.
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another, for example, CTRL+P or ALT+F4.

Where to Get More Information

This section contains information on how to obtain technical support from RadView worldwide, should you encounter any problems.

Online Help

WebLOAD provides a comprehensive on-line help system with step-by-step instructions for common tasks.

You can press the **F1** key on any open dialog box for an explanation of the options or select **Help** > **Contents** to open the on-line help contents and index.

Technical Support Website

The technical support page on our website provides:

- The option of opening a ticket
- Links to WebLOAD documentation

Technical Support

For technical support in your use of this product, contact:

North American Headquarters	International Headquarters
e-mail: support@RadView.com Phone: 1-888-RadView (1-888-723-8439) (Toll Free) 908-526-7756 Fax: 908-864-8099	e-mail: support@RadView.com Phone: +972-3-915-7060 Fax: +972-3-915-7011

Note: We encourage you to use e-mail for faster and better service.

When contacting technical support please include in your message the full name of the product, as well as the version and build number.

Chapter 2

Overview of the WebLOAD Integrated Development Environment

This section provides a brief overview to the WebLOAD Integrated Development Environment.

About WebLOAD Recorder

WebLOAD Recorder is an easy-to-use tool for recording, creating, and authoring protocol test scripts for the WebLOAD environment.

WebLOAD Recorder is a visual environment for creating protocol test scripts (referred to as scripts) that provides the following features:

- Recording scripts
- Editing scripts
- Running and Debugging scripts

WebLOAD Recorder records your action in a Web browser and saves it as a JavaScript script. WebLOAD Recorder provides two editing modes, the Visual Editing mode and the JavaScript Editing mode, that enable you to edit your JavaScript script.

WebLOAD Recorder enables you run and play back the script in a number of ways, such as full playback without any breakpoints, with breakpoints, or step-by-step. After the script is run, WebLOAD Recorder returns response data from the Web server. WebLOAD Recorder provides various views of the response data to help you debug and edit the script. These views include a Web Page view, HTTP Header view, JavaScript view, DOM view, and HTML view.

The script can then be used in the WebLOAD environment to test the performance of your Web application.

The User Flow

As you develop a Web application, you and your organization will usually do the following:

1. Plan your session to include the basic tasks that you want the test to perform.
2. Create the Test script in WebLOAD Recorder.
3. Test the application in WebLOAD using the script created in WebLOAD Recorder.

   You do not need to modify the test script as it can run from WebLOAD Recorder to WebLOAD seamlessly.

WebLOAD emulates multiple users on a network or server, testing to be sure the application *scales* as needed. These tests ensure that your application operates "normally" under load and stress, and your application appears as per your specifications and to your visitors' expectations.

The scripts are executed during WebLOAD test sessions by multiple Virtual Clients in parallel, achieving simultaneous access to the SUT and generating the load burden necessary for effective testing. Each execution of the script generates an independent instance running in parallel during your WebLOAD test session.

Note: Refer to the WebLOAD documentation for more information about using WebLOAD.

Script Creation

You create a JavaScript script in WebLOAD Recorder so that you can test applications by running the JavaScript script in WebLOAD to simulate the actions of real users.

A *Script* is a test script written in JavaScript code that is used to test the functionality of a Web application under a load. It contains a sequence of HTTP protocol calls sent by Virtual Clients to your System Under Test (SUT). For example, if you want to test the performance of your Web application when clients access a certain page, your script must contain the code for accessing the page.

A script can be generated automatically using the recording tools supplied with WebLOAD Recorder, or it can be created manually by writing a script. This guide describes the recording tools supplied with WebLOAD Recorder for developing test scripts automatically and provides instructions for developing test scripts manually.

Before creating a script, you should consider and plan which actions you want to include in a test session.

Create a script by carrying out the following steps:

1. Recording the script.
2. Editing / enhancing the script.
3. Running and debugging the script.

The first step of creating a script is recording. As you execute a typical sequence of activities, WebLOAD Recorder records your accesses, creating a precise, detailed record of all your activities and application responses that occur during a recording session.

WebLOAD Recorder operates in conjunction with a Web browser, such as Microsoft's Internet Explorer. The basic 'Building Blocks' of a test session are your actions. As you work with a test application in the browser, (navigating between pages, typing text into a form, clicking the mouse, and so on), WebLOAD Recorder stores information about you actions in a script file. Externally, your activities are represented in WebLOAD Recorder by a set of icons arranged in a Visual Script Tree. Internally, WebLOAD Recorder records these actions and automatically creates scripts that act as scripts, recreating the actions and verifying the functionality of Web sites under realistic conditions.

The second step of creating a script is editing the code of the recorded script. This can be done in Visual Editing mode and/or JavaScript Editing mode. For example, if you want a script to vary a sequence of accesses, submit randomized data read from a file, or work with Java or COM components, a certain degree of programming is required. This guide describes how to edit the code in your scripts to add more complex functionality to your testing sessions.

The last step is to run your script in WebLOAD Recorder to perform testing so you can emulate how your script will run when executed in WebLOAD. You can then use the debugging tools to correct or modify your script so that it acts as you expected.

After completing these basic steps, you can incorporate your script into a WebLOAD test.

Note: For examples of basic scripts, see the *WebLOAD Scripting Guide*.

The Recording Tool

WebLOAD Recorder is supplied with a recording tool to perform the following:

- Recording on any site, including sites that use SSL security.
- Recording in any browser that supports a configurable proxy.

The recording tool runs independently of the WebLOAD Recorder. It runs under Microsoft Windows 2000, XP, 2003 and 2007.

For detailed instructions on using WebLOAD Recorder to record scripts, see *Recording Scripts* (on page 33).

The Editing Modes

WebLOAD Recorder provides two modes in which to write lines of code:

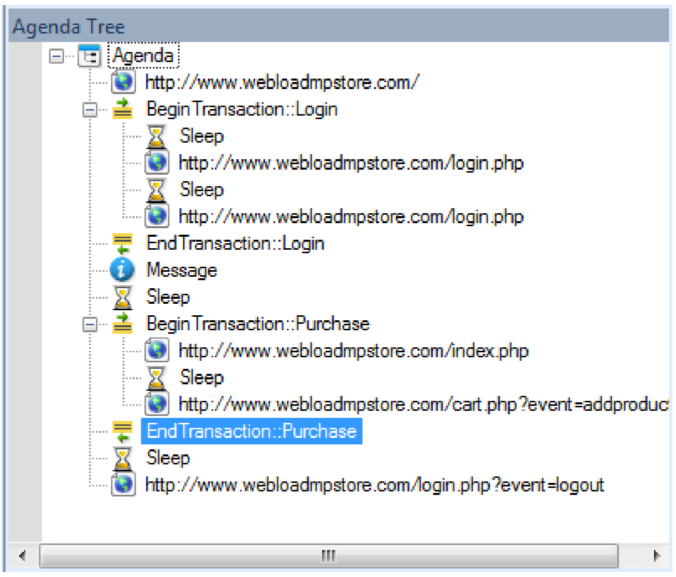
- Visual Editing mode
- JavaScript Editing mode

You can switch between modes while customizing scripts.

Visual Editing Mode

In Visual Editing mode, rather than writing numerous lines of code to describe the actions you want to test, you simply record the actions in a browser without programming. Your interactions with your Web application are captured, recorded, and presented graphically in the Script Tree.

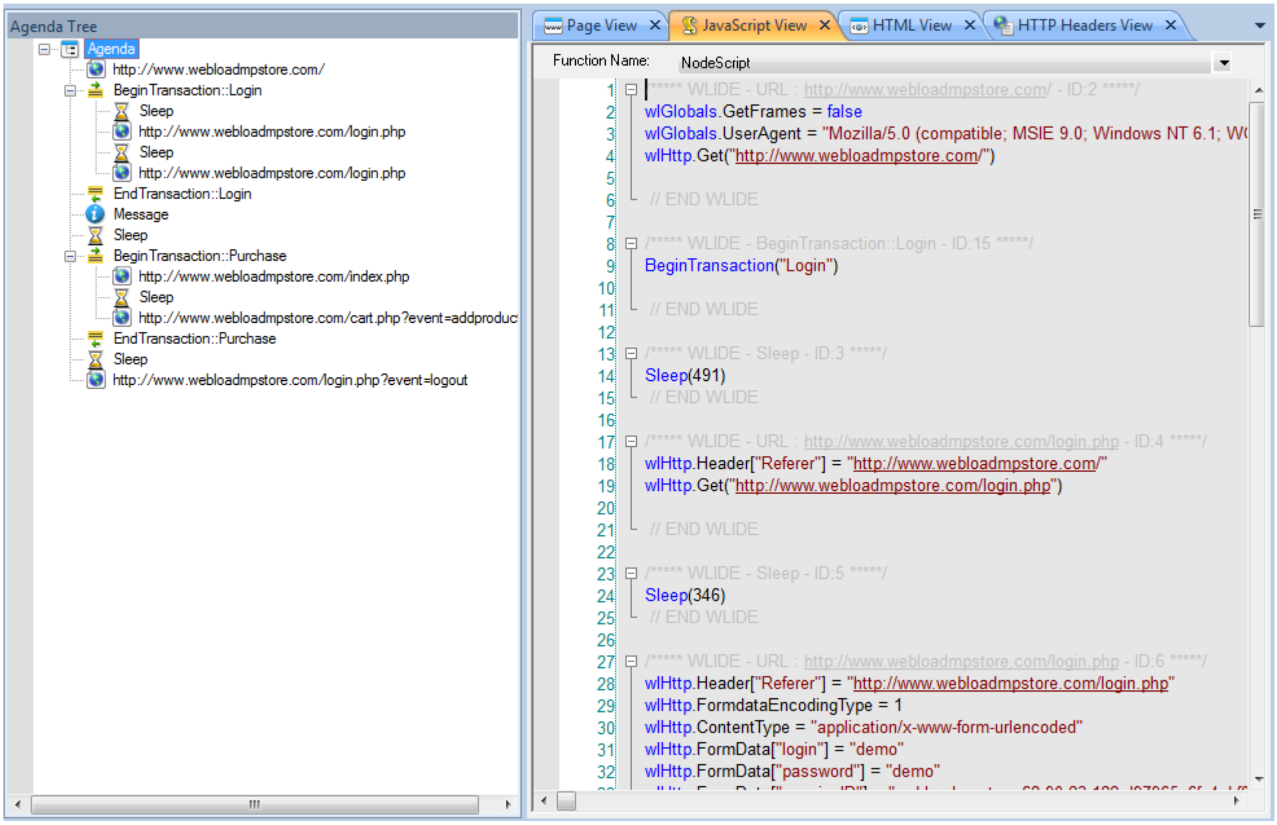
When editing a script, you can also drag and drop items from the WebLOAD Recorder toolboxes into the script Tree. This makes programming easier by building the code behind an intuitive drag-and-drop interface.



```
Agenda Tree
Agenda
    http://www.webloadmpstore.com/
    BeginTransaction::Login
        Sleep
        http://www.webloadmpstore.com/login.php
        Sleep
        http://www.webloadmpstore.com/login.php
    EndTransaction::Login
    Message
    Sleep
    BeginTransaction::Purchase
        http://www.webloadmpstore.com/index.php
        Sleep
        http://www.webloadmpstore.com/cart.php?event=addproduc
    EndTransaction::Purchase
    Sleep
    http://www.webloadmpstore.com/login.php?event=logout
```

Figure 1: Script Tree

Each node in the Script Tree is a graphical representation of the JavaScript code. JavaScript code that cannot be edited appears grayed out.



```
Agenda
    http://www.webloadmpstore.com/
    BeginTransaction::Login
        Sleep
        http://www.webloadmpstore.com/login.php
        Sleep
        http://www.webloadmpstore.com/login.php
    EndTransaction::Login
    Message
    Sleep
    BeginTransaction::Purchase
        http://www.webloadmpstore.com/index.php
        Sleep
        http://www.webloadmpstore.com/cart.php?event=addproduc
    EndTransaction::Purchase
    Sleep
    http://www.webloadmpstore.com/login.php?event=logout
```

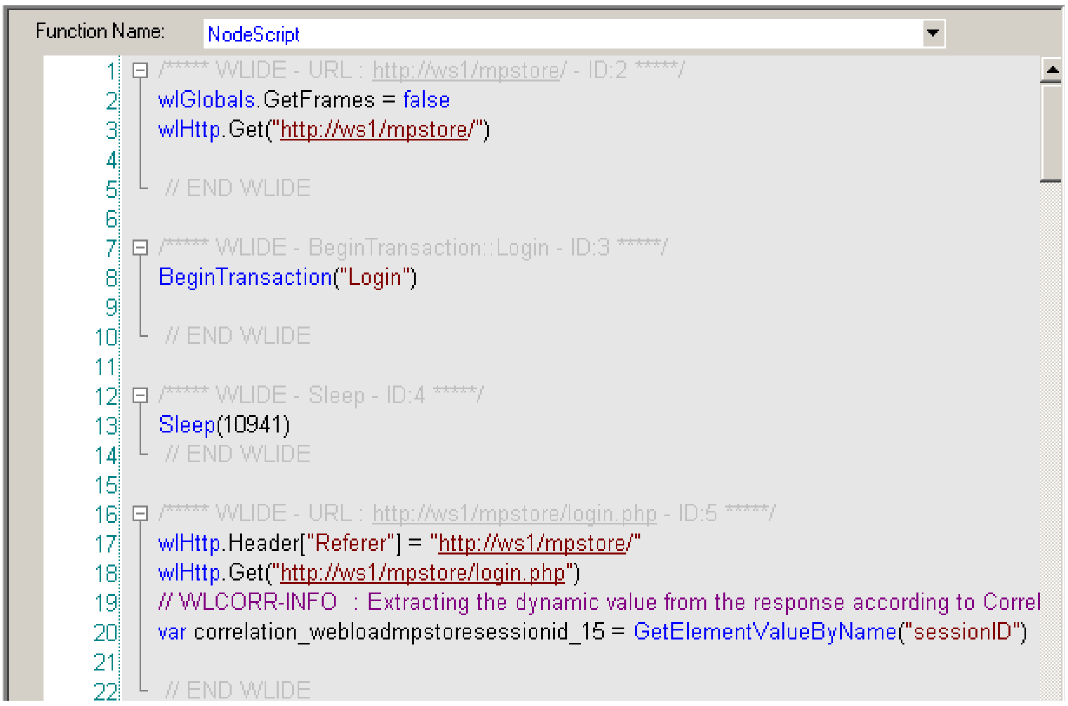
Function Name: NodeScript

```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:2 *****/
wlGlobals.GetFrames = false
wlGlobals.UserAgent = "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; W
wlHttp.Get("http://www.webloadmpstore.com/")


 // END WLIDE

/***** WLIDE - BeginTransaction::Login - ID:15 *****/
BeginTransaction("Login")

 // END WLIDE

/***** WLIDE - Sleep - ID:3 *****/
Sleep(491)
 // END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:4 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/"
wlHttp.Get("http://www.webloadmpstore.com/login.php")

 // END WLIDE

/***** WLIDE - Sleep - ID:5 *****/
Sleep(346)
 // END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:6 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/login.php"
wlHttp.FormdataEncodingType = 1
wlHttp.ContentType = "application/x-www-form-urlencoded"
wlHttp.FormData["login"] = "demo"
wlHttp.FormData["password"] = "demo"
```

Figure 2: Script Tree with JavaScript Code

JavaScript Editing Mode

WebLOAD Recorder provides complete testing flexibility with the JavaScript Editor, enabling you to add your own code into the recorded script or import a JavaScript file. Each block of code is presented graphically in the Script Tree.



Function Name: NodeScript

```
/***** WLIDE - URL : http://ws1/mpstore/ - ID:2 *****/
wlGlobals.GetFrames = false
wlHttp.Get("http://ws1/mpstore/")

 // END WLIDE

/***** WLIDE - BeginTransaction::Login - ID:3 *****/
BeginTransaction("Login")

 // END WLIDE

/***** WLIDE - Sleep - ID:4 *****/
Sleep(10941)
 // END WLIDE

/***** WLIDE - URL : http://ws1/mpstore/login.php - ID:5 *****/
wlHttp.Header["Referer"] = "http://ws1/mpstore/"
wlHttp.Get("http://ws1/mpstore/login.php")
// WLCORR-INFO  : Extracting the dynamic value from the response according to Correl
var correlation_webloadmpstoresessionid_15 = GetElementValueByName("sessionID")

 // END WLIDE
```

Figure 3: JavaScript Editor

WebLOAD Recorder provides the following programming assistance to manually edit a script:

- IntelliSense Editor mode for the JavaScript View pane.
- Insert menu with commonly used functions and commands.
- Syntax Checker that checks the syntax of the code in your script file and catches simple syntax errors before you spend any time running a test session.
- Import JavaScript files.

Note: For detailed information about the JavaScript language, see *The Core JavaScript Language* in the *Netscape JavaScript Guide.* This guide is supplied in Adobe Acrobat format with the WebLOAD software. You may also learn the elements of JavaScript programming from many books on Web publishing. Keep in mind that some specific JavaScript objects relating to Web publishing do not exist in the WebLOAD test environment.

The Run Mode

WebLOAD Recorder enables you to run the script and view the results. You can then debug the script.

WebLOAD Recorder provides a debugger that enables you to correct or modifying your script so that is acts as you expected. It includes a variety of tools to help with the task of debugging your script, such as setting breakpoints and specifying watch variables and expressions.

Chapter 3

Before You Begin using WebLOAD Recorder

This section provides information before you begin using WebLOAD Recorder.

Before You Begin

Before you begin recording scripts using WebLOAD Recorder, there are configuration steps that you may have to complete, depending on the Web application you want to test.

If you plan to record a script that includes retrieving a page that you accessed previously during that script, you should clear the cache and cookies in the browser. See Clearing the Cache and Cookies in Your Browser (on page 13).

When you have completed these startup steps, you can either start working with WebLOAD Recorder immediately, or you can configure the recording options first. For more information about configuring the recording options, see *Configuring the Recording and Script Generation Options* (on page 174).

Clearing the Cache and Cookies in Your Browser

If your browser is set to use a cache file, steps such as loading a page that you have already visited are bypassed when you record a script.

If your browser loads a page from the cache file, that action is not recorded because retrieving a file from the cache is not an HTTP protocol call. Typically, this behavior is appropriate because you want to emulate the behavior of an actual browser during a test. However, if you want each visit to a page during a test to connect through an actual GET statement, you must work without a cache file when you record a script.

When you start recording, WebLOAD automatically changes the browser's proxy settings to clear the cache and cookies, according to the definitions in the Recording and Script Generation dialog box (see *Configuring the Recording and Script Generation Options* on page 174). This enables WebLOAD Recorder to record all HTTP traffic. If you do not want to clear the cache and cookies automatically, you can manually clear the cache and cookies in your browser by following the instructions provided by your browser.

Configuring the Proxy Value for Your Browser

Before you begin recording scripts using WebLOAD Recorder, your browser must be configured to use a specific proxy setting. This is usually done automatically when opening WebLOAD Recorder, but can also be done manually in the browser.

Note: If your system is already configured to use a proxy setting, WebLOAD will preserve this setting for internal use, and will restore the setting after recording is complete.

The procedures described here describe how to configure the proxy server for Internet Explorer, Netscape Navigator, and Mozilla Firefox. If you are using a different browser, read through the proxy setting procedures and modify them as necessary for configuring your browser.

Note: If your system is already using the WebLOAD Recorder default port (9884) for another application, you may designate an alternate port number (see *Setting the Proxy Options* on page 200).

When recording is finished, reset the browser proxy to its original setting.

Configuring the Proxy Value in Internet Explorer

To configure the proxy value in Internet Explorer:

1. Open WebLOAD Recorder (see *Starting WebLOAD Recorder* on page 34).
2. Locate the Proxy Port number in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame. Usually this port number is 9884 (see *Setting the Proxy Options* on page 200).
3. Determine if your organization has a Proxy Server that must be used to access the Internet when you record scripts.
4. If your organization *has a Proxy Server*:
   a. Determine the proxy name and port number.
   b. If the proxy port that it uses *is not* the proxy port number found in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame, go to step 6.

   c. If the proxy port number *is* the proxy port number found in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame, go to step 7.

5. If your organization *does not use a Proxy Server*, go to step 7.
6. Configure your organization's proxy as the application proxy in WebLOAD Recorder:

   a. Open WebLOAD Recorder.

   b. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon and then select the **Proxy Options** tab.

   c. Select the **Use the following definitions for the application's proxy server** option.

   d. In the **HTTP Proxy/Port** fields, type the name and port number of your organization's proxy.

   e. Click **OK.**

7. Open Internet Explorer.
8. Select **Tools** > **Internet Options** and then select the **Connections** tab.
9. Click **LAN Settings**.
10. In the Local Area Network LAN Settings dialog box, select the **Use a proxy server option**.
11. In the **Address** field, type `localhost`.
12. In the **Port** field, type the proxy port number found in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame.
13. Verify that the **Bypass proxy server for local addresses** checkbox is cleared.
14. Click **OK.**

    You are finished configuring your proxy value.

Configuring the Proxy Value in Netscape Navigator

To configure the proxy value in Netscape Navigator:

1. Open WebLOAD Recorder (see *Starting WebLOAD Recorder* on page 34).
2. Locate the Proxy Port number in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame. Usually this port number is 9884 (see *Setting the Proxy Options* on page 200).
3. Determine if your organization has a Proxy Server that must be used to access the Internet when you record scripts.

4. If your organization *has a Proxy Server*:
   a. Determine the proxy name and port number.
   b. If the proxy port that it uses *is not* the proxy port number found in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame, go to step 6.
   c. If the proxy port number *is* the proxy port number found in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame, go to step 7.
5. If your organization *does not use a Proxy Server*, go to step 7.
6. Configure your organization's proxy as the application proxy in the WebLOAD Recorder:
   a. Open WebLOAD Recorder.
   b. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon and then select the Proxy Options tab.
   c. Select the **Use the following definitions for the application's proxy server** option.
   d. In the **HTTP Proxy/Port** fields, type the name and port number of your organization's proxy.
   e. Click **OK.**
7. Open Netscape Navigator and do one of the following:
   - If you are using Navigator 3.x, select **Options** > **Network Preferences**.
   - If you are using Navigator 4.x, select **Edit** > **Preferences**.
8. Within Netscape Navigator, do one of the following:
   - If you are using Navigator 3.x, in the Preferences dialog box, select the Proxies tab.
   - If you are using Navigator 4.x, in the Preferences dialog box, under Category, expand Advanced and then select Proxies.
9. Select the **Manual Proxy Configuration** option.
10. In the Manual Proxy Configuration dialog box, in the **HTTP Address** field, type `localhost.`
11. In the corresponding **Port Number** field, type the proxy port number found in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame.
12. Click **OK** to close the Manual Configuration dialog box.
13. Click **OK** to close the Preferences dialog box.

    You are finished configuring your proxy value.

Configuring the Proxy Value in Mozilla Firefox

To configure the proxy value in Mozilla Firefox:

1. Open WebLOAD Recorder (see *Starting WebLOAD Recorder* on page 34).
2. Locate the Proxy Port number in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame. Usually this port number is 9884 (see *Setting the Proxy Options* on page 200).
3. Determine if your organization has a Proxy Server that must be used to access the Internet when you record scripts.
4. If your organization *has a Proxy Server*:
    a. Determine the proxy name and port number.
    b. If the proxy port that it uses *is not* the proxy port number found in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame, go to step 6.
    c. If the proxy port number *is* the proxy port number found in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame, go to step 7.
5. If your organization *does not use a Proxy Server*, go to step 7.
6. Configure your organization's proxy as the application proxy in the WebLOAD Recorder:
    a. Open WebLOAD Recorder.
    b. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon and then select the **Proxy Options** tab.
    c. Select the **Use the following definitions for the application's proxy server** option.
    d. In the **HTTP Proxy/Port** fields, type the name and port number of your organization's proxy.
    e. Click **OK.**
7. Open Mozilla Firefox.
8. Select **Tools** > **Options**.
9. Click **Advanced** and then click the **Network** tab.
10. In the Connection area, click **Settings**.
11. Click **Manual proxy configuration**.
12. In the **HTTP Proxy** field, type `localhost`.

13. In the **Port** field, type the proxy port number found in the Recording and Script Generation Options dialog box – Proxy Options tab – Recording Proxy Options frame.
14. Select the **Use this proxy for all protocols** checkbox.
15. Click **OK.**

    You are finished configuring your proxy value.

Chapter 4

WebLOAD Recorder Quick Start

Welcome to WebLOAD Recorder, part of the premier load testing tool that helps you quickly and easily test the functionality of your application under load. WebLOAD Recorder serves as the recorder for WebLOAD. Using an intuitive visual interface, WebLOAD Recorder helps you create, edit and debug your own test scripts, and prepare them for automatically testing your Web based applications.

WebLOAD Recorder's visual environment gives you easy-to-use editing tools. Once you understand the components of the product and a few basic techniques, you can use these methods throughout WebLOAD Recorder.

This Quick Start explains how to start the program and use the features of the WebLOAD Recorder interface.

Getting Started

This section shows you how you can get started quickly, using the RadView Software test site at www.webloadmpstore.com.

You will be working with a test script. The basic steps are:

1. Recording your script – describes the steps in recording a basic script (see *Creating a* on page 20).
2. Editing your script – explains how to edit and modify your script, insert new items into your script, and parameterize form data to create data driven tests (see *Editing Your Script* on page 26).
3. Running and debugging your script – explains run and debug your script (see *Running and Debugging Your Script* on page 29).

Note: We recommend that you follow the steps in order. All examples are interrelated and dependent on earlier steps.

Creating a Script

The first step in creating a script is to record your actions as you interact with your Web application.

To create a script:

1. Start WebLOAD Recorder by selecting **Start** > **Programs** > **RadView** > **WebLOAD** > **WebLOAD Recorder**.

   WebLOAD Recorder opens.

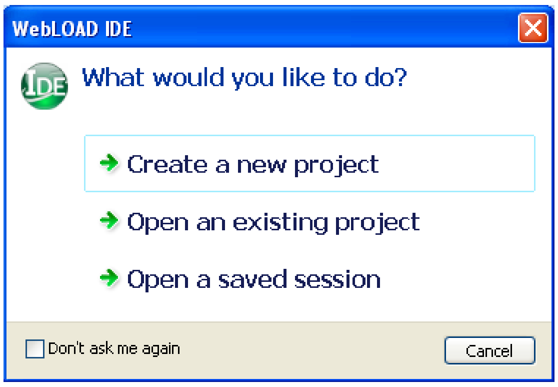


Figure 4: WebLOAD Recorder Startup Dialog Box

2. Select **Create a new project.**

   The WebLOAD Recorder main window opens in Visual Editing Mode, for you to begin creating your script.

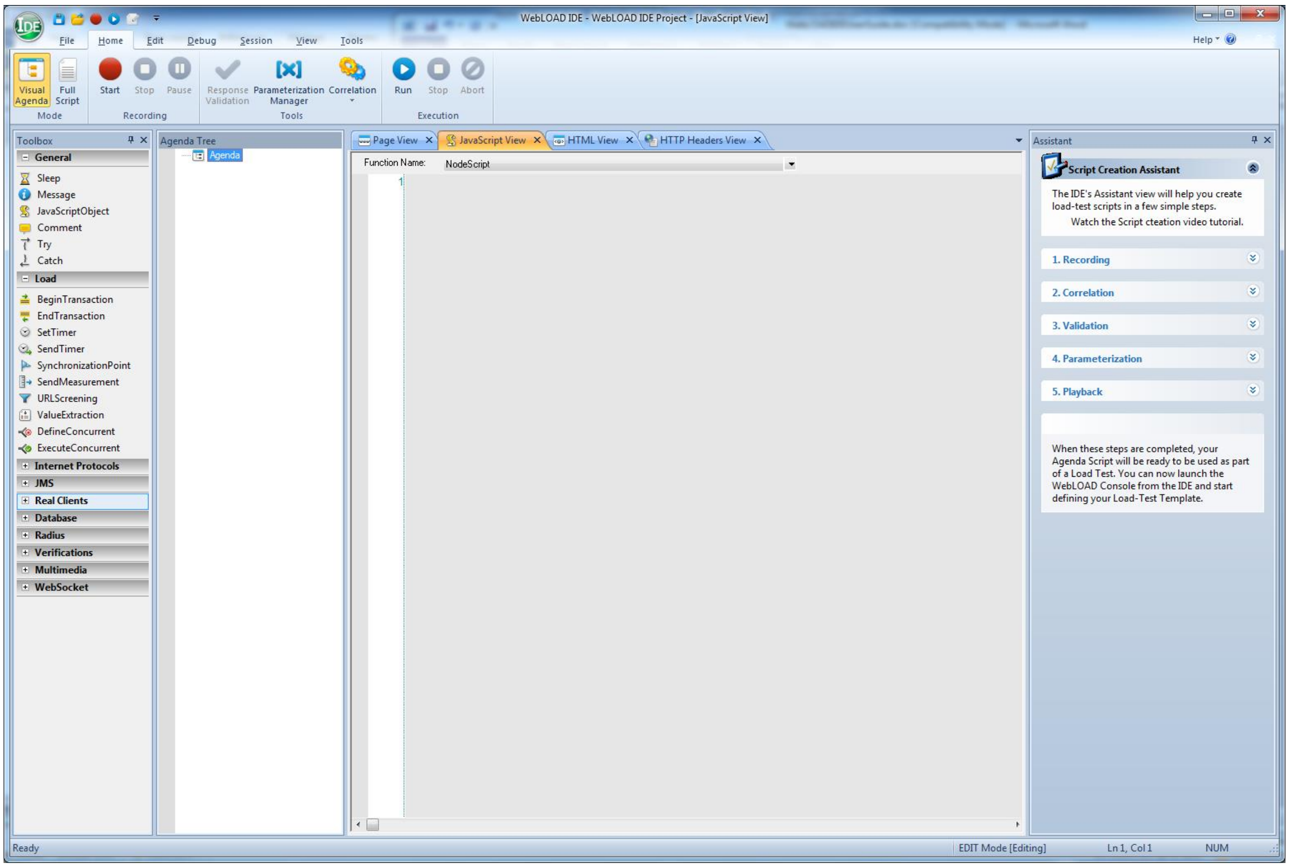
   When the WebLOAD Recorder main window first opens, it opens in Visual Editing Mode. In this mode, there are several active panes. The Script Tree appears on the left, and various view panes appear on the right: JavaScript View, Page View, HTML View, and HTTP headers View.

   In Visual Editing mode, you can simply record the actions in a browser without programming. Your interactions with your Web application are captured, recorded, and presented graphically in the Script Tree.

   Each node in the Script Tree is actually a visual representation of JavaScript code. You can view the contents of the nodes in the JavaScript view pane.

   To the left of the Script Tree are WebLOAD Recorder toolboxes that can be used to edit a script by dragging and dropping items from the WebLOAD Recorder toolboxes into the Script Tree. This makes programming easier by building the code behind an intuitive drag-and-drop interface.

To the right of the View panes is the Assistant pane, which contains simple instructions to help you create your load test script. Click a link in the Assistant to go to the relevant item.



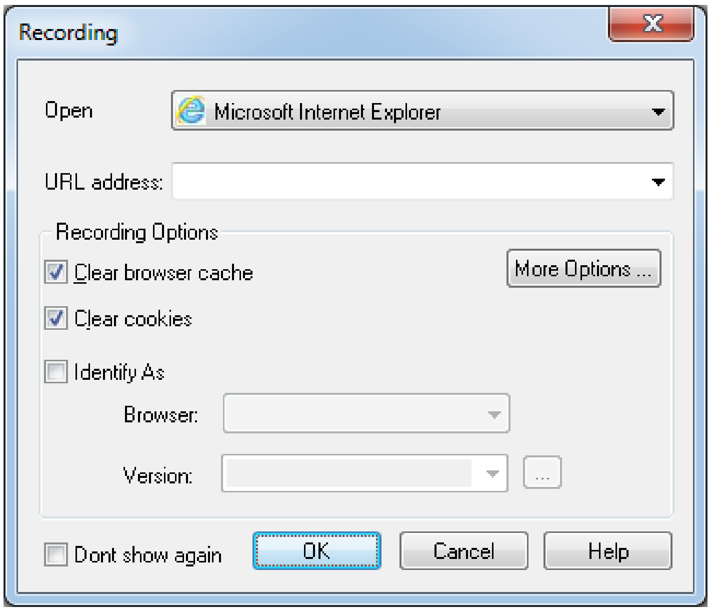
The IDE's Assistant view will help you create load-test scripts in a few simple steps.
Watch the Script ctreation video tutorial.

When these steps are completed, your Agenda Script will be ready to be used as part of a Load Test. You can now launch the WebLOAD Console from the IDE and start defining your Load-Test Template.

Figure 5: WebLOAD Recorder Main Window in Editing Mode

3. In the main window, in Visual Editing Mode, click **Start** in the **Home** tab of the ribbon to begin recording.

The following dialog appears:


Recording

Open: Microsoft Internet Explorer

URL address:

Recording Options

- [✓] Clear browser cache
- [✓] Clear cookies
- [ ] Identify As

Browser:

Version:

[ ] Dont show again

Figure 6: Recording Dialog Box

4. Click **OK**.

   WebLOAD Recorder begins recording all of the actions you perform in the browser, as indicated by the recording notification in the WebLOAD Recorder status bar.

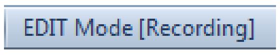

EDIT Mode [Recording]

Figure 7: Status Bar

   A blank browser window opens.

5. In the address bar, enter the Web address www.webloadmpstore.com to go to the WebLOAD test site.

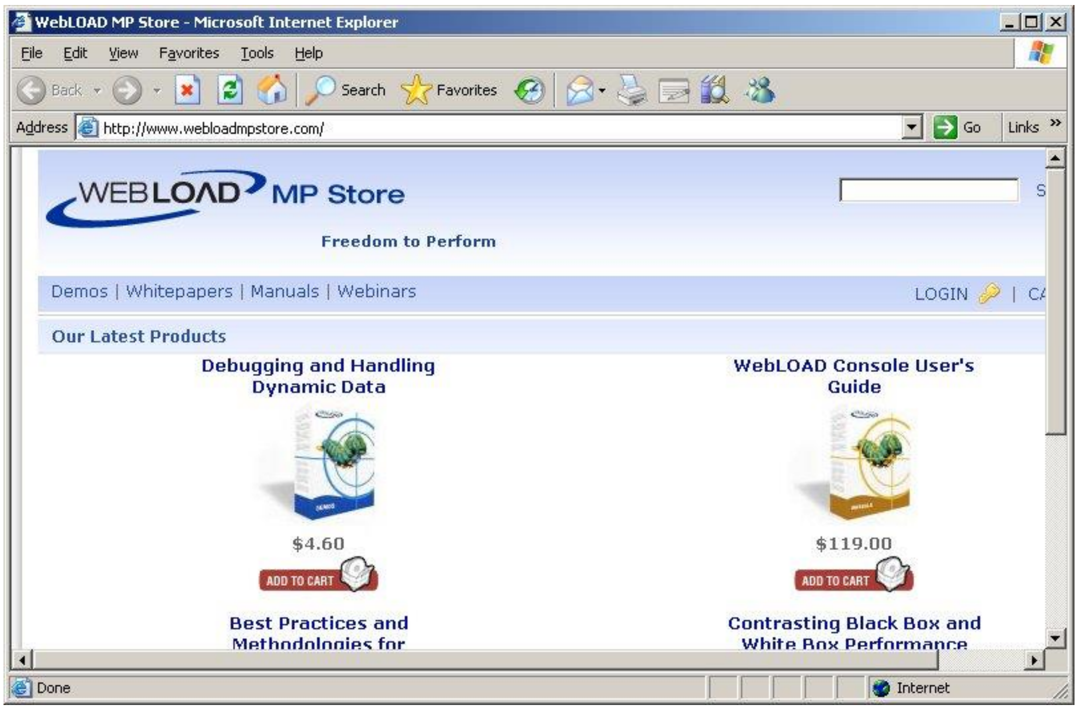

WebLOAD MP Store - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.webloadmpstore.com/
Go Links
WEBLOAD MP Store
Freedom to Perform
Demos | Whitepapers | Manuals | Webinars
LOGIN
Our Latest Products
Debugging and Handling Dynamic Data
$4.60
ADD TO CART
WebLOAD Console User's Guide
$119.00
ADD TO CART
Best Practices and Methodologies for
Contrasting Black Box and White Box Performance
Done
Internet

Figure 8: WebLOAD Test Site

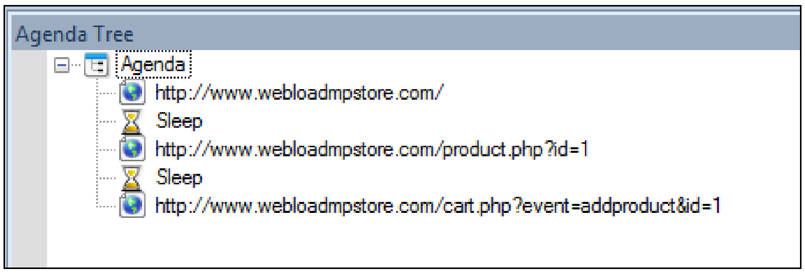
6. Navigate through the site, performing the actions you want to test.

   For example:

   a. Click a product to view the product details.

   b. Click **Add to Cart**.

   Your actions are recorded and appear in the Script Tree as you navigate the site. (If you see more nodes in the Script Tree with different URLs, this may be traffic generated by browser plug-ins or extensions, for example, third-party toolbars.)


Agenda Tree

```
Agenda
  http://www.webloadmpstore.com/
  Sleep
  http://www.webloadmpstore.com/product.php?id=1
  Sleep
  http://www.webloadmpstore.com/cart.php?event=addproduct&id=1
```

Figure 9: Script Tree

7. Click **Stop Recording** in the **Home** tab of the WebLOAD Recorder ribbon to stop the recording.

8. Click **Save** in the **Home** tab of the WebLOAD Recorder ribbon to save your script.

9. Type in `QuickStart` for the name of the script in the Save As dialog box and click **Save**.

The script is saved with the extension `*.wlp`.

You now have a basic script that can be used in a WebLOAD template. For complete information on creating, editing, modifying scripts, and adding functionality to your script, see the rest of the *WebLOAD Recorder User's Guide*.

Viewing Your Script

You can view the recorded script in four views:

- JavaScript View

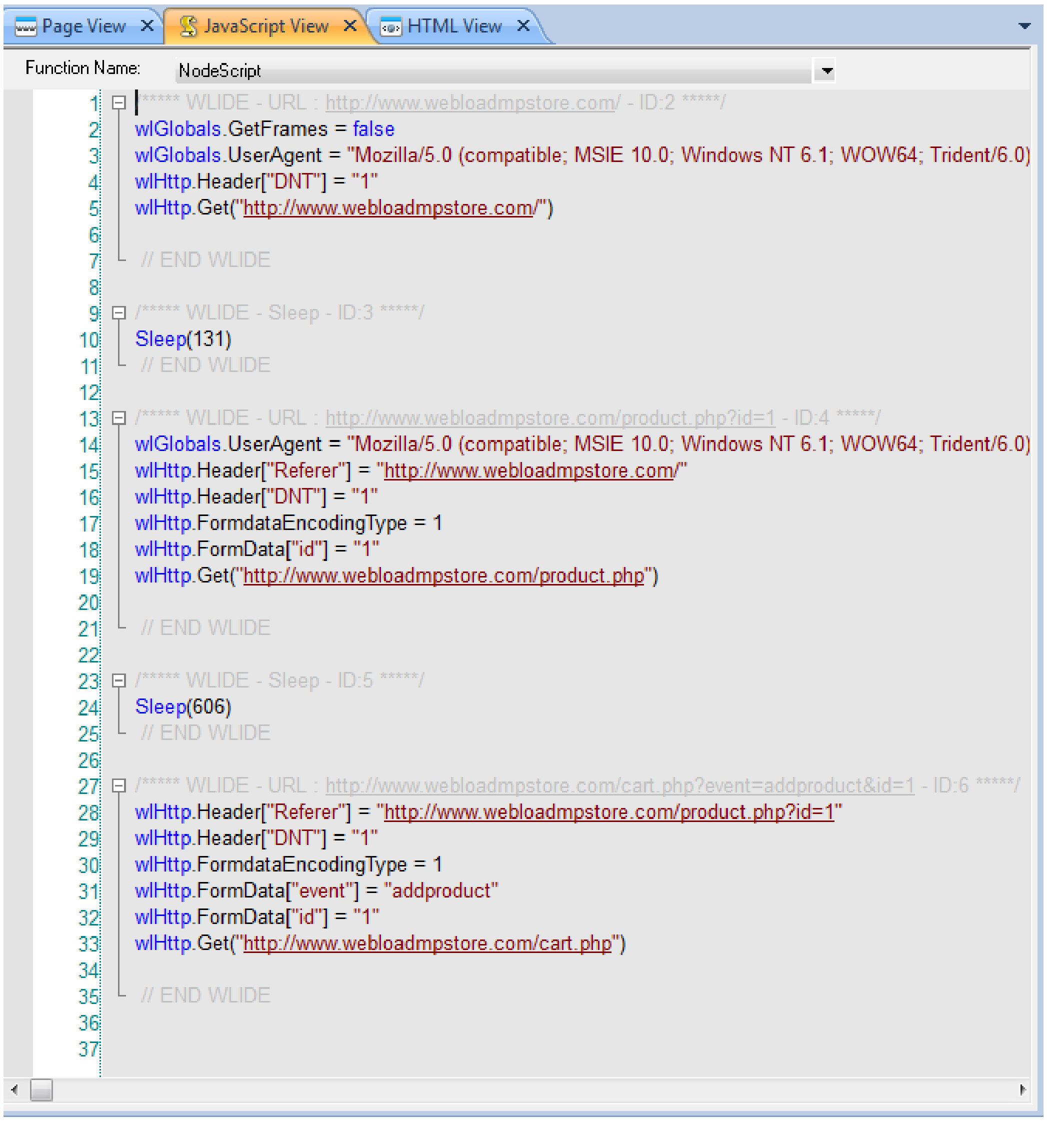
  When the WebLOAD Recorder main window first opens, it opens in Visual Editing Mode. In this mode, there are several active panes. The Script Tree appears on the left, and various view panes appear on the right: JavaScript View, Page View, HTML View, and HTTP headers View.

  When recording, your interactions with your Web application are captured, recorded, and presented graphically in the Script Tree.

  Each node in the Script Tree is actually a visual representation of JavaScript code. You can view the contents of the nodes in the various View panes.

  In the JavaScript view pane, you can do the following:

  - Display the code for each node individually.
  - View code for the entire script as a whole.
  - View the code for different sections in the script, by clicking the Agenda root node in the Script Tree and selecting a section from the Function Name list at the top of the JavaScript view pane.



```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:2 *****/
wlGlobals.GetFrames = false
wlGlobals.UserAgent = "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
wlHttp.Header["DNT"] = "1"
wlHttp.Get("http://www.webloadmpstore.com/")

 // END WLIDE

/***** WLIDE - Sleep - ID:3 *****/
Sleep(131)
 // END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/product.php?id=1 - ID:4 *****/
wlGlobals.UserAgent = "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/"
wlHttp.Header["DNT"] = "1"
wlHttp.FormdataEncodingType = 1
wlHttp.FormData["id"] = "1"
wlHttp.Get("http://www.webloadmpstore.com/product.php")

 // END WLIDE

/***** WLIDE - Sleep - ID:5 *****/
Sleep(606)
 // END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/cart.php?event=addproduct&id=1 - ID:6 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/product.php?id=1"
wlHttp.Header["DNT"] = "1"
wlHttp.FormdataEncodingType = 1
wlHttp.FormData["event"] = "addproduct"
wlHttp.FormData["id"] = "1"
wlHttp.Get("http://www.webloadmpstore.com/cart.php")

 // END WLIDE
```

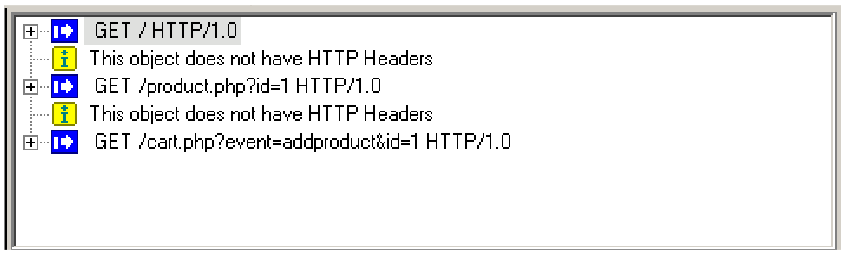
Figure 10: Function Name List in JavaScript View Pane

- HTTP Headers View

  Each node in the Script Tree also has a visual representation of response headers. These response headers were received when the script was recorded. You can view the headers of the nodes in the HTTP Headers view pane. Since each node has a correlated response header, but not all nodes contain HTTP methods, some headers will not have a response header. These nodes will have the message “This object does not have HTTP Headers” associated with them.

  In the HTTP Headers view pane, you can do the following:

  - Display the header for each node individually.
  - View headers for the entire script as a whole.



```
GET / HTTP/1.0
This object does not have HTTP Headers
GET /product.php?id=1 HTTP/1.0
This object does not have HTTP Headers
GET /cart.php?event=addproduct&id=1 HTTP/1.0
```

Figure 11: HTTP Headers View Pane

- HTML View – See *Viewing the Recorded Script in the HTML View Pane* (on page 55).
- Page View – See *Using the Page View to View Results* (on page 132).

Editing Your Script

To edit your script:

1. Edit the runtime settings using the Default and Current Project Options.
2. Toggle between Visual Editing mode and JavaScript Editing mode. The default setting is the Visual Editing mode.
3. In Visual Editing mode, you can edit the Script Tree:
   a. Drag and drop items from the WebLOAD Recorder toolbox into the Script Tree.
   b. Right-click to insert new items.
4. In JavaScript Editing mode:
   a. Modify the JavaScript code.

Important: Each block of code starts with a comment that contains "WLIDE", description, and ID number. The ID number is automatically generated by WebLOAD Recorder and is the connection between the script node and the specific header. It is recommended that you do not change the contents of this comment. If you do, important data might be lost.

   b. Right-click to insert functions and commands.
   c. Use the Syntax Checker to check the syntax of the code in your script file.
   d. Import JavaScript files.

Note: For complete reference information on all JavaScript objects, variables, and functions used in WebLOAD Recorder scripts, see the *WebLOAD JavaScript Reference Guide*.

Toggling Between Edit Modes

You can toggle between Visual mode and Full Script mode. The default setting is the Visual Editing mode.

To toggle between Edit Modes:

- Click **Visual Mode** in the **Home** tab of the ribbon,

  -Or-

  Click the **Full Script** in the **Home** tab of the **ribbon**.

Basic Editing Techniques

WebLOAD Recorder is designed for you to be able to create and edit your script easily, using the visual interface. Once you understand the basic techniques, you can use them throughout the WebLOAD Recorder interface.

Here are some simple techniques, described in this section, that you can use in WebLOAD Recorder:

- Drag and drop items into your Script Tree.
- Right-click within the script and select an available option from the Insert menu.

Drag and Drop

WebLOAD Recorder enables you to drag script items from the WebLOAD Recorder toolbox and drop them into your Script Tree.

To drag and drop items into your script:

1. Place the mouse pointer over the item in the WebLOAD Recorder toolbox that you want to add to your script, such as a Message.
2. Press and hold the left mouse button.
3. Drag the item into the Script Tree, and place the mouse pointer at the step in the script *after* which you want to add the item.
4. Release the mouse button.

   A dialog box to enter the parameters opens or the item appears in the Script Tree.

5. Click the script item in the Script Tree to view and/or edit the JavaScript code in the JavaScript view pane.

Right-Click Menus

Throughout WebLOAD Recorder, context-sensitive menus appear when you click the right mouse button, giving you the appropriate options to select at that point.

You can also right-click any script item in the Script Tree to display a menu.

To insert a new item:

1. Right-click the script item and click **Insert** from the menu.
2. Select an item from the options available.

Adding Script Items

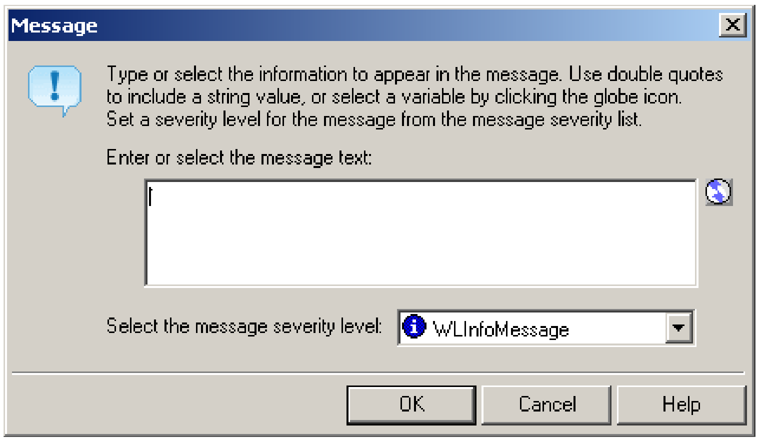
You can drag and drop an item, such as Message, from the WebLOAD Recorder toolbox. For the list of toolboxes, see *The WebLOAD Recorder Toolbox Items* (on page 231).

In the following instructions, adding a Message is used as an example. While running a test session, WebLOAD Recorder and WebLOAD Recorder's Log windows display information about session execution. You can include Message nodes in your script, defining points at which to send error and/or notification messages to the Log window.

To add a Message script item:

1. Place the mouse pointer over the Message icon in the WebLOAD Recorder toolbox.
2. Press and hold the left mouse button.
3. Drag the Message item into the script, and place the mouse pointer after the Web page to which you want to add the message.
4. Release the mouse button.

   The Message dialog box opens.


Message

Type or select the information to appear in the message. Use double quotes to include a string value, or select a variable by clicking the globe icon. Set a severity level for the message from the message severity list.

Enter or select the message text:

Select the message severity level: WLInfoMessage

Figure 12: Message Dialog Box

5. Enter the text you want to appear in the message.
6. To add a global variable to the message text, click the **globe** icon ( ) to the right of the input text box and select a global variable from the drop-down list.

Note: When entering a string value to the message, the string must be enclosed in quotation marks, for example, "Sample Message".

7. Select a severity level for the message from the drop-down list. The following severity levels are available:
   - Information message (`WLInfoMessage`)
   - Minor error message (`WLMinorError`)
   - Error message (`WLError`)
   - Severe error message (`WLSevereError`)
8. Click **OK**.

   The Message item appears in the Script Tree.

Running and Debugging Your Script

After your script has been developed, you run it to test for errors in your application. You can then debug your script.

Running Your Script

To run your script:

1. Click **Run** in the **Debug** tab of the ribbon.

As the script is running:

- A yellow arrow points to the node being executed in the Script Tree.
- If the JavaScript View tab is open, you will also see the yellow arrow pointing to the script.
- If the Page View tab is open, you will see the pages that return from the Web server.
- Nodes are added to the Execution Tree as they occur.
- The GET and POST HTTP protocol commands are displayed in the HTTP Headers view pane.
- Messages and errors generated by the test appear in the Log Window at the bottom of the screen.

2. At the prompt asking whether to save the changes to the project, click **Yes** and enter a file name to save your script file.

Note: If there is more than one tester and the tests are to be shared between testers, the root directory (test plans and the results of the test plans) and the tests must be saved to a network drive.

Debugging Your Script

WebLOAD Recorder provides an integrated debugger with a variety of tools to help locate bugs in your script. The debugger provides special menus, windows, dialog boxes, and grids of fields for debugging. You can pause the debugger and trigger WebLOAD Recorder to wait for user input before proceeding with running the script. In the script, you can set breakpoints and step into / over / out. You can also abort the debugger without executing the `TerminateClient` and `TerminateAgenda` functions, as opposed to stopping it completely.

To debug your script:

- Click **Step Into** or click **Run** in the **Debug** tab of the WebLOAD Recorder ribbon,

  -Or-

  Add breakpoints by clicking **Toggle Breakpoint** in the **Debug** tab of the WebLOAD Recorder ribbon, and then clicking **Run** to run the script.

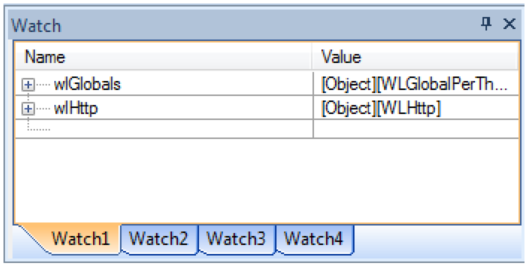
Note: If you stop the debugger prematurely (for example, by closing the WebLOAD Recorder or returning to edit mode), you can instruct WebLOAD, in the Settings dialog box, to prompt you to save the debugging session file. For more information about the Settings dialog box, see *Configuring the Settings* (on page 208).

Debugging Using the Watch Window

You can use the Watch window to specify variables and expressions that you want to watch while debugging your program.

To debug using the Watch window:

1. Start debugging.
2. Select the **Watch Window** checkbox in the **Debug** tab



Watch

Name	Value
wlGlobals	[Object][WLGlobalPerTh...
wlHttp	[Object][WLHttp]

Figure 13: Watch Window

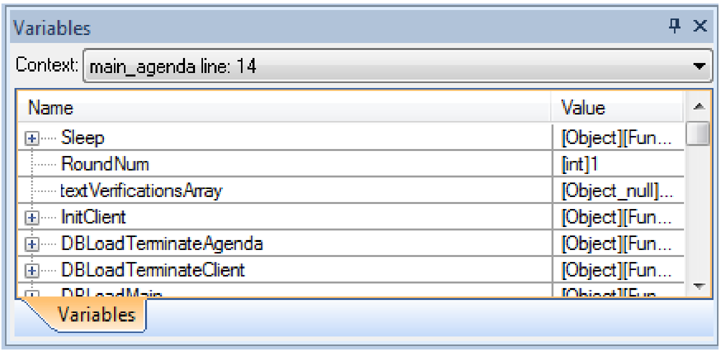
In the Name column, plus sign (+) or minus sign (-) boxes may appear. These appear if you added an array or object variable to the Watch window. Use these boxes to expand or collapse your view of the variable.

Debugging Using the Variables Window

The Variables window provides quick access to variables that are important in the scripts current context.

To debug using the Variables Window:

1. Start debugging.
2. Check the **Variables Window** checkbox in the **Debug** tab of the ribbon.



Variables

Context: main_agenda line: 14

Name	Value
Sleep	[Object][Fun...
RoundNum	[int]1
textVerificationsArray	[Object_null]...
InitClient	[Object][Fun...
DBLoadTerminateAgenda	[Object][Fun...
DBLoadTerminateClient	[Object][Fun...
DBLoadMain	[Object][Fun

Figure 14: Variables Window

The Variables window displays variables used in the current statement and in the previous statement. It also displays return values when you step over or out of a function.

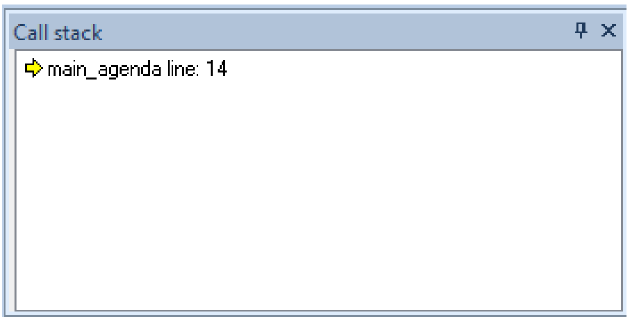
The Variables window contains a grid with fields for the variable name and value. The debugger automatically fills in these fields. You cannot add variables or expressions to the Variables window. The Context dropdown list displays the current scope of the variables displayed.

Debugging Using the Call Stack Window

The Call Stack window lists the function calls that led to the current statement, with the current function on the top of the stack.

To debug using the Call Stack Window:

1. Start debugging.
2. Select the **Call Stack** checkbox in the **Debug** tab of the ribbon.



main_agenda line: 14

Figure 15: Call Stack Window

This Quick Start has shown you an example of how to record, create, edit, run, and debug a script in WebLOAD Recorder. For more information about all the options available in WebLOAD Recorder, see the rest of the *WebLOAD Recorder User's Guide* and the *WebLOAD Recorder Online Help*.

Chapter 5

Recording Scripts

This section provides instructions for recording scripts with WebLOAD Recorder.

About Recording Scripts with WebLOAD Recorder

Use WebLOAD Recorder to create test scripts as a baseline for testing your Web application in the WebLOAD Console. As you navigate through a Web application, WebLOAD Recorder records your actions, automatically generating a script that reflects your actions in JavaScript. WebLOAD Recorder creates your scripts for you, writing GET and POST HTTP protocol commands automatically.

As your actions are recorded, WebLOAD Recorder displays them in the Script Tree, which is a tree hierarchy with visual indications of the information recorded. WebLOAD Recorder records only HTTP protocol calls that place a load on the System Under Test (SUT). Activities that are not relevant to the script, such as moving windows for a more comfortable display or opening another application, are not recorded. While your script is being recorded, you can edit it with the WebLOAD Recorder Toolbox set. For information on editing your script using the WebLOAD Recorder Toolbox set, see *Editing your Script Using the WebLOAD Recorder Toolbox Set* (on page 85).

This process creates the basic script. You can then view the recorded script as JavaScript code in the JavaScript view pane, revise the script to test more objects in more detail, and run and debug the script. For information on editing your script, see *Editing Scripts* (on page 69). For information on running and debugging your script, see *Running and Debugging Scripts* (on page 111).

The script can then be used with WebLOAD for load and scalability testing of your application.

Starting WebLOAD Recorder

To start WebLOAD Recorder:

1. Select **Start** > **Programs** > **RadView** > **WebLOAD** > **WebLOAD Recorder**.

   WebLOAD Recorder opens.

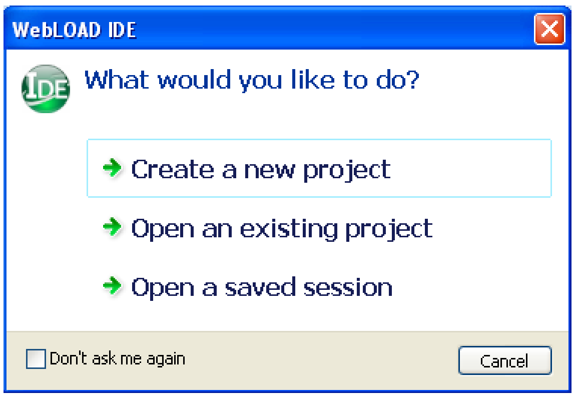


Figure 16: WebLOAD Recorder Startup Dialog Box

2. Check or uncheck **Don't ask me again**.
3. Click one of the following options:
   - **Create a new project** – Opens a new project. WebLOAD supports several types of projects, the default project being Internet Protocol Project. The new project type is set according to the type of project which was last open. To create a different type of project, select **New Project** in the **File** tab of the WebLOAD Recorder ribbon and select the desired project type.
   - **Open an existing project** – Browse to the project.
   - **Open a saved session** – Browse to the session.

   The WebLOAD Recorder main window opens in Editing Mode (*Figure 5*), enabling you to begin creating or editing your script.

Recording a script

You can either start working with WebLOAD Recorder immediately, or you can configure the recording options first. For more information about configuring the recording options, see *Configuring the Recording and Script Generation Options* (on page 174).

When you record a script, WebLOAD Recorder displays the script being created in real time. You can watch WebLOAD Recorder record your actions as you navigate in the Web browser.

If you start and stop recording more than once during a single recording session (for example, to skip an irrelevant step in the application you plan to test) each subsequent set of JavaScript commands is appended to the end of the script. If you open an existing script and start recording new Web activity, WebLOAD Recorder also appends the new JavaScript commands to the end of the script.

To record a script:

1. Start **WebLOAD Recorder** (see *Starting WebLOAD Recorder* on page 34),

   -Or-

   Start **WebLOAD Recorder** from your Explorer by double-clicking the WebLOAD Recorder project file (`.wlp`) or session WebLOAD Recorder session file (`.wls`).

   The WebLOAD Recorder main window opens in Editing Mode, enabling you to begin recording your script.

2. To create a new script, click **New Project** in the **File** tab of the ribbon.
3. To open an existing script:
   a. Click **Open** in the **File** tab of the ribbon.
   b. Select a file.
4. Click **Start** in the **Home** tab of the ribbon.

By default, the Recording dialog box appears.

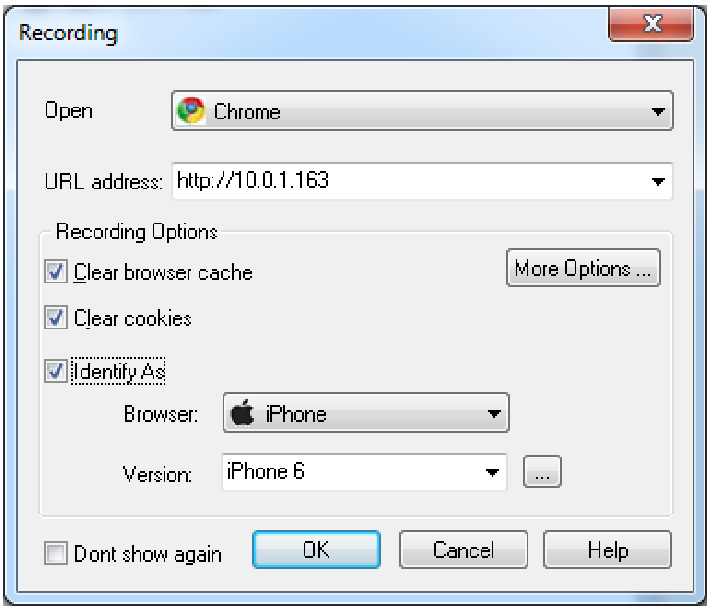


Figure 17: Recording Dialog Box

The Recording dialog box enables you to quickly define the basic settings for the default Web browser which you will be using during the recording.

Note: Any changes to the settings in the Recording dialog box affect the settings of the Browser Settings tab of the Recording and Script Generation Options dialog box (Figure 114). For more information, see *Configuring the Default Browser* (on page 186).

5. Optionally change the browser settings:

Table 3: Recording Dialog Box Options

Field	Description
Open	Select one of the following options as your default browser: • Microsoft Internet Explorer. • Mozilla Firefox. • Google Chrome. • Native Mobile Recording. For further explanations, refer to *Recording Mobile Applications* (on page 375). • None – No default browser. If you selected Mozilla Firefox as your browser, and Mozilla Firefox was installed on the machine *after* WebLOAD Recorder was installed, a message appears recommending that you install the Firefox extension responsible for setting the proxy definitions automatically. If you accept, the extension is installed.

Field	Description
URL address	Enter an address of a web page if you wish to open that specific page in the browser you selected. This option is useful if you wish to open the same page multiple times.
Clear browser cache	Select this option to clear the browser cache before recording. This option is selected, by default.
Clear cookies	Select this option to clear the browser's cookie history before recording. This option is selected, by default.
Identify as	Select this option to simulate a mobile web application.
Browser	Select the browser type you wish to simulate.
Version	Select the browser version you wish to simulate. Alternatively, click the Change button ... to edit the browser version definition. See *Editing Browser Version Definitions* (on page 164).
Don't show again	Select this checkbox if you do not wish to see this dialog box every time you select **Start Recording**.

In addition, you can optionally click **More Options** to open the Browser Settings tab of the Recording and Script Generation Options dialog box (Figure 119) and define the default browser settings in full detail.

6. Click **OK**. The Recording dialog box closes.

   A floating WebLOAD Recording toolbar appears. Throughout any recording session, the WebLOAD Recording toolbar always appears on top of the active window.

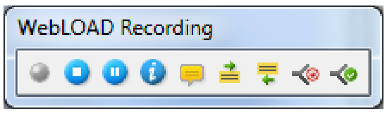


Figure 18: WebLOAD Recording Toolbar

The following table describes the function of each button in the WebLOAD Recording toolbar:

Table 4: WebLOAD Recording Toolbar Buttons

Button	Purpose
●	Start recording.
■	End recording.
⏸	Pause or resume recording.
ℹ	Insert message.
💬	Insert comment.

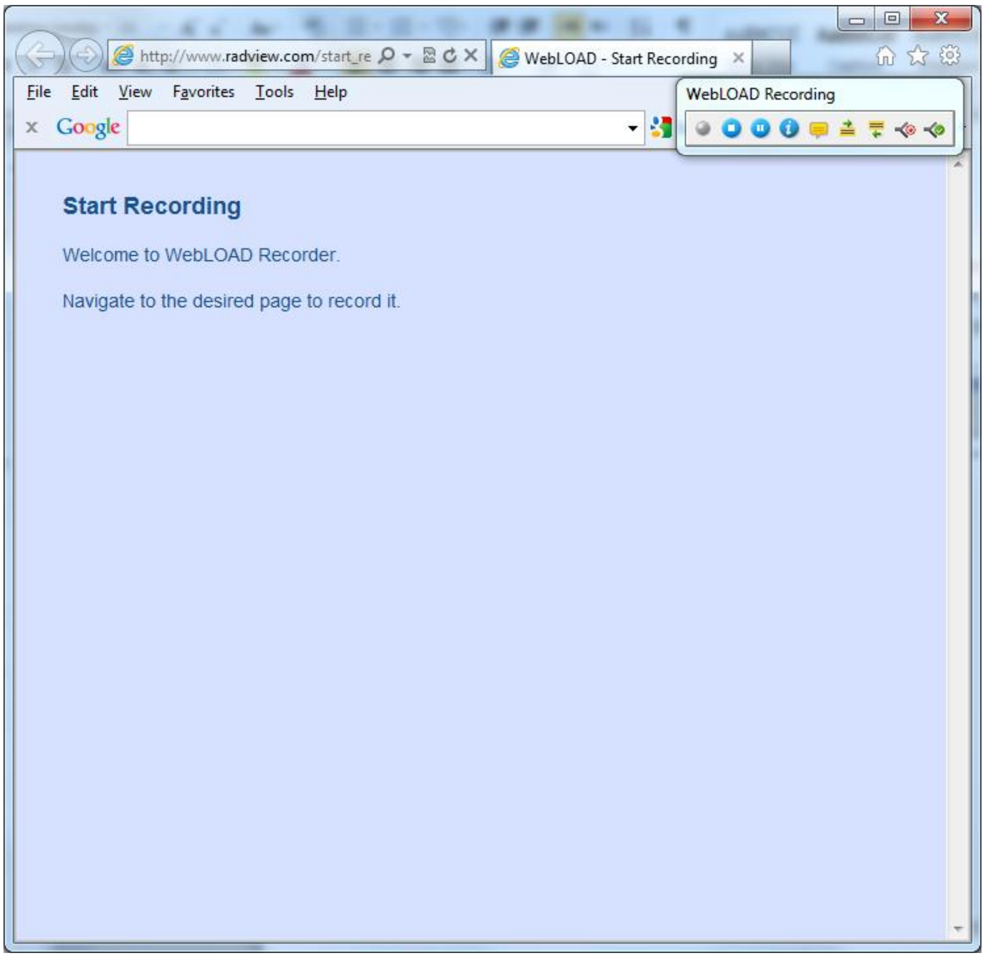
Button	Purpose
	Begin transaction. Adds named transactions to the script to measure the performance of logical actions in your script, such as a Login process. By inserting named transactions into your script, you can take a series of simple actions, define them as a single transaction, and set success or failure criteria for the complete transaction.
	End transaction.
	Define concurrent. Defines a starting point after which the WebLOAD engine collects all Post and Get HTTP requests, but does not execute them until an `Execute Concurrent` function is run.
	Execute concurrent. Defines a starting point after which the WebLOAD engine stops collecting and begins executing all the Post and Get HTTP requests that were defined since the last `Define Concurrent` function, concurrently (using multithreading).

WebLOAD Recorder begins recording all actions you perform in the browser, as indicated by the recording notification in the WebLOAD Recorder status bar.

EDIT Mode [Recording]

Figure 19: Status Bar

If this is the first time that you are recording after WebLOAD Recorder was launched, the default browser opens automatically with its predefined home page. This enables you to start recording and then access a page.


http://www.radview.com/start_re
WebLOAD - Start Recording
File Edit View Favorites Tools Help
Google
WebLOAD Recording

Start Recording

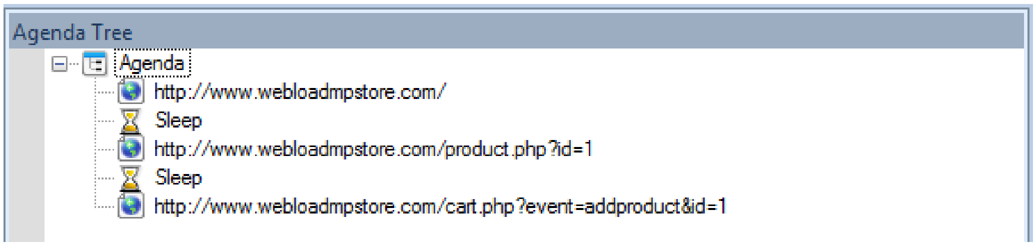
Welcome to WebLOAD Recorder.

Navigate to the desired page to record it.

Figure 20: Default Web Browser

7. In the Web browser window, access the System Under Test (SUT).
8. Perform the steps that you want to test, retrieving and submitting information found on different site pages and locations. Try to emphasize the actions whose performance you need to measure in your test sessions.

   Watch how WebLOAD Recorder adds nodes to the script as you work. Your actions are recorded and appear in the Script Tree as you navigate the site. (If you see more nodes in the Script Tree with different URLs, this may be traffic generated by browser plug-ins or extensions, for example, third-party toolbars.)



Agenda Tree
- Agenda
  - http://www.webloadmpstore.com/
  - Sleep
  - http://www.webloadmpstore.com/product.php?id=1
  - Sleep
  - http://www.webloadmpstore.com/cart.php?event=addproduct&id=1

Figure 21: Script Tree Node

a. Click the **JavaScript View** tab to watch the JavaScript of the pages as they are being recorded.

Note: During recording, the InitAgenda and TerminateAgenda sections of the script are not generated and therefore are not visible.

b. Click the **HTTP Headers View** tab to watch the response headers of the pages as they are being recorded.

c. Click the **HTML View** tab to watch the HTTP data as it is being recorded.

Note: When switching between the JavaScript, HTTP Headers, Browser, and HTML Views, the new view displays the node that is selected in the Script Tree (during edit mode) or Execution Tree (during debug mode). These views are available during recording, after the recording is finished, and after opening a saved script.

9. When you are finished, select **WebLOAD Recorder**.
10. Click **Stop Recording** in the WebLOAD Recorder recording floating toolbar or in the **Home** tab of the WebLOAD Recorder ribbon.

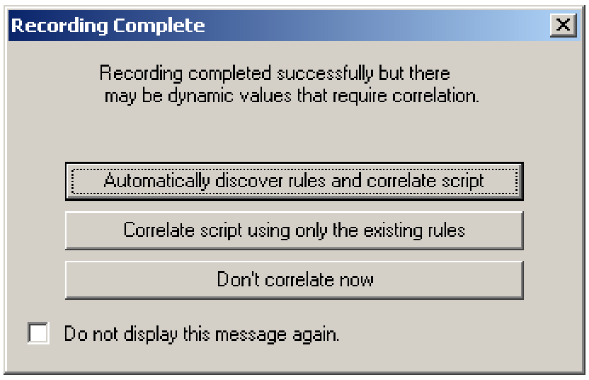
    WebLOAD Recorder stops recording.

11. Click **Save** in the **File** tab of the WebLOAD Recorder ribbon.
12. In the **File name** field in the Save As dialog box, type a descriptive name for the script, and then click **Save**.

    Your script is saved with the file extension `*.wlp`.

13. Close the Browser window to work in WebLOAD Recorder.

    The Recording Complete dialog box opens.


Recording Complete

Recording completed successfully but there may be dynamic values that require correlation.

Automatically discover rules and correlate script
Correlate script using only the existing rules
Don't correlate now
Do not display this message again.

Figure 22: Recording Complete Dialog Box

14. Select one of the following:

    - **Automatically discover rules and correlate script** to run the correlation engine using the existing rules, and apply auto-discovery correlation to find potential correlation rules. For more information, see *Automatic Discovery of Correlation Rules* (on page 94.)

- **Correlate script using only the existing rules** to run the correlation engine using the existing defined rules. For more information, see *Configuring the Correlation Rules* (on page 98).
- **Don't correlate now** to view the recorded script without correlating the script. You can manually correlate the script later.

Notes: Although by default the Recording Complete dialog box appears, this depends on your settings in the Correlation Options tab of the Recording and Script Generations options dialog box. For more information, see *Setting the Default Correlation Action* (on page 93).

You can customize the script in a variety of ways or you can run your script as recorded. For information on editing your script, see *Editing Scripts* (on page 69). For information on running your script, see *Running and Debugging Scripts* (on page 111).

Notes: If actions that you are interested in were not recorded, check the cache settings in your browser. WebLOAD Recorder may be skipping steps that you want to record because your browser is using a system cache file. For more information, see Clearing the Cache and Cookies in Your Browser (on page 13).

When you stop the recording, if no actions were recorded (that is, the script is blank), WebLOAD Recorder automatically displays the recording troubleshooting information.

Pausing a Recording

WebLOAD Recorder enables you to pause a recording so that you can edit the script.

To pause a recording:

1. Click ⏸ from the WebLOAD Recording toolbar,

   -Or-

   Click **Pause** in the **Home** tab of the ribbon.

   The recording pauses.

2. To restart the recording, click ⏸ from the WebLOAD Recording toolbar,

   -Or-

   Click **Start** in the **Home** tab of the ribbon.

   The recording restarts.

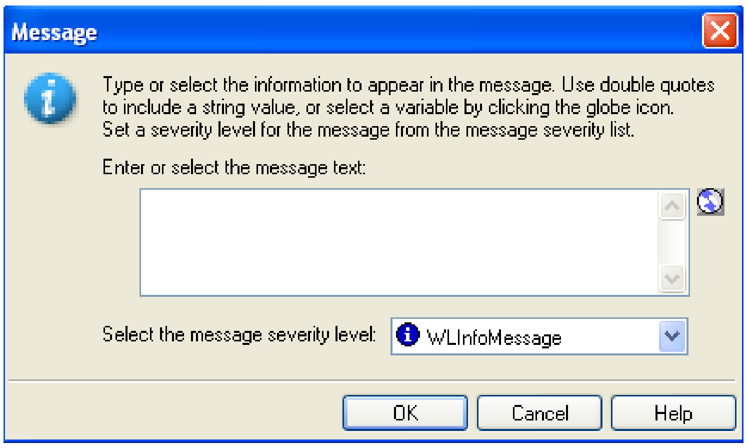
Inserting Messages in a Recording

WebLOAD Recorder enables you to insert messages while recording, defining points at which to send error and/or notification messages to the Log window.

To insert a message:

1. Click ℹ from the WebLOAD Recording toolbar at the desired location in the recording.

   The Message dialog box opens.



Message

Type or select the information to appear in the message. Use double quotes to include a string value, or select a variable by clicking the globe icon. Set a severity level for the message from the message severity list.

Enter or select the message text:

Select the message severity level: WLInfoMessage

Figure 23: Message Dialog Box

2. Create a text message by typing the text you want to appear in the message in the input text box.

Note: When you enter a string value in the message, you must enclose it in quotation marks; for example, “Sample Message”.

3. To add a global variable to the message text, click the globe icon to the right of the input text box, and select a global variable from the drop-down list.
4. Select a severity level for the message from the drop-down list.

   The following severity levels are available:

   - Information message (WLInfoMessage)
   - Minor error message (WLMinorError)
   - Error message (WLError)
   - Severe error message (WLSevereError)
   - Debug message (WLDebugMessage)
5. Click **OK**.

The Message item appears in the Script Tree, and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

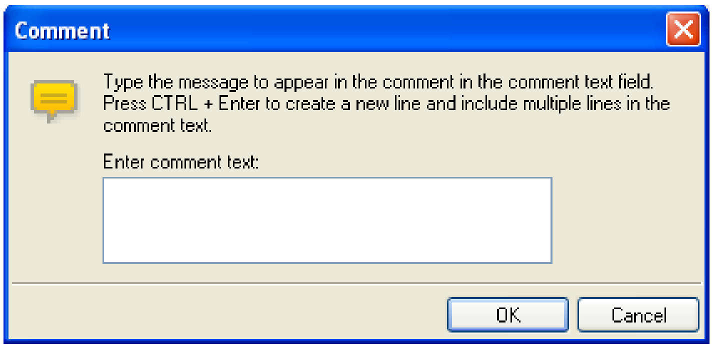
Inserting Comments in a Recording

WebLOAD Recorder enables you to insert comments while recording to describe an activity or provide information about a specific operation.

To insert a comment:

1. Click from the WebLOAD Recording toolbar at the desired location in the recording.

   The Comment dialog box opens.



Comment

Type the message to appear in the comment in the comment text field. Press CTRL + Enter to create a new line and include multiple lines in the comment text.

Enter comment text:

Figure 24: Comment Dialog Box

2. Enter the text you want to appear in the comment.
3. Click **OK**.

   The Comment item appears in the Script Tree, and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Inserting Begin and End Transactions in a Recording

In addition to the automatic transactions provided by WebLOAD, you can add named transactions during a recording to measure the performance of logical actions in your script, such as a Login process. By inserting named transactions into your script, you can take a series of simple actions, define them as a transaction, and set success or failure criteria for the transaction. Each transaction can be a simple action, such as a query, or a complex action that may include several steps.

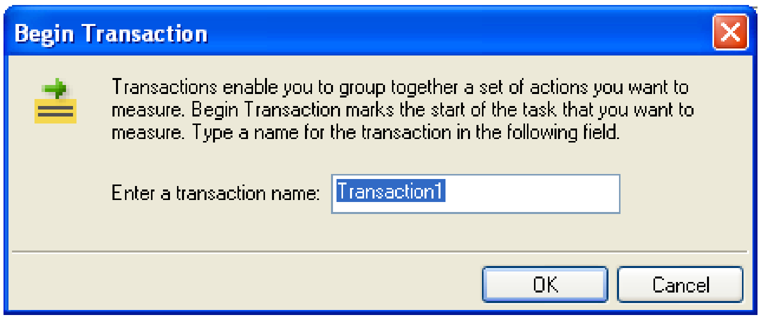
To measure transactions, you must mark the beginning and end of the transaction in your script. During runtime, WebLOAD measures the time it takes to complete the transaction and reports the results in the WebLOAD Integrated reports, Statistics reports, and Data Drilling report.

Note: You can add an unlimited number of transactions into your script. Each transaction must have a different name.

To mark the beginning of a transaction:

1. Click from the WebLOAD Recording toolbar just before the first action you want to include in the transaction.

   The Begin Transaction dialog box opens.



Transactions enable you to group together a set of actions you want to measure. Begin Transaction marks the start of the task that you want to measure. Type a name for the transaction in the following field.

Enter a transaction name: Transaction1

Figure 25: Begin Transaction Dialog Box

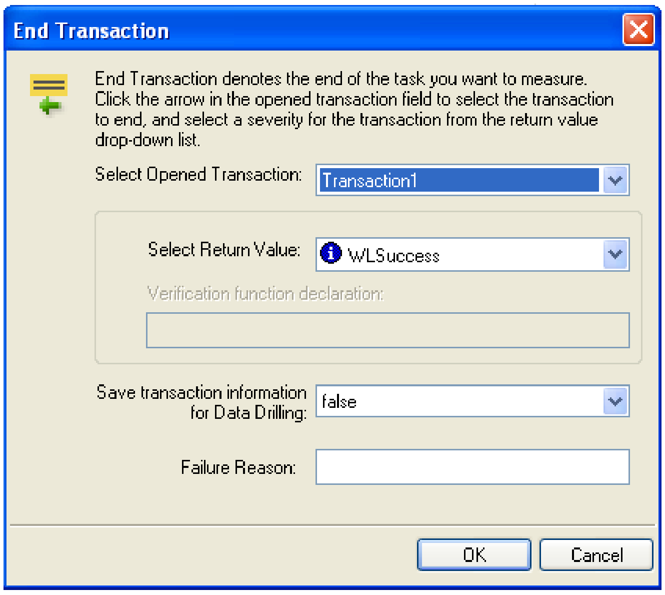
2. Enter a logical name for the transaction; for example, “Login”.
3. Click **OK**.

   The Begin Transaction item appears in the Script Tree, and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

To mark the end of a transaction:

1. Click from the WebLOAD Recording toolbar directly after the last action you want included in the script.

   The End Transaction dialog box opens.



End Transaction denotes the end of the task you want to measure. Click the arrow in the opened transaction field to select the transaction to end, and select a severity for the transaction from the return value drop-down list.

Select Opened Transaction: Transaction1

Select Return Value: WLSuccess

Verification function declaration:

Save transaction information for Data Drilling: false

Failure Reason:

Figure 26: End Transaction Dialog Box

2. Select the transaction to end from the Select Opened Transaction drop-down list.
3. Select a return value for the transaction from the Select Return Value drop-down list.

   You can select from the return values provided, or select **Custom Function** to create your own verification function to call when the transaction is complete.

   For information on creating custom functions, see the *WebLOAD Scripting Guide*.

4. To set WebLOAD to save the results of all transaction instances, successes, and failures for later analysis with Data Drilling, select **true** in the **Save transaction information for Data Drilling** field. Select **false** (default) to save only results of failed transaction instances that triggered some sort of error flag.
5. Optionally, enter a text string to specify a possible reason for a transaction failure within your transaction verification function in the **Failure Reason** field. This reason will also appear in the Statistics Report.
6. Click **OK**.

   The End Transaction item appears in the Script Tree, and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Defining Concurrent in a Recording

WebLOAD Recorder enables you to collect Post and Get HTTP requests and simultaneously execute them by two or more threads, as defined in the MultiThread Virtual Clients number. This is configured in the Browser Parameters tab in WebLOAD Console's Script Options dialog box.

Note: WebLOAD Recorder does not perform the Post and Get HTTP requests concurrently.

To simultaneously execute Post and Get HTTP requests, you must define where in the script to begin collecting the requests and where to stop collecting and begin executing them. The HTTP requests are collected until the engine encounters an `Execute Concurrent` function in the script. For more information about the Execute Concurrent Building Block, see *Executing Concurrent Definition in a* Recording (on page 46).

To define when to start collecting HTTP requests in a script:

- Click from the WebLOAD Recording toolbar at the desired location in the recording.

  The Define Concurrent Building Block is added to the Script Tree. The JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Executing Concurrent Definition in a Recording

WebLOAD Recorder enables you to define the Execute Concurrent function. Then in WebLOAD Console, you can simultaneously execute all the Post and Get HTTP requests that were defined since the last Define Concurrent function by two or more threads, as defined by the MultiThread Virtual Clients number. This is configured in the Browser Parameters tab in WebLOAD Console's Script Options dialog box.

Note: The Execute Concurrent function can only be inserted in your script *after* a Define Concurrent function. For more information about the Define Concurrent function, see *Define Concurrent* (on page 246).

When the engine encounters the `Execute Concurrent` function, it stops collecting the HTTP requests in the script and starts their execution.

To insert concurrently executing HTTP requests in a script:

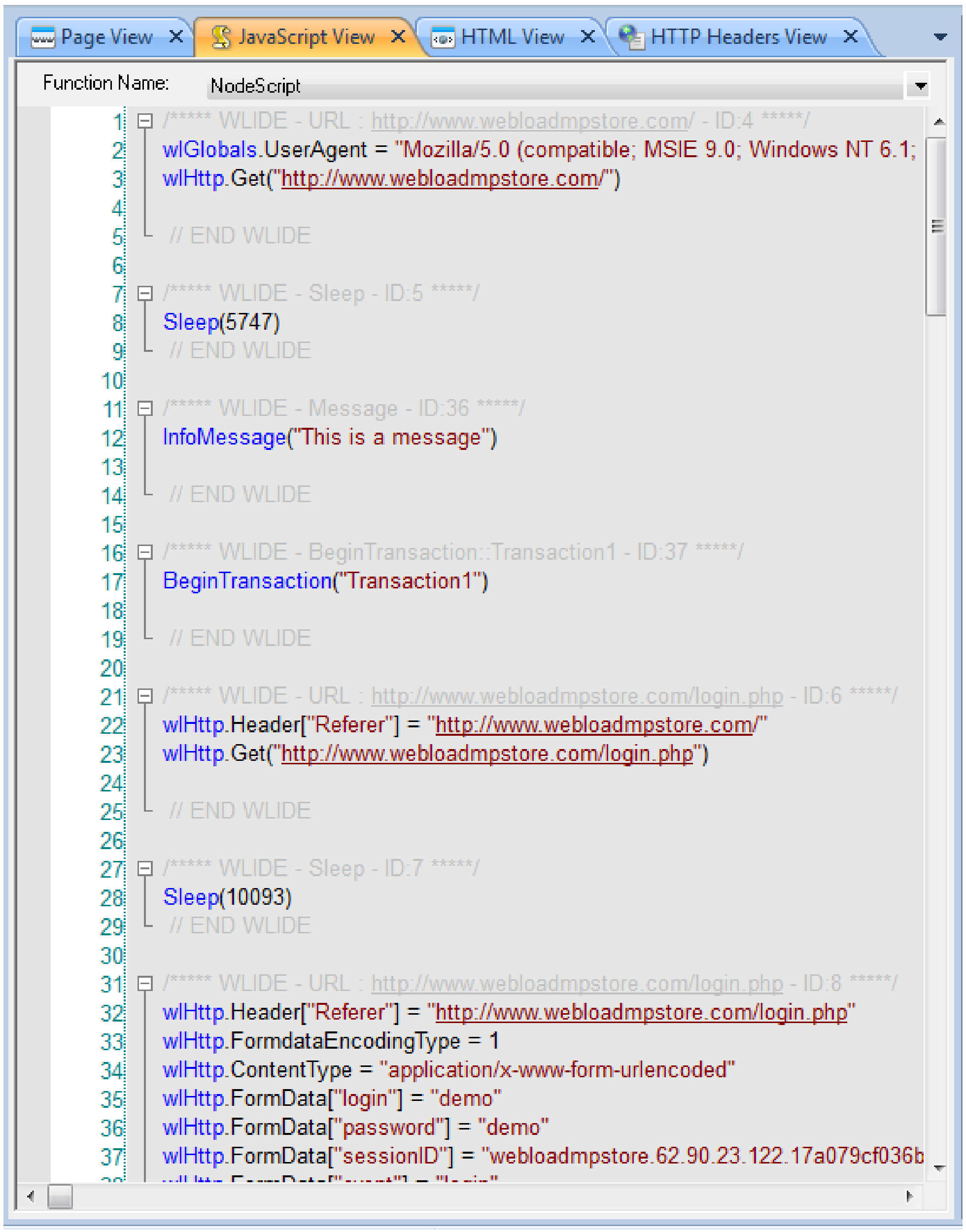
- Click from the WebLOAD Recording toolbar at the desired location in the recording.

The Execute Concurrent Building Block is added to the Script Tree. The JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Viewing the Recorded Script

WebLOAD Recorder creates scripts by recording your actions as you interact with your Web application or System Under Test (SUT). Your recorded script serves as a baseline, which is subsequently used in the WebLOAD environment to test the performance of your Web application.

WebLOAD Recorder presents each recorded action visually in the Script Tree and as code in the JavaScript View pane.



```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:4 *****/
wlGlobals.UserAgent = "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;
wlHttp.Get("http://www.webloadmpstore.com/")

 // END WLIDE

/***** WLIDE - Sleep - ID:5 *****/
Sleep(5747)
 // END WLIDE

/***** WLIDE - Message - ID:36 *****/
InfoMessage("This is a message")

 // END WLIDE

/***** WLIDE - BeginTransaction::Transaction1 - ID:37 *****/
BeginTransaction("Transaction1")

 // END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:6 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/"
wlHttp.Get("http://www.webloadmpstore.com/login.php")

 // END WLIDE

/***** WLIDE - Sleep - ID:7 *****/
Sleep(10093)
 // END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:8 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/login.php"
wlHttp.FormdataEncodingType = 1
wlHttp.ContentType = "application/x-www-form-urlencoded"
wlHttp.FormData["login"] = "demo"
wlHttp.FormData["password"] = "demo"
wlHttp.FormData["sessionID"] = "webloadmpstore.62.90.23.122.17a079cf036b
```

Figure 27: Recorded Actions in JavaScript View Pane

Viewing the Recorded Script in the Script Tree

As you navigate through a Web application, WebLOAD Recorder records your actions.

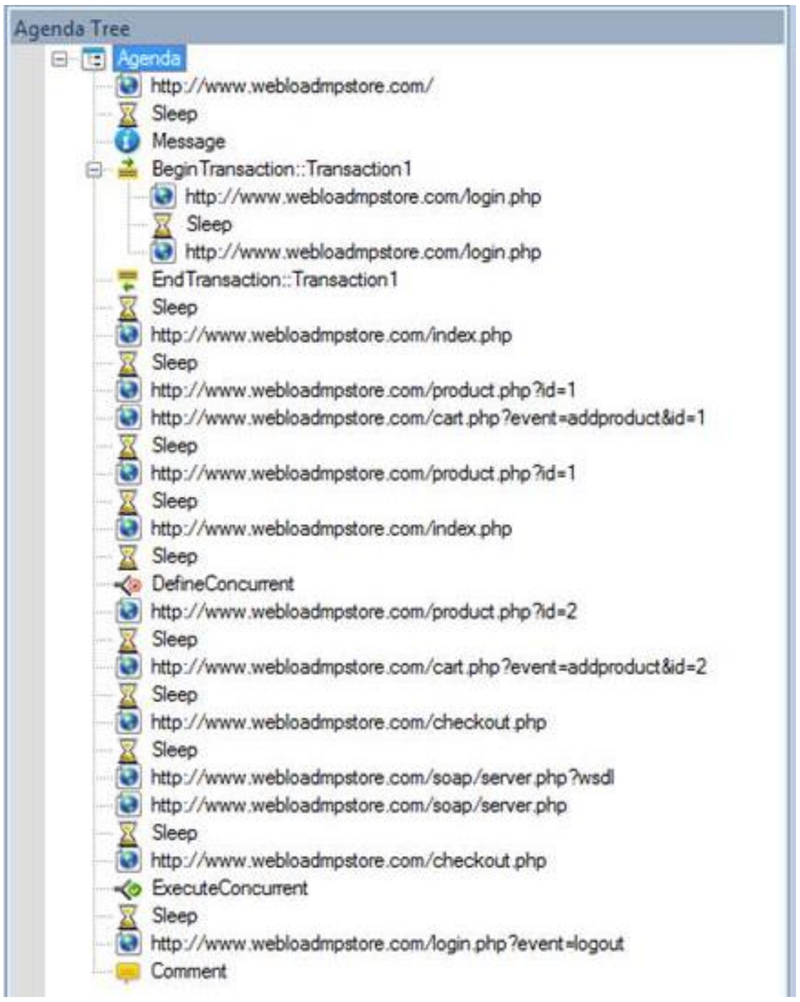

Agenda Tree
- Agenda
  - http://www.webloadmpstore.com/
  - Sleep
  - Message
  - BeginTransaction::Transaction1
    - http://www.webloadmpstore.com/login.php
    - Sleep
    - http://www.webloadmpstore.com/login.php
  - EndTransaction::Transaction1
  - Sleep
  - http://www.webloadmpstore.com/index.php
  - Sleep
  - http://www.webloadmpstore.com/product.php?id=1
  - http://www.webloadmpstore.com/cart.php?event=addproduct&id=1
  - Sleep
  - http://www.webloadmpstore.com/product.php?id=1
  - Sleep
  - http://www.webloadmpstore.com/index.php
  - Sleep
  - DefineConcurrent
  - http://www.webloadmpstore.com/product.php?id=2
  - Sleep
  - http://www.webloadmpstore.com/cart.php?event=addproduct&id=2
  - Sleep
  - http://www.webloadmpstore.com/checkout.php
  - Sleep
  - http://www.webloadmpstore.com/soap/server.php?wsdl
  - http://www.webloadmpstore.com/soap/server.php
  - Sleep
  - http://www.webloadmpstore.com/checkout.php
  - ExecuteConcurrent
  - Sleep
  - http://www.webloadmpstore.com/login.php?event=logout
  - Comment

Figure 28: Script Tree with Nodes

WebLOAD Recorder displays the following as nodes in the Script Tree:

- Pages accessed
- Sleep time
- Messages
- Comments
- Begin Transaction
- End Transaction
- Define Concurrent
- Execute Concurrent

When navigating to a new page in the Web application, WebLOAD Recorder inserts a node with the URL into the Script Tree.

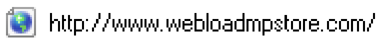

http://www.webloadmpstore.com/

Figure 29: Page Access Node

When you pause while navigating in the Web application, WebLOAD Recorder inserts a Sleep node into the Script Tree. The Sleep node represents your thinking time.

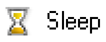

Sleep

Figure 30: Sleep Node

When you insert a message while recording, WebLOAD Recorder inserts a Message node into the Script Tree.

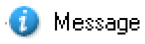

Message

Figure 31: Message Node

When you insert a comment while recording, WebLOAD Recorder inserts a Comment node into the Script Tree.

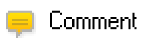

Comment

Figure 32: Comment Node

When you begin a transaction while recording, WebLOAD Recorder inserts a BeginTransaction node into the Script Tree. All actions that occur during the Transaction appear on a lower hierarchical level in the Script Tree. You can collapse or expand this node by clicking ⊟ or ⊞ respectively.

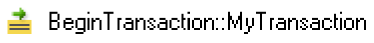

BeginTransaction::MyTransaction

Figure 33: BeginTransaction Node

When you end a transaction while recording, WebLOAD Recorder inserts an EndTransaction node into the Script Tree.

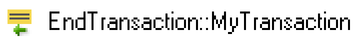

EndTransaction::MyTransaction

Figure 34: EndTransaction Node

When you define concurrent while recording, WebLOAD Recorder inserts a DefineConcurrent node into the Script Tree.

Figure 35: DefineConcurrent Node

When you define concurrent while recording, WebLOAD Recorder inserts an ExecuteConcurrent node into the Script Tree.

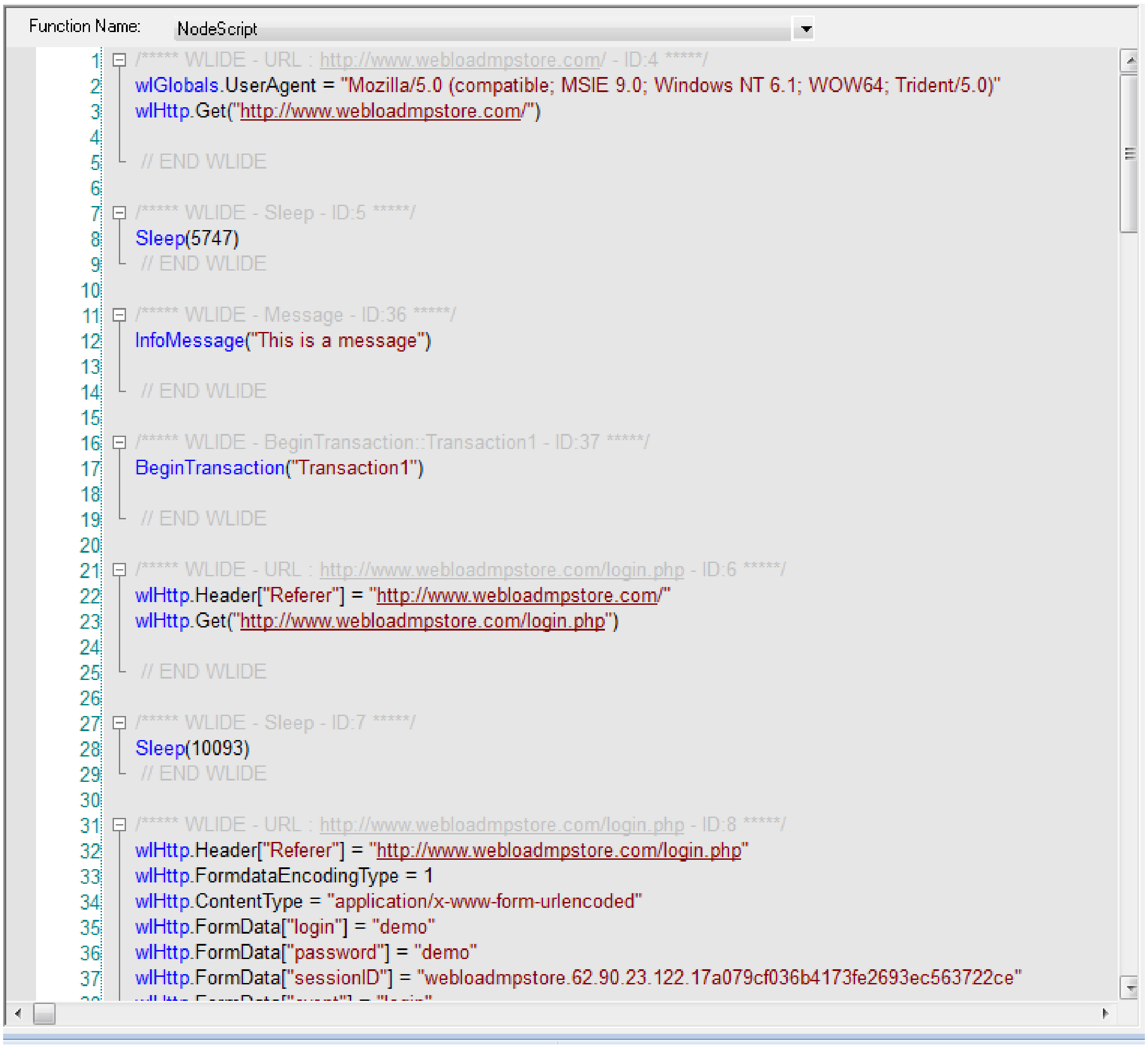
Figure 36: ExecuteConcurrent Node

After the script is recorded, you can edit the script (see *Editing a Script in the Script Tree* on page 70).

After your script has been developed, you can run and debug it. While the script is running, you can view it in the Script Tree (see *Viewing the Execution Sequence in the Script Tree* on page 112).

Viewing the Recorded Script in the JavaScript View Pane

Each node in the Script Tree is actually a visual representation of JavaScript code. You can view the contents of the nodes in the JavaScript view pane.



Function Name: NodeScript

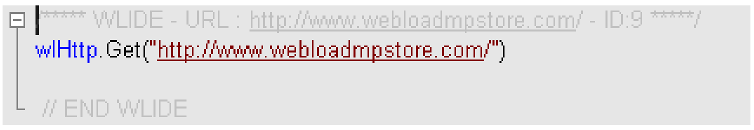
```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:4 *****/
wlGlobals.UserAgent = "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)"
wlHttp.Get("http://www.webloadmpstore.com/")

 // END WLIDE

/***** WLIDE - Sleep - ID:5 *****/
Sleep(5747)
 // END WLIDE

/***** WLIDE - Message - ID:36 *****/
InfoMessage("This is a message")

 // END WLIDE

/***** WLIDE - BeginTransaction::Transaction1 - ID:37 *****/
BeginTransaction("Transaction1")

 // END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:6 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/"
wlHttp.Get("http://www.webloadmpstore.com/login.php")

 // END WLIDE

/***** WLIDE - Sleep - ID:7 *****/
Sleep(10093)
 // END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:8 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/login.php"
wlHttp.FormdataEncodingType = 1
wlHttp.ContentType = "application/x-www-form-urlencoded"
wlHttp.FormData["login"] = "demo"
wlHttp.FormData["password"] = "demo"
wlHttp.FormData["sessionID"] = "webloadmpstore.62.90.23.122.17a079cf036b4173fe2693ec563722ce"
```

Figure 37: Node Contents in JavaScript View Pane

In the JavaScript view pane, you can do the following:

- Display the code for each node individually.
- View code for the entire script as a whole.
- View the code for different sections in the script, by clicking the Agenda root node in the Script Tree and selecting a section from the Function Name list at the top of the JavaScript view pane.

Each block of code starts with a header that contains "WLIDE", description, and ID number, and ends with the "END WLIDE" closing comment. The ID number is automatically generated by WebLOAD Recorder and is the connection between the script node and the specific header. The comments in the script are a light grey color.



```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:9 *****/
wlHttp.Get("http://www.webloadmpstore.com/")

// END WLIDE
```

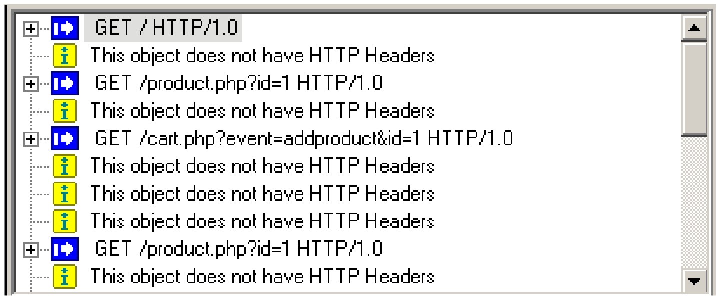
Figure 38: Block of JavaScript Code

After the script is recorded, you can edit the script (see *Editing a Script in the JavaScript View Pane* on page 72).

After your script has been developed, you can run and debug it. While the script is running, you can view it in the JavaScript View pane (see *Viewing the Execution Sequence in the JavaScript View Pane* on page 113).

Viewing the Recorded Script in the HTTP Headers View Pane

Each node in the Script Tree is also a visual representation of response headers. You can view the headers of the nodes in the HTTP Headers view pane.



```
GET / HTTP/1.0
This object does not have HTTP Headers
GET /product.php?id=1 HTTP/1.0
This object does not have HTTP Headers
GET /cart.php?event=addproduct&id=1 HTTP/1.0
This object does not have HTTP Headers
This object does not have HTTP Headers
This object does not have HTTP Headers
GET /product.php?id=1 HTTP/1.0
This object does not have HTTP Headers
```

Figure 39: HTTP Headers View Pane

In the HTTP Headers view pane, you can do the following:

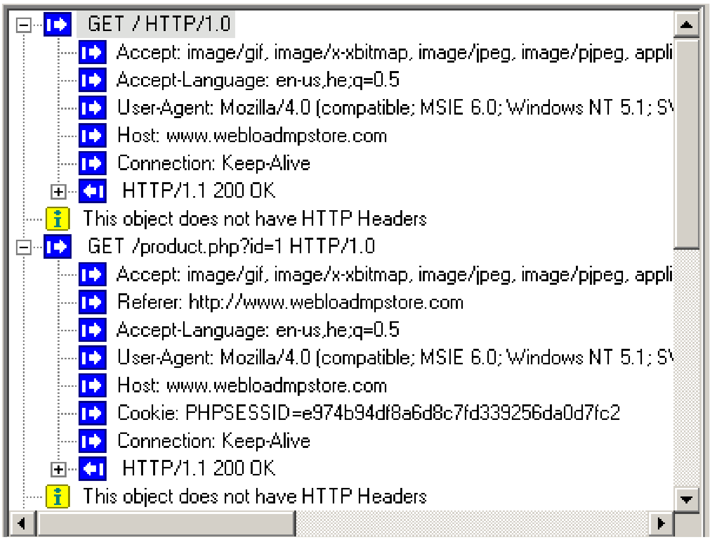
- Display the header for each node individually.
- View headers for the entire script as a whole.

When you click a node in the Script Tree, you can view the header for that node.

GET / HTTP/1.0

Figure 40: Node Header

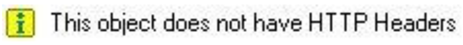
You can expand the header to view all of the gets and posts for that node.



```
GET / HTTP/1.0
    Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, appli
    Accept-Language: en-us,he;q=0.5
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV
    Host: www.webloadmpstore.com
    Connection: Keep-Alive
    HTTP/1.1 200 OK
This object does not have HTTP Headers
GET /product.php?id=1 HTTP/1.0
    Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, appli
    Referer: http://www.webloadmpstore.com
    Accept-Language: en-us,he;q=0.5
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV
    Host: www.webloadmpstore.com
    Cookie: PHPSESSID=e974b94df8a6d8c7fd339256da0d7fc2
    Connection: Keep-Alive
    HTTP/1.1 200 OK
This object does not have HTTP Headers
```

Figure 41: Expanded Header

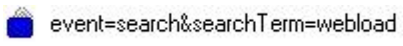
For an object that does not have a header, such as a Sleep node, the following is displayed:



This object does not have HTTP Headers

Figure 42: Display for Objects without Headers

The `Post` command involves sending data to the HTTP server, as opposed to the `Get` command, which is used only for retrieving data. In the HTTP Headers View, the `Post` command also includes the actual data that was sent to the server. This part of the HTTP message is marked by a special icon:



event=search&searchTerm=webload

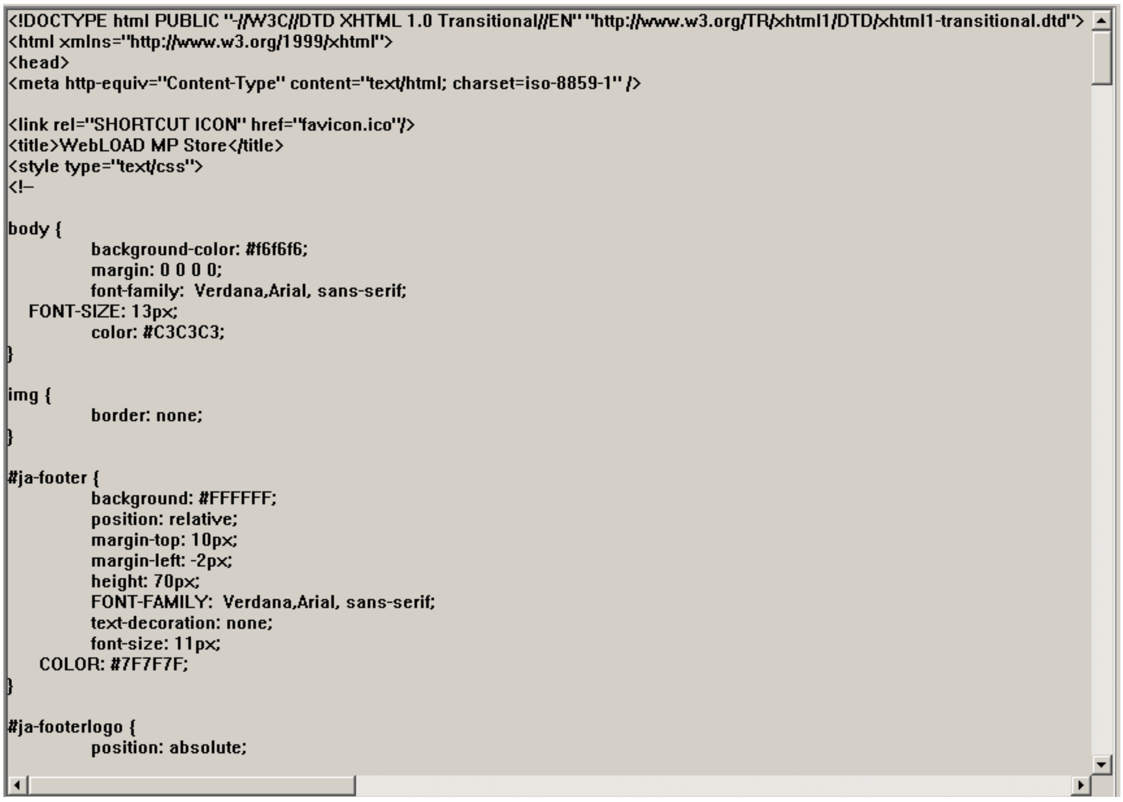
Figure 43: Event Header

After the script is recorded, you can edit the script (see *Editing a Script in the JavaScript View Pane* on page 72).

After your script has been developed, you can run and debug it. While the script is running, you can view it in the JavaScript View pane (see *Viewing the Execution Sequence in the JavaScript View Pane* on page 113).

Viewing the Recorded Script in the HTML View Pane

Each node in the Script Tree is actually a visual representation of HTML code. You can view the HTML preview of each page and frame requested in the script in the HTML view pane.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<link rel="SHORTCUT ICON" href="favicon.ico"/>
<title>WebLOAD MP Store</title>
<style type="text/css">
<!–

body {
            background-color: #f6f6f6;
            margin: 0 0 0 0;
            font-family:  Verdana,Arial, sans-serif;
   FONT-SIZE: 13px;
            color: #C3C3C3;
}

img {
            border: none;
}

#ja-footer {
            background: #FFFFFF;
            position: relative;
            margin-top: 10px;
            margin-left: -2px;
            height: 70px;
            FONT-FAMILY:  Verdana,Arial, sans-serif;
            text-decoration: none;
            font-size: 11px;
    COLOR: #7F7F7F;
}

#ja-footerlogo {
            position: absolute;
```

Figure 44: Node Contents in HTML View Pane

In the HTML view pane, you can display the code for each node individually.

After your script has been developed, you can run and debug it. For more information on debugging with the HTML View, see *Using the HTML View to View Results* (on page 133).

Performing Script Regeneration

When a script is recorded, all of the HTTP traffic is saved even if not all of it is used to generate the script (the recorded traffic is saved to the `.wle` file of the script). The script is created according to the settings defined in the Script Generation tab in the Record and Script Generation Options dialog box. You can regenerate the script any time after recording, to include additional traffic information that was originally recorded. This is done by modifying the settings in the Script Generation tab and then regenerating the script.

For example, by default when a script is recorded, the HTTP headers settings for the HTTP requests are not displayed in the script even though they are recorded. After selecting the **Generate All Headers** checkbox in the Script Generation tab and regenerating the script, the script includes the wlHttp headers property. For more information on the script content that can be regenerated, see *Specifying the Script Content to be Generated* on page 176.

In addition, script regeneration is affected by changes in the settings defined in the Post Data tab in the Record and Script Generation Options dialog box. You can record a script with content type x in the DATA list, DATAFile list, or not in any list, (which means it will be recorded as FORMDATA) and then change the settings and regenerate the script and it will play back according to the new settings.

To perform script regeneration:

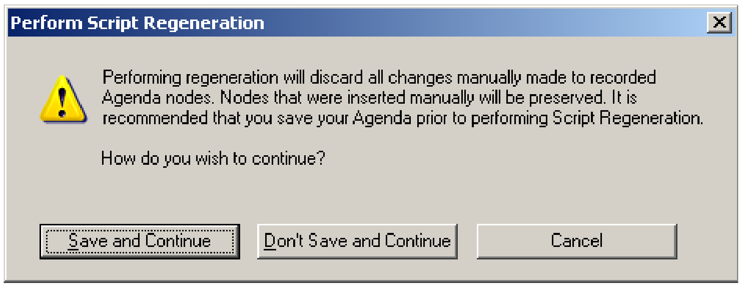
1. Click **Correlation** in the **Home** tab of the ribbon and select **Regenerate Script** from the drop-down list.

   -Or-

   Right-click a node in the Script Tree and select **Regenerate Script** from the pop-up menu. This regenerates only the selected node.

Note: If your script was created manually, and not recorded, WebLOAD informs you that your script does not contain any recorded nodes and cannot regenerate the script. When performing script regeneration, any modifications to the script's originally recorded nodes are lost. Any nodes that were added after the recording will remain in the script after the script is regenerated.

   The Perform Script Regeneration dialog box appears.


Perform Script Regeneration

Performing regeneration will discard all changes manually made to recorded Agenda nodes. Nodes that were inserted manually will be preserved. It is recommended that you save your Agenda prior to performing Script Regeneration.

How do you wish to continue?

Figure 45: Perform Script Regeneration Dialog Box

2. Click **Save and Continue** to save the changes in your script and regenerate the script.

   -Or-

   Click **Don't Save and Continue** to regenerate the script without saving the changes in your script.

-Or-

Click **Cancel** to close the Perform Script Regeneration dialog box without regenerating the script.

Note: Click **Edit** ➤ **Undo** to discard the newly regenerated script and revert back to the previous script.

Saving a Script

You must save your scripts so that you can use them in test sessions.

To save a script:

1. Click **Save** in the **File** tab of the ribbon and select **Save** or **Save As**.

   The Save As dialog box appears.

2. Type the script name in the **File name** field.
3. Click **Save**.

   Your script is saved with the file extension `*.wlp`. You may now run a test using the script.

Saving Additional Project Information

The Additional Information dialog box provides details about the project that help identify it; for example:

- A descriptive title
- The author name
- The subject of the test
- The system under test
- Other important information about the project

Use the Additional Information dialog box to save information about the project.

To save additional information properties for the project:

1. Select **Additional Information** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Project Additional Information dialog box opens.

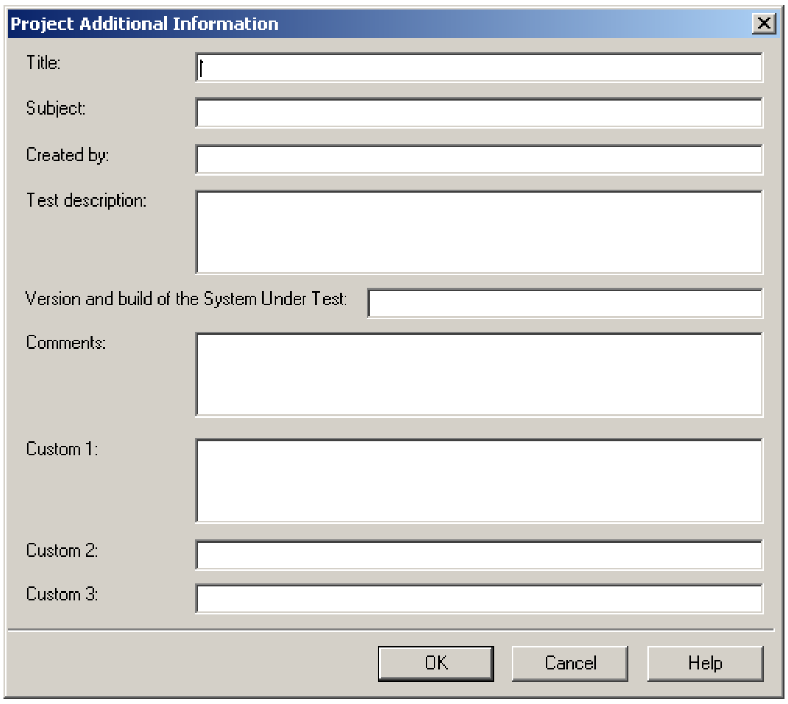


Figure 46: Project Additional Information Dialog Box

2. Fill in the fields to save additional information, useful for later reference, with the project.
3. Click **OK**.

The following table describes the fields of the Project Additional Information dialog box.

Table 5: Project Additional Information Dialog Box Fields

Field	**Description**
Title	Provides a space for you to type a title for this project. The title can be different then the project file name.
Subject	Provides a space for you to type a description of the subject of the project. Use this property to group similar projects together.
Created by	Provides a space for you to type the name of the person who authored this project.
Test description	Provides a space for you to type a description of the test objectives and what the project is designed to test.
Version and build of the System Under Test	Provides a space for you to type the name, version and build number of the System Under Test (SUT).
Comments	Provides a space for you to type any comments regarding the project.
Custom	Provides a space for you to type any comments you want saved with this project.

Recording Desktop Web Applications

Desktop web applications, such as Rich Internet Applications (RIAs), are web based applications that have the features and functionality of local desktop applications. A desktop web application can run either in a web browser or in software that is installed and run locally on the user's desktop (for example, Adobe AIR applications).

Recording desktop web applications in scripts using WebLOAD Recorder, involves configuring the web applications to use a specific proxy setting. This proxy setting is usually configured automatically when opening WebLOAD Recorder for browser-based applications in Internet Explorer or Mozilla Firefox. To record any other web application that does not run within the browser, configure the web application to pass the traffic to the server through the WebLOAD Recorder proxy server, using one of the following methods:

- *Recording WebLOAD Scripts Using the Client's Proxy Setting* (on page 59).
- *Recording WebLOAD Scripts Using the LAN Settings* (on page 59).
- *Recording WebLOAD Scripts Using Proxy Tunneling* (on page 61).

Recording WebLOAD Scripts Using the Client's Proxy Setting

Web applications that support working through a proxy server can be configured to pass the traffic to the server through the WebLOAD Recorder proxy server. Follow the web application's proxy setting instructions to specify the WebLOAD Recorder proxy server as the application's proxy server.

WebLOAD Recorder's default proxy server settings are:

- Host: localhost.
- Port: 9884.

Recording WebLOAD Scripts Using the LAN Settings

Some operation systems provide the ability to configure the proxy setting of the LAN connection. The following example demonstrates configuring the Internet application's proxy setting of the LAN connection in Windows XP:

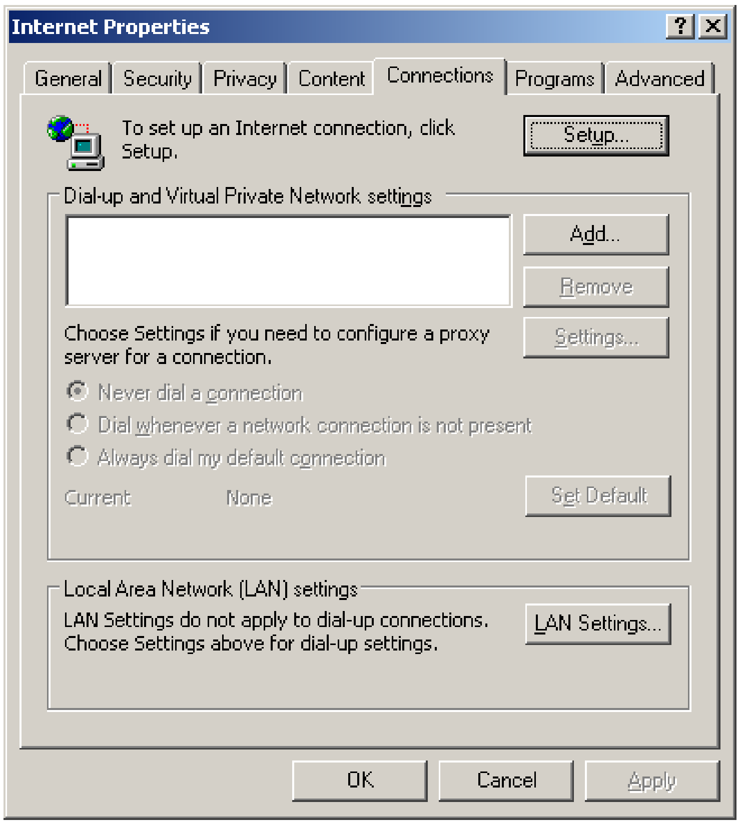
To configure the proxy setting in Windows XP:

1. From the **Start** menu, select **Settings** > **Control Panel**.

   The Control Panel dialog box appears.

2. Select **Internet Properties** and then select the **Connections** tab.

The Internet Properties – Connections tab appears.


Internet Properties

General | Security | Privacy | Content | Connections | Programs | Advanced

To set up an Internet connection, click Setup. [Setup...]

Dial-up and Virtual Private Network settings [Add...] [Remove] [Settings...]

Choose Settings if you need to configure a proxy server for a connection.

[● Never dial a connection]
○ Dial whenever a network connection is not present
○ Always dial my default connection

Current None [Set Default]

Local Area Network (LAN) settings

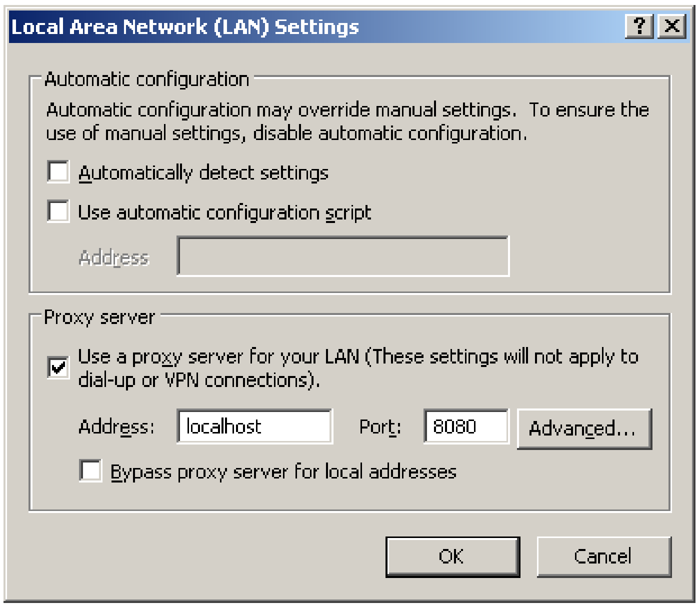
LAN Settings do not apply to dial-up connections. Choose Settings above for dial-up settings. [LAN Settings...]

[OK] [Cancel] [Apply]

Figure 47: Internet Properties – Connections Tab

3. In the Local Area Network (LAN) settings, click **LAN settings**.

   The Local Area Network (LAN) Settings dialog appears.


Local Area Network (LAN) Settings

Automatic configuration

Automatic configuration may override manual settings. To ensure the use of manual settings, disable automatic configuration.

☐ Automatically detect settings
☐ Use automatic configuration script

Address

Proxy server

[✓ Use a proxy server for your LAN (These settings will not apply to dial-up or VPN connections).]

Address: localhost Port: 8080 [Advanced...]

☐ Bypass proxy server for local addresses

[OK] [Cancel]

Figure 48: Local Area Network (LAN) Settings

4. In the Proxy server area check **Use a proxy server for your LAN**.
5. In the Address and Port fields enter WebLOAD Recorder's proxy server setting. By default, this is localhost:9884.

Note: This setting is necessary for the recording process only and should be removed before the load test execution.

6. Click **OK**.

   The proxy setting of the LAN connection in Windows XP is configured.

Recording WebLOAD Scripts Using Proxy Tunneling

Proxy tunneling is a general method to handle desktop web applications that do not support working through a proxy server. Proxy tunneling involves using an external utility to redirect the outgoing web traffic on the client machine. The redirection enforces the traffic to pass through WebLOAD's proxy recorder, which is listening on port 9884, during the recording stage.

To record scripts using Proxy Tunneling:

1. Enable a port interception service on the client machine. Configure the service to redirect the outgoing traffic from the application to pass through WebLOAD's proxy. The following are possible methods for intercepting web application traffic:
   - Use an external utility that is capable of controlling and rerouting HTTP traffic. For example, the Proxifier application, which is available at: http://www.proxifier.com.
   - Configure your firewall. Some firewalls provide advanced services of traffic manipulation.
   - Configure your hardware. Four to seven layer switches may be used for controlling and routing web based traffic.
2. Configure WebLOAD Recorder's recording options as follows:
   a. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

      The Recording and Script Generation Options dialog appears (see Figure 114).

   b. Select the Browser Settings tab.

      The Browser Settings tab appears (see Figure 119).

   c. In the Automatic Browser Settings area, uncheck **Set the Proxy definitions automatically**.

   d. Select the Proxy Options tab.

      The Proxy Options tab appears (see Figure 125).

e. In the Recording Proxy Options area, check **Use Transparent Proxy**. This enables WebLOAD Recorder to record from any Web client that does not support proxy configurations.

3. Start recording your script.
4. Run the web application. While the web application is running, all the http traffic generated on the client machine is directed to WebLOAD Recorder. WebLOAD handles this traffic as if it is received from a browser client and generates the appropriate script.
5. Stop recording the script when the application is finished running.
6. Disable the interception service.

Note: Before starting the test and running the generated script, the interception service must be stopped. Otherwise, the load traffic generated by the Console will be directed back to the recorder.

The script is recorded successfully. You can now run the test in WebLOAD Console.

In certain cases, the WebLOAD Proxy Recorder may timeout during recording. This may be due to a slow network and/or SUT. The default timeout is 60. To avoid this situation, you can increase the default timeout.

To change the default timeout:

1. Right-click the `wlproxyinclude.js` file in <RadView directory>\Include and select **Edit**.
2. Add the following line to the file:

```
ProxyObject.RProxyCOptConnectionTimeOut = 300
```

3. Save the file.

Troubleshooting

Refer to the following table if you are having recording related problems. Before assuming the problem is with WebLOAD, make sure that your internet settings are correct and that you can access the internet without recording.

Table 6: Troubleshooting

Problem	Possible Options	Solution
Script is not created while recording	Make sure browser opens while recording.	Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon and select the **Browser Settings** tab. Ensure that the settings are correct for your browser. See *Configuring the Default Browser* (on page 186) for more information.
	Check browser proxy settings.	**For Internet Explorer:** 1. From your browser's menu, select **Tools ➤ Internet Options** and select the **Connections Options** tab. 2. Click **Lan Settings**. 3. If none of the checkboxes are selected, you have a direct connection to the internet and the browser proxy settings are not the problem. If **Automatically detect settings** and/or **Use automatic configuration script** are checked, you must disable the automatic settings. Before disabling the automatic settings, contact your system administrator for your proxy server information. If **Use a proxy server for your LAN** is checked, copy the Address and Port field's current proxy settings. In WebLOAD, click **Recording and Script Generation Options** in the **Tools** tab of the ribbon and select the **Proxy Options** tab. Check **Use the following definitions for the application's proxy server** and enter the current proxy information into the HTTP Proxy/Port and SSL Proxy/Port fields.

Problem	Possible Options	Solution	
		For Mozilla Firefox:	1. From your browser's menu, select **Tools** > **Options**. 2. At the top of the Options dialog box, select the **Advanced** icon and select the **Network** tab. 3. In the Connection area, click **Settings**. 4. If **Direct connection to the internet** is selected, the browser proxy settings are not the problem. If **Auto-detect proxy settings for this network** or **Automatic proxy configuration URL** are selected, you must disable the automatic settings. Before disabling the automatic settings, contact your system administrator for your proxy server information. If **Manual proxy configuration** is selected, copy the **HTTP Proxy** and **Port** field's current proxy settings. In WebLOAD, click **Recording and Script Generation Options** in the **Tools** tab of the ribbon and select the **Proxy Options** tab. Check **Use the following definitions for the application's proxy server** and enter the current proxy information into the HTTP Proxy/Port and SSL Proxy/Port fields.
	Recording with a browser other than Internet Explorer or Mozilla Firefox.	1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon and select the **Browser Settings** tab. Select **Other browser** or **None** as the default browser setting. 2. While recording, you must manually change the client's proxy setting. In your browser, manually configure the proxy settings to use WebLOAD's default port: 9884. This points the browser's proxy settings to the WebLOAD recorder enabling WebLOAD to record the browser's HTTP clients.	
	LAN settings options for IE are disabled by IT Group Policies	Request that your Network Administrator change the policies for the specific machine on which you are recording. For more information about Group Policy see the Microsoft TechNet Library.	

Problem	Possible Options	Solution

	Advanced options	1. While recording, open the task manager and ensure that a single `proxynator.exe` process is active. If there is no `proxynator.exe` process or there is more than one `proxynator.exe` process contact RadView support for further assistance.
		2. WebLOAD uses ports 9884, 9010, and 9000. Enter `netstat -a -p tcp` in the command line to ensure that the ports are not being used by another application on your machine. A list of the unavailable ports appears.
		If port 9884 is unavailable, click **Recording and Script Generation Options** in the **Tools** tab of the ribbon and select the **Proxy Options** tab. In the **Recording Proxy Options** frame, modify the Proxy Port value from 9884 to an available port number.
		If port 9010 or port 9000 are unavailable, open the WebLOAD.ini file in `<RadView directory>\bin` and locate the following lines:
		`TESTTALK_CLIENT_PORT="9010"`
		`TESTTALK_NETWORK_PORT="9001"`
		Modify the port value of the unavailable port from 9010 and/or 9000 to an available port number.

Local sites are not recorded in the Script	Proxy settings are set to bypass local addresses	1. From your browser's menu, select **Tools** > **Internet Options** and select the **Connections Options** tab. 2. Click **Lan Settings**. 3. Check **Bypass proxy server for local addresses** to ensure that the proxy settings are not set to bypass local addresses or any other server that you want to record.
	WebLOAD does not record from `http://localhost`	**For Internet Explorer:** Use any of the following instead of `http://localhost`: • `http://<your machine name>` • `http://<your IP address>` • `http://localhost./` **For Mozilla Firefox:** 1. From your browser's menu, select **Tools** > **Options**. 2. Select the **Advanced** > **Network** tab and click **Settings**. 3. Clear the **No proxy for property** checkbox.

Problem	Possible Options	Solution
Secured sites are not recorded in the script	Proxy settings do not point to the recorder.	**For Internet Explorer:** 1. From your browser's menu, select **Tools** > **Internet Options** and select the **Connections Options** tab. 2. Click **Lan Settings**. 3. Ensure that **Use a proxy server for your LAN** is checked and modify the port setting to 9884. **For Mozilla Firefox:** 1. From your browser's menu, select **Tools** > **Options**. 2. At the top of the Options dialog box, select the **Advanced** icon and select the **Network** tab. 3. In the Connection area, click **Settings**. 4. Ensure that **Manual proxy configuration** is selected and modify the port setting to 9884.
	A different proxy is needed for SSL	You must configure an SSL proxy. For instructions, see *Setting the Proxy Options* on page 200.
Certificate Error is displayed in the browser during recording		The browser correctly detects the recorder and warns the user. You can safely ignore the warning and continue. **Note:** You can prevent the warning if you configure WebLOAD to use the server's certificate. Set the proxy certificate options in the Recording and Script Generation Options dialog box. For more information see *Setting the Proxy Certificates* (on page 204).
A partial script is created while recording	Browser cache needs to be cleared.	**For Internet Explorer or Mozilla Firefox:** 1. In WebLOAD Recorder, click **Recording and Script Generation Options** in the **Tools** tab of the ribbon. 2. Select the **Browser Settings** tab. 3. In the Automatic Browser Settings area, check **Clear the browser cache** and click **OK.** **For any other browser:** • From your browser's menu, select the command that clears the browser's cache.

Problem	Possible Options	Solution
	Proxy settings need to be modified	Modify the proxy setting's file extensions and content types to record specific extensions, since WebLOAD by default only records top-level URLs, such as HTML, XML, and text. 1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon. 2. In the Recording and Script Generation Options dialog box select the **Content Types** and **File Locations** tabs. 3. Add the specific extension and content types that are not being recorded. For more information see *Configuring the Content Types to Record* (on page 198) and *Setting File Locations* (on page 210).
The Internet Explorer proxy settings are locked		1. Click **Start** > **Run**. 2. Type `Regedit` and click **OK**. 3. Select **HKEY_CURRENT_USER** > **Software** > **Policies** > **Microsoft** > **Internet Explorer** > **Control Panel**. 4. Set the data value for each key in this directory to `0`.

Chapter 6

Editing Scripts

This section provides instructions for editing scripts with WebLOAD Recorder.

About Editing Scripts with WebLOAD Recorder

WebLOAD Recorder is both flexible and extendable to fit all of your script editing needs, from the most basic to the most advanced. On the simplest level, you use the WebLOAD Recorder GUI to record your basic script. You can edit your script either while it is being recorded or after it has finished recording to add functionality through the options available in the GUI. In most cases, the options available through the GUI meet all testing needs. For advanced functionality where programming is required, the JavaScript Editor is available to add further functionality to your script.

In the script, each request and event is based on previous input, tying the entire script into a whole, making many actions interdependent. Items such as JavaScript Objects, Comments, Messages, and Sleeps can be added to the script, but changing the sequence of items in effect means changing the sequence of activities, and may destroy the functionality of the script. For more information on recording scripts, see *Recording Scripts* (on page 33).

When editing your script, you can work at whatever level you prefer.

The following script editing tools are discussed:

- *Editing a Script in the Script Tree* (on page 70) describes how to add script items and JavaScript Objects, and edit a script by right-clicking in the Script Tree.
- *Editing a Script in the JavaScript View Pane* (on page 72) describes how to use JavaScript objects to create scripts with the full functionality of JavaScript code programs. The WebLOAD Recorder JavaScript Editor includes a set of context-sensitive prompts that help you code your script more effectively.
- *Editing your Script Using the WebLOAD Recorder Toolbox Set* (on page 85) describes how to use the WebLOAD Recorder toolbox that contains drag-and-drop items to create a script with minimal coding.

Editing a Script in the Script Tree

This section describes how to edit a script in the Script Tree. If you are editing your script while it is being recorded, you can focus on any specific node in the Script Tree and edit its JavaScript in the JavaScript view pane.

Note: You must be in Visual Editing mode.

Adding Script Items and JavaScript Objects to a Script

WebLOAD Recorder contains shortcuts to frequently performed actions. This section describes how to place script items and JavaScript Objects from the **Insert** menu into a script. For guidelines for replacing the placeholder variables with your own, see *Guidelines for Editing JavaScript Code* (on page 83).

To add items and JavaScript Objects to a script:

1. In the main window, click **Open** in the **File** tab of the ribbon and open the script you want to edit.
2. Make sure that you are in Visual Editing mode.
3. Right-click the Agenda root node or the script item where you want to place the new script item.

   A pop-up menu appears.

4. From the pop-up menu, click **Insert**.

   The following list of shortcuts appears.

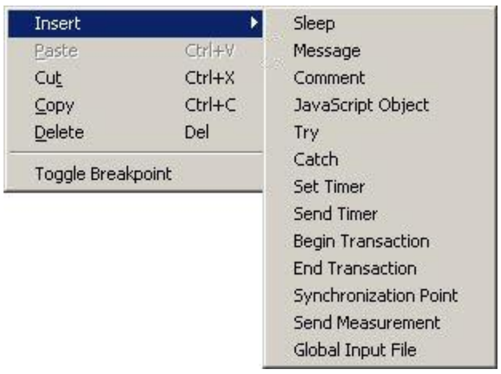


Figure 49: Insert Menu

5. Select a script item or JavaScript Object.

   The script item or JavaScript is inserted on a new line in the script, immediately after the selected node.

The script items and JavaScript Objects that you can insert are also available through the WebLOAD Recorder toolbox, see *Editing your Script Using the WebLOAD Recorder Toolbox Set* (on page 85).

Editing a Script by Right-Clicking in the Script Tree

You can edit directly in the Script Tree using the right mouse button. When you right-click a script item, a menu gives you options that vary according to the script item selected and the mode.

To right-click menus in Edit mode:

1. In the main window, click **Open** in the **File** tab of the ribbon and open the script you want to edit.
2. Make sure that you are in Visual Editing mode.
3. In the Script Tree, right-click the Agenda root node or right-click a script item in the tree.

   A pop-up menu appears. The menu for the Agenda root differs slightly from the menu for a script item, as described in *Table 7*.

   The following table describes the menu options:

Table 7: Menu Options

Right-Click Menu Option	Purpose
Synchronize (Agenda root menu only)	Synchronize the Script Tree, with the edits made to the JavaScript code in Visual Editing mode In most cases, synchronization is performed automatically. Only available at the Agenda root level.
Insert	Insert a script item or JavaScript Object into the script (see *Adding Script Items and JavaScript Objects to a Script* on page 70). The script items and JavaScript Objects that you can insert are also available through the WebLOAD Recorder toolbox, described in *Editing your Script Using the WebLOAD Recorder Toolbox Set* (on page 85).

Right-Click Menu Option	Purpose
Paste	Paste the script item you cut or copied, after the current script item. **Note:** If you copied a script item, you can paste it more than once. Each time you paste, the node ID automatically changes. If you cut a script item, you can paste it only once, and the node ID does not change.
Cut (script item menu only)	Cut the script item from the tree to paste elsewhere.
Copy (script item menu only)	Copy the script item from the tree to paste elsewhere.
Delete (script item menu only)	Delete the script item from the tree.
Toggle Breakpoint	Add or remove a breakpoint at the selected script item in the Script Tree. For more information, see *Setting Breakpoints* (on page 121).
Current Project Options	Display the Current Project Options dialog box. Only available at the script level. For more information, see *Configuring the Default and Current Project Options* (on page 155).
Regenerate Script	Regenerate the script. For more information, see *Performing Script Regeneration* (on page 55).
Response Validation	Add response validation to the script. For more information, see *Validating Responses* (on page 139)

Editing a Script in the JavaScript View Pane

You can edit directly in the JavaScript View pane using the right mouse button. When you right-click a script item, a menu gives you options that vary according to the mode.

Editing the JavaScript Code for a Script Item

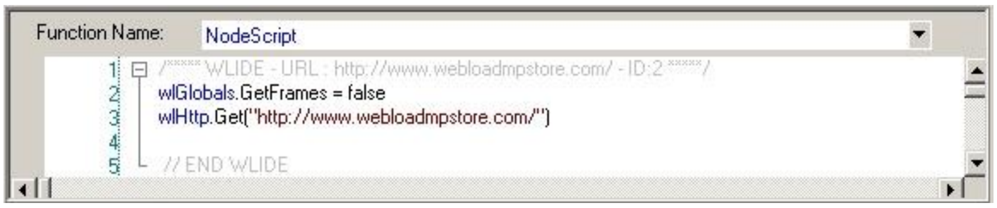
You can edit the JavaScript code generated by WebLOAD Recorder for any item in the script.

Note: When you select the Agenda root node, the entire script appears in the JavaScript View pane as read only. To edit the entire script, see *Using the JavaScript Editor* (on page 75).

To edit the JavaScript code for a script item:

1. In the main window, click **Open** in the **File** tab of the ribbon and open the script you want to edit.
2. Make sure that you are in Visual Editing mode.
3. Select the **JavaScript View** checkbox in the **View** tab to open the JavaScript View pane.
4. Select the item in the Script Tree.

   The JavaScript script code for that item appears in the JavaScript View pane.


Function Name: NodeScript

```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:2 *****/
wlGlobals.GetFrames = false
wlHttp.Get("http://www.webloadmpstore.com/")

// END WLIDE
```

Figure 50: JavaScript View Pane

5. Edit the script (see *Editing the JavaScript Code* on page 79).

Important: The ID number is automatically generated by WebLOAD Recorder and is the connection between the script node and the specific header. It is recommended that you do not change the contents of this comment. If you do, important data might be lost.

Editing the JavaScript Code Functions

A script includes a few sections of code, including functions. At the Agenda root node only, you can select these sections from the **Function Name** drop-down list.

When you select the `NodeScript` for the Agenda root node, the entire script appears in the JavaScript View pane as read only. You can only edit the script as a whole file when in JavaScript Editing mode (see *Using the JavaScript Editor* on page 75).

When you select a section other than `NodeScript` for the Agenda root node, the code appears in the JavaScript View pane. In the JavaScript View pane, you can edit the JavaScript code for functions called in the script. By default, WebLOAD Recorder calls the `InitAgenda(), InitClient(), TerminateClient(), and TerminateAgenda()` functions ?for each script.

Table 8: WebLOAD Functions

Function	Description
`InitAgenda`	Optional. Creates a JavaScript function `InitAgenda` to begin the script. `InitAgenda` is typically where global variables are defined.
`InitClient`	Optional. Creates a JavaScript function `InitClient` to begin a client process. Usually there will be only one client in a WebLOAD Recorder session; WebLOAD uses multiple clients.
`TerminateClient`	Optional. Creates a JavaScript function `TerminateClient` to end a client process. Usually there will be only one client in a WebLOAD Recorder session; WebLOAD uses multiple clients.
`TerminateAgenda`	Optional. Creates a JavaScript function `TerminateAgenda` to end the script.

The function properties do not need to be edited unless you want to make special customizations, such as including a function from a different file and using the `IncludeFile()` function.

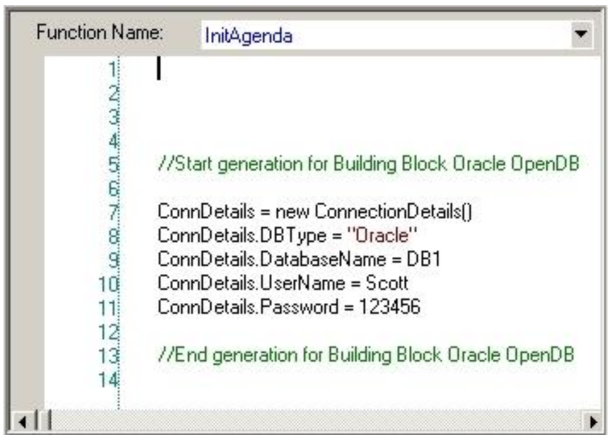
To edit the JavaScript code for functions:

1. In the main window, click **Open** in the **File** tab of the ribbon and open the script you want to edit.
2. Make sure that you are in Visual Editing mode.
3. Select the **JavaScript View** checkbox in the **View** tab to open the JavaScript View pane.
4. Select the script item in the Script Tree.

   The JavaScript script code for the script item appears in the JavaScript View pane. The JavaScript for the Agenda root node will include the whole script.

5. From the **Function Name** drop-down list, located at the top of the JavaScript View pane, select the name of the function.

   The JavaScript code for the function appears in the JavaScript View pane.



Function Name: InitAgenda

```
//Start generation for Building Block Oracle OpenDB

ConnDetails = new ConnectionDetails()
ConnDetails.DBType = "Oracle"
ConnDetails.DatabaseName = DB1
ConnDetails.UserName = Scott
ConnDetails.Password = 123456

//End generation for Building Block Oracle OpenDB
```

Figure 51: JavaScript View Pane

6. Type the JavaScript code to include in the `InitClient, InitAgenda, TerminateClient,` or `TerminateAgenda` (see *Editing the JavaScript Code* on page 79).

   For guidelines for replacing the placeholder variables with your own, see *Guidelines for Editing JavaScript Code* (on page 83).

Note: You cannot add a WebLOAD Recorder protocol block in the middle of a function. When in Visual Editing mode, this option is disabled.

Using the JavaScript Editor

Although represented visually, all scripts are written in JavaScript. The JavaScript code within a script is created from the actions you record and the verification tests you place in the script. You can add JavaScript Objects to your recorded script, allowing you to add additional written code directly to your script. The JavaScript Editor is both a viewer and an editor for adding and editing JavaScript code in the script.

WebLOAD Recorder provides the following features for manually editing a script:

- Import JavaScript Files

  WebLOAD Recorder enables you to import JavaScript files into your script.

- WebLOAD Recorder Protocol Block

  WebLOAD Recorder enables you to add code to your script which is then represented visually in the Script Tree.

- An IntelliSense Editor mode for the JavaScript View pane

  Add new lines of code to your script or edit existing JavaScript functions through the IntelliSense Editor mode of the JavaScript View pane. The IntelliSense Editor helps you write the JavaScript code for a new function by formatting new code and prompting with suggestions and descriptions of appropriate code choices and syntax as programs are being written. IntelliSense supports the following shortcut keys:

- **Period (".")** – Enter a period after the object name, to display a drop-down list of the object's available properties that can be added to the script (see Figure 52).
- **<CTRL> <Space>** – While typing the name of an object, you can type <CTRL> <Space> to display a drop-down list of the available objects that begin with the letters that you entered. For example, if you type `wl` the IntelliSense Editor displays a drop-down list of all of the objects that begin with `wl` (such as `wlhttp`).

In addition, the IntelliSense Editor gives a structure to the code with the outline bar and line numbering.

Collapsing the code enables you to view the heading of the section, without seeing the code within the section. To expand or collapse different sections of the code:

- Click the plus sign (+) or minus sign (-) on the outline bar,

  -Or-
- Right-click within the IntelliSense Editor and select **Outlining** from the pop-up menu. The available outlining options are:
  - **Toggle outline** – collapses or expands the section at the mouse location.
  - **Toggle all outline** – collapses or expands all outlines.
  - **Collapse to definition** – collapses all outlines.

You can enable or disable both the outline bar and line numbering features by:

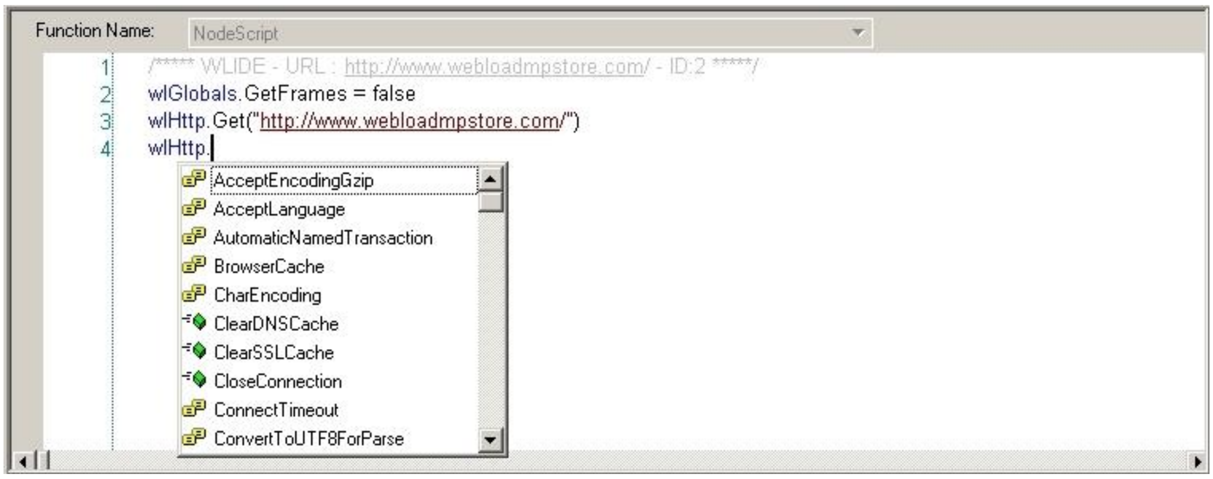
- Right-clicking within the IntelliSense Editor and selecting **Enable Outlining** or **Line Numbers** from the pop-up menu.

When these features are enabled, a checkmark appears next to the name in the pop-up menus. By default, these features are enabled, but WebLOAD opens with the settings that were saved during the previous WebLOAD session. During playback and debug modes, all outlines are expanded.

Use WebLOAD Recorder's predefined delimiters to keep your code structured and organized. The available delimiters include:

- For JavaScript functions, use "{" as the start delimiter and "}" as the end delimiter.
- For Script Tree nodes, insert a WLIDE comment from the General WebLOAD Recorder toolbox. This automatically inserts a start delimiter "//" and end delimiter "End WLIDE".

For more information, see the *WebLOAD Scripting Guide.*


Function Name: NodeScript

```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:2 *****/
wlGlobals.GetFrames = false
wlHttp.Get("http://www.webloadmpstore.com/")
wlHttp.
```

Figure 52: IntelliSense Editor Mode for JavaScript View Pane

- A selection of the most commonly used functions and commands, available through the **Insert** menu.

  You can choose to program your own JavaScript Object code within your script and take advantage of the WebLOAD Recorder GUI to simplify your programming efforts. Rather than manually typing out the code for each command, with the risk of making a mistake, even a trivial typographical error, and adding invalid code to the script file, you may select an item from the **Insert** menu, illustrated in the following figure, to bring up a list of available commands and functions for the selected item. WebLOAD Recorder automatically inserts the correct code for the selected item into the JavaScript Object currently being edited. You may then change specific parameter values without any worries about accidental mistakes in the function syntax.

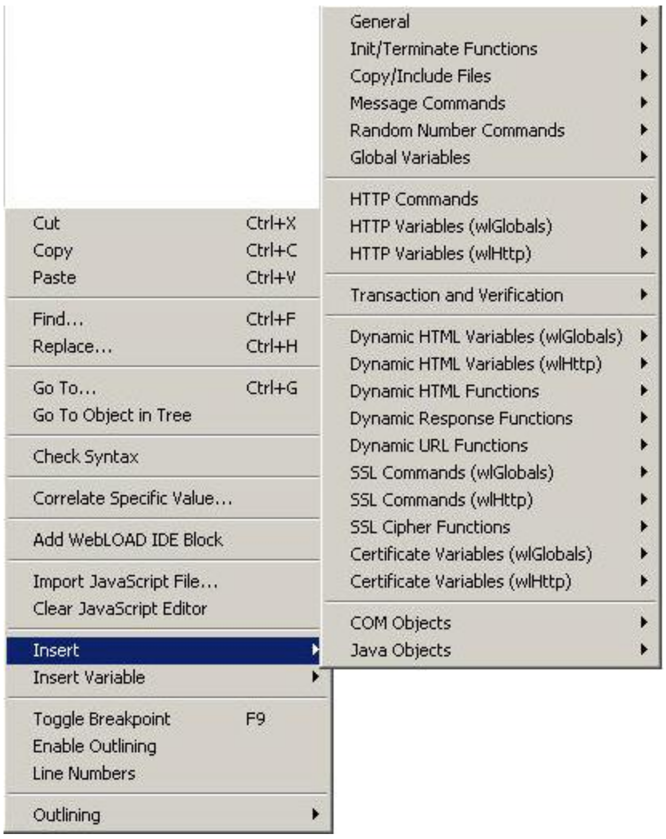


Figure 53: Insert Menu

In addition to the Insert menu, you may select an item from the Insert Variable menu, to add system and user-defined parameters to the script. This eliminates the need for manual coding. For more information about adding user-defined parameters to the script, see *Inserting User-Defined Parameters in a Script* (on page 228).

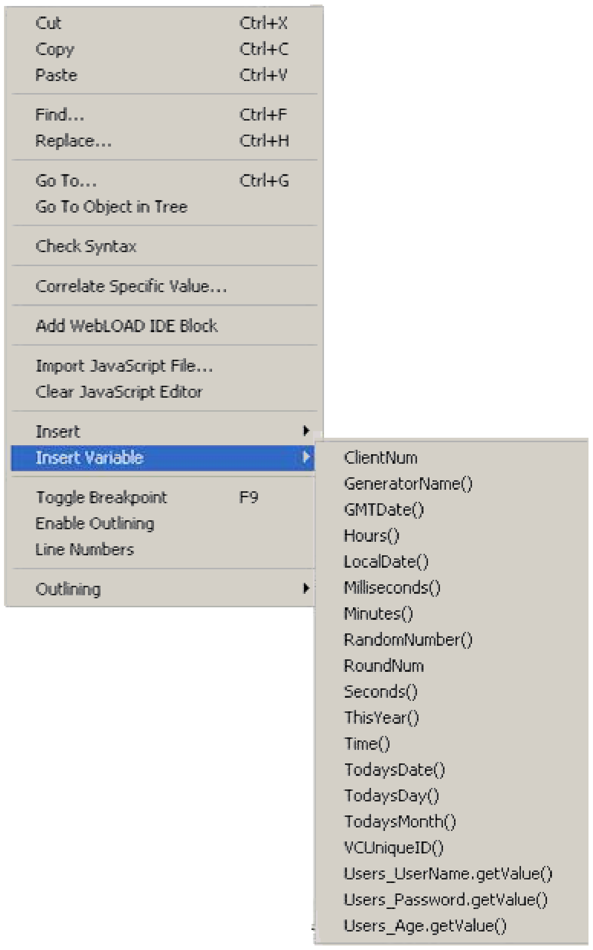


Figure 54: Insert Variable Menu

- A Syntax Checker that checks the syntax of the code in your script file and catches simple syntax errors before you spend any time running a test session. While standing in the JavaScript View pane of the WebLOAD Recorder desktop, click **Syntax Checker** in the **Edit** tab of the ribbon, or right-click and select **Check Syntax** from the pop-up menu to check the syntax of the code in your script file.

Important: WebLOAD Recorder scripts should be edited only within the confines of WebLOAD Recorder, not within an external editor. If you use an external editor to modify the JavaScript code in a script file generated by WebLOAD Recorder, your visual script will be lost.

Editing the JavaScript Code

Note: Any part of the code that is edited in the JavaScript Editing mode is inserted into the script as a JavaScript block, which cannot be edited in the Visual Editing mode.

To edit the JavaScript code for the script:

1. In the main window, click **Open** in the **File** tab of the ribbon and open the script you want to edit.

2. Select **Full Script** in the **Home** tab of the ribbon to open the script in JavaScript Editing mode.

   The entire script appears.

3. Position the cursor where you want to edit the JavaScript code.

Note: To add a new JavaScript node, place the cursor after the END WLIDE comment of the previous node before you start writing your JavaScript code. When you switch back to Visual Editing mode a JavaScript node is automatically created, containing your code.

4. Type the JavaScript code that you want this item to contain.
5. Add functions and commands from the **Insert** menu (see *Adding Commands and Functions to a Script* on page 82).
6. Import a JavaScript file:
   a. Right-click in the script.
   b. Click **Import JavaScript File** from the pop-up menu.

      The JavaScript code is added to the script.
7. Add a WebLOAD Recorder protocol block from the pop-up menu (see *Adding WebLOAD Recorder Protocol Blocks* on page 80).
8. Perform a syntax check:
   a. Right-click in the script.
   b. Select **Check Syntax** from the pop-up menu.

      WebLOAD Recorder performs a syntax check and displays the errors.
9. Toggle a breakpoint (for more information, see *Setting Breakpoints* on page 121).

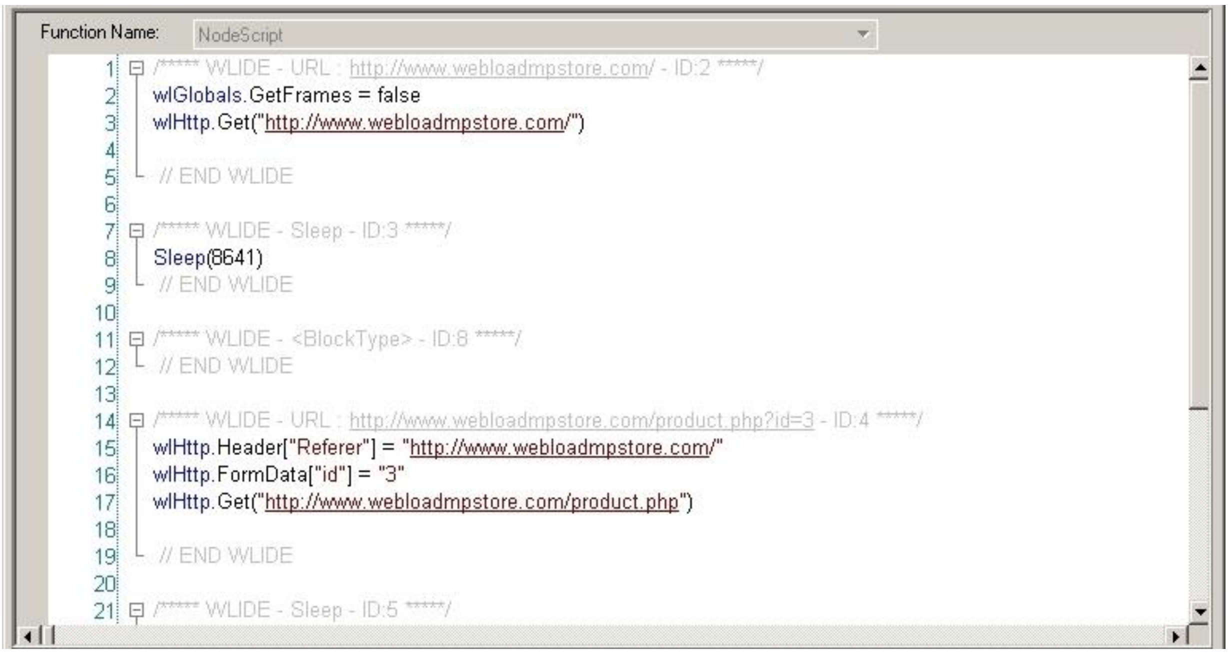
Note: To clear the JavaScript View pane, click **Clear JavaScript Editor** in the **Edit** tab of the ribbon.

Adding WebLOAD Recorder Protocol Blocks

To add WebLOAD Recorder Protocol Blocks to a script:

1. In the main window, click **Open** in the **File** tab of the ribbon and open the script you want to edit.
2. Click **Full Script** in the **Home** tab to edit the script in full screen editing mode.
3. In the JavaScript View pane, position the cursor where you want to place the WebLOAD Recorder protocol block.
4. Right-click in the script, and click **Add WebLOAD Recorder Block** from the pop-up menu.

A WebLOAD Recorder protocol block header is inserted on a new line in the script, immediately after the line where the cursor is located, and a script item is added to the Script Tree.


Function Name: NodeScript

```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:2 *****/
wlGlobals.GetFrames = false
wlHttp.Get("http://www.webloadmpstore.com/")

 // END WLIDE

/***** WLIDE - Sleep - ID:3 *****/
Sleep(8641)
 // END WLIDE

/***** WLIDE - <BlockType> - ID:8 *****/
 // END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/product.php?id=3 - ID:4 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/"
wlHttp.FormData["id"] = "3"
wlHttp.Get("http://www.webloadmpstore.com/product.php")

 // END WLIDE

/***** WLIDE - Sleep - ID:5 *****/
```

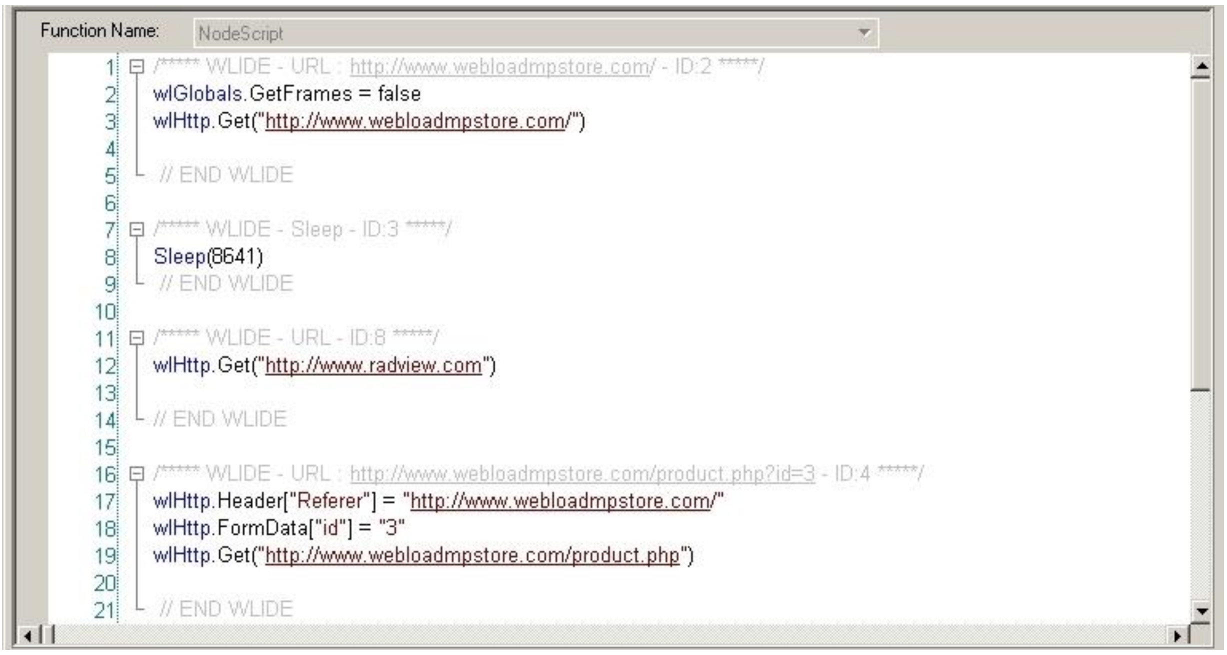
Figure 55: WebLOAD Recorder Protocol Block Header

5. Replace the placeholder `<Block Type>` with a description.

   For example: Replace `<Block Type>` with `URL`.

6. Add the JavaScript code after the WebLOAD Recorder protocol block header.

   The code is added to the script.


Function Name: NodeScript

```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:2 *****/
wlGlobals.GetFrames = false
wlHttp.Get("http://www.webloadmpstore.com/")

 // END WLIDE

/***** WLIDE - Sleep - ID:3 *****/
Sleep(8641)
 // END WLIDE

/***** WLIDE - URL - ID:8 *****/
wlHttp.Get("http://www.radview.com")

// END WLIDE

/***** WLIDE - URL : http://www.webloadmpstore.com/product.php?id=3 - ID:4 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/"
wlHttp.FormData["id"] = "3"
wlHttp.Get("http://www.webloadmpstore.com/product.php")

 // END WLIDE
```

Figure 56: JavaScript Code added to WebLOAD Recorder Protocol Block Header

Adding Commands and Functions to a Script

WebLOAD Recorder contains shortcuts to frequently performed actions. This section describes how to place Commands, and functions from the **Insert** menu in a script. For guidelines for replacing the placeholder variables with your own, see *Guidelines for Editing JavaScript Code* (on page 83).

To add commands and functions to a script:

1. In the main window, click **Open** in the **File** tab and open the script you want to edit.
2. In the JavaScript View pane, position the cursor where you want to place the command or function.
3. Right-click in the script and click **Insert**.

   The list of shortcuts appears.

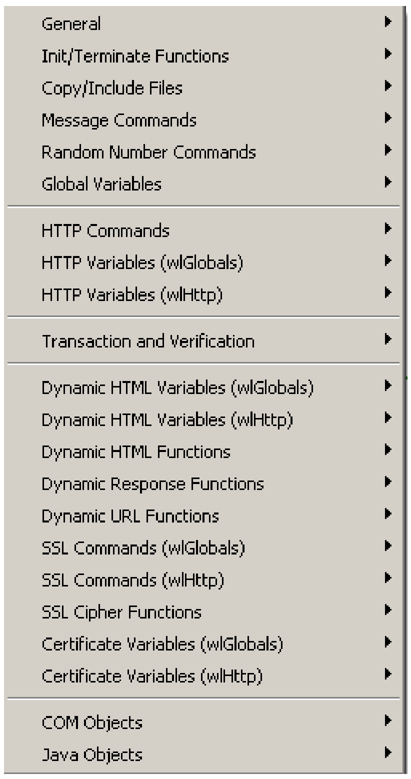


Figure 57: Shortcuts List

4. Select a command or function.

   The command or function selected is inserted on a new line in the script, immediately after the line where the cursor is located.

Guidelines for Editing JavaScript Code

Use the following guidelines to edit commands and functions you have placed in a script through the JavaScript Editor:

- Placeholders between brackets < > that appear in generic examples *must* be replaced with the literal name of a variable.

  For example, the generic example:

  ```
  wlHttp.PassWord = "<Password>"
  ```

  must be replaced with the string:

  ```
  wlHttp.PassWord = "Blue"
  ```

- Placeholders between square brackets within parentheses ([ ]) are optional function parameters. It is not mandatory to include them in the command.

For example, the generic example:

```
<Line_Array> = GetLine("<File_Name>" [,"<Separator>"])
```

can be replaced with the string:

```
MyFile = GetLine("C:\\InputFile.txt")
```

- Placeholders between square brackets [ ] are array variables and *must* be replaced with the literal name of a variable, enclosed with square brackets.

  For example:

  ```
  wlHttp.Header["<Key>"]= "<Value>"
  ```

  must be replaced with the string:

  ```
  wlHttp.Header["proxy-connection"]="Keep-Alive"
  ```

- In a WebLOAD Recorder protocol block, replace the placeholder `<Block Type>` with a description.

  For example:

  ```
  <Block Type>
  ```

  can be replaced with:

  ```
  SSL Certificate
  ```

See the *WebLOAD Scripting Guide* for more information.

Editing your Script Using the WebLOAD Recorder Toolbox Set

The WebLOAD Recorder provides a set of objects, such as Sleep, that you can drag and drop to add script items in the Script Tree while recording or viewing your script. The WebLOAD Recorder bar is referred to as the toolbox.

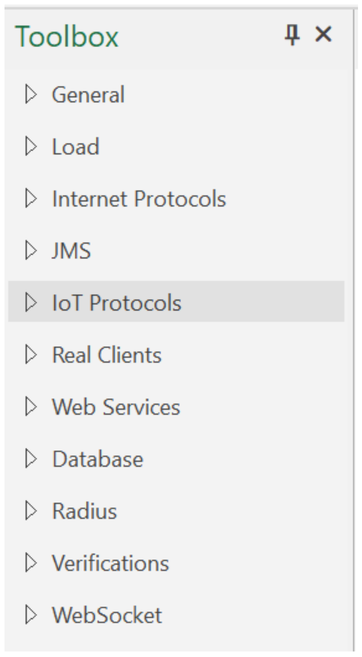


Figure 58: WebLOAD Recorder Toolbox

Use the WebLOAD Recorder toolboxes to add the following items to your script:

- General objects, such as Message or Sleep timers. These objects are used in all test scripts, run in both WebLOAD Recorder and WebLOAD. General toolbox tools are described in *The WebLOAD Recorder General Toolbox* (on page 233).
- Load objects, such as transactions and synchronization points used in WebLOAD tests. Load toolbox tools are described in *The WebLOAD Recorder Load Toolbox* (on page 237).
- Internet Protocols functionality, such as downloading data from an FTP site for a WebLOAD Recorder test. Internet Protocols Building Blocks are described in *The WebLOAD Recorder Internet Protocols Toolbox* (on page 247).
- JMS functionality, such as sending and receiving JMS message. JMS Building Blocks are described in *The JMS Toolbox* (on page 354).
- IoT Protocols functionality, such as connecting to a broker. IoT Protocols Building Blocks are described in *The WebLOAD Recorder IoT Protocols Toolbox* (on page 300).
- Real Clients functionality, such as Selenium actions and a Perfecto Mobile script. Real Clients building blocks are described in *The Real Clients Toolbox* (on page 361).

- Web Service functionality, such as adding web service calls or HTTP requests to your test session script. Web Services Building Blocks are described in *The WebLOAD Recorder Web Services Toolbox* (on page 348).
- Database actions, such as opening and getting data from a database for a WebLOAD Recorder test. Database Building Blocks are described in *The WebLOAD Recorder Database Toolbox* (on page 309).
- Verification functionality, such as verifying specific elements within HTTP responses in your script. Verifications Building Blocks are described in *The WebLOAD Recorder Verifications Toolbox* (on page 337).
- WebSocket functionality, such as creating a WebSocket connection to a specific URL address. WebSocket building blocks are described in *The WebLOAD Recorder WebSocket Toolbox* (on page 345).

Adding Script Items from a WebLOAD Recorder Toolbox

To drag and drop a WebLOAD Recorder toolbox item into your script:

1. Place the mouse over the item in the WebLOAD Recorder toolbox that you want to add.
2. Press and hold the mouse button (just “clicking” has no effect).
3. Drag the item into the Script Tree, highlighting the item *after* which you want to add the new item.
4. Release the script item you have inserted.
5. For many of the items, such as Message, Comments, and Sleep objects, additional dialog boxes are used to prompt you for the information necessary to add messages, comments, and pause times. Enter the necessary information, and click **OK**.

   The item with its toolbox icon appears in the Script Tree at the point where you placed the item.

6. For JavaScript Objects, add JavaScript code to the script (see *Using the JavaScript Editor* on page 75).

Working with JavaScript Files

WebLOAD Recorder enables you to open a JavaScript file and convert it to a WebLOAD Recorder project file or continue working with the file as a JavaScript file.

You may want to save it as a JavaScript file if it is an Include file (component of a whole script) and not the main script.

We recommend that you convert the JavaScript file to a WebLOAD Recorder project file for the following reasons:

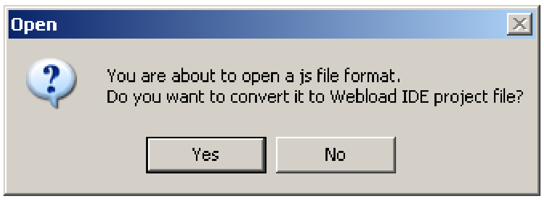
- The project file is better suited to the WebLOAD Recorder visual environment.
- Enables you to save additional information to the script, such as the Current Project options.

Note: When you convert a JavaScript file to a WebLOAD Recorder project file, the original JavaScript file is not deleted. If you convert it to the new format, you can always save it as a regular JavaScript file, using the Save As option.

To work with a JavaScript File:

1. In the main window, click **Open** in the **File** tab.
2. Select a JavaScript file.

   The Open message appears.


Open

You are about to open a js file format.
Do you want to convert it to Webload IDE project file?

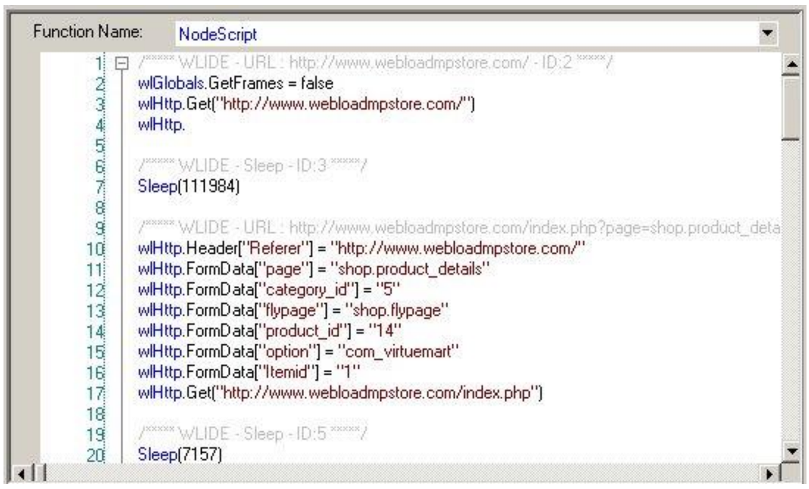
Figure 59: Open Message Box

3. Click **Yes** to convert the JavaScript file to a WebLOAD Recorder project file,

   -Or-

   Click **No** to continue working with the file as a JavaScript file.

   If you continue working with the file as a JavaScript file, the file appears in the JavaScript View pane as a JavaScript file, and the WebLOAD Recorder block shows that it is a JavaScript file.


Function Name: NodeScript

```
/***** WLIDE - URL : http://www.webloadmpstore.com/ - ID:2 *****/
wlGlobals.GetFrames = false
wlHttp.Get("http://www.webloadmpstore.com/")
wlHttp.

/***** WLIDE - Sleep - ID:3 *****/
Sleep(111984)

/***** WLIDE - URL : http://www.webloadmpstore.com/index.php?page=shop.product_deta
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/"
wlHttp.FormData["page"] = "shop.product_details"
wlHttp.FormData["category_id"] = "5"
wlHttp.FormData["flypage"] = "shop.flypage"
wlHttp.FormData["product_id"] = "14"
wlHttp.FormData["option"] = "com_virtuemart"
wlHttp.FormData["Itemid"] = "1"
wlHttp.Get("http://www.webloadmpstore.com/index.php")

/***** WLIDE - Sleep - ID:5 *****/
Sleep(7157)
```

Figure 60: JavaScript File in JavaScript View Pane

Important: If you save the file as a JavaScript file, the next time you open the file, the **Open** message will *not* appear.

Chapter 7

Correlating Scripts

This section provides instructions for correlating scripts with WebLOAD Recorder. The WebLOAD correlation engine helps you overcome one of the main challenges of recording or replaying Web application load tests: dynamic data.

Dynamically generated data changes every time you run a Web application. For example, the session ID that uniquely identifies a user's active session is allocated by the Web server or the application every time such a session is initiated. (The session ID is also used for session management. For more information, see *Session Management* on page 107.) Such dynamic data cannot simply be recorded as is and played back, because the playback will inevitably fail.

WebLOAD enables you to correlate the most common methods used to pass dynamic data between a server and a client. The methods are:

- **Cookies** – This method is mostly used for session management, but cookies may also contain additional dynamic data sent from a server. In most situations, the browser returns the sent cookie in subsequent requests. This scenario is handled automatically by WebLOAD and no additional correlation activities are required. WebLOAD also supports cases where a value received in a cookie is sent as request data or a cookie is created in the client-side JavaScript.
- **URL rewriting** – This method is most commonly resolved by assigning the returned Session ID to a local variable and using this variable throughout the rest of the script.
- **Hidden form fields** – This method is most commonly used for passing localized dynamic data that is not necessarily within the scope of the full session.
- **Any response content** – Some applications return dynamic data within various types of HTTP responses, including client-side JavaScript and XML content.

About Correlating Scripts with WebLOAD Recorder

WebLOAD contains a powerful rule based correlation engine. You can define rules that describe how dynamic values should be extracted in their application.

WebLOAD also provides automatic discovery of potential correlation rules. Using auto-discovery of rules eliminates the need to manually define correlation rules in some common cases.

WebLOAD Recorder identifies dynamic data using correlation rules. These rules can be configured to suit your correlation needs. For more information on correlation rules, see *Configuring the Correlation Rules* (on page 98).

The WebLOAD correlation engine enables you to:

- **Automatically discover potential correlation rules** – WebLOAD can automatically suggest correlation rules for common scenarios, based on the values sent and received in the script, or for a specific value.
- **Reuse correlation logic** – Once the rules are defined or discovered, the correlation engine uses the rules to make all necessary changes to the script. The rules can be used unchanged in all future scripts with the same scenarios.
- **Re-run correlation at any time** – You can run the correlation process multiple times on previously recorded scripts. For example, if you perform correlation based on a particular set of rules, you can correlate that script another time based on a different set of rules, without re-recording the script.
- **Add comments to your JavaScript** – When dynamic data is correlated according to the correlation rules you define, WebLOAD Recorder can add comments to your JavaScript to help you keep track of the changes made by the correlation engine.
- **Keep a detailed correlation log** – The correlation engine can keep track of all the correlation operations performed on your script in the form of a textual log file. You can determine the location of this log file as well as the level of information it stores. Either you or RadView Technical Support can use the correlation log file to identify, investigate, and solve correlation problems.

Correlating To and From Cookies

WebLOAD enables correlating to and from cookies.

Most cookies get their values from a previous response's Set-Cookie header, originating from the server. These values are handled automatically by WebLOAD and do not require any action. They do not appear in the script.

Client-side cookies are created by the client using JavaScript in response to a user preference, client-side generated content, such as a random number, or for some other reason. These cookies appear in the script as `wlCookie.Set()` commands. The values of these cookies can be correlated like any other dynamic data.

Performing Correlation

Correlation is performed on your script, based on the correlation rules. Correlation rules are defined in one of the following ways:

- **Auto-discovery of correlation rules** – When performing correlation with auto-discovery of correlation rules, the correlation engine compiles a list of the suggested correlation rules, enabling you to select the rules that are applicable to your application. For more information, see *Approving the Correlation Engine Rules* (on page 94).
- **In the Correlation Rules Editor** – Before running the correlation engine, you can use the correlation rules editor to add and edit the rules.

Note: You can turn discovered rules into permanent rules and then edit the rules in the correlation rules editor.

For more information, see *Configuring the Correlation Rules* (on page 98).

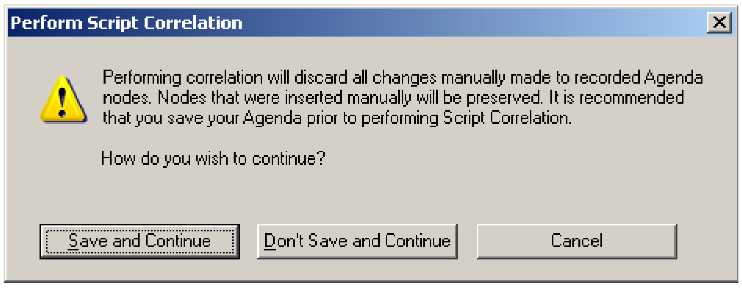
Note: Correlation cannot be performed on scripts that were not recorded.

Performing Auto-discovery Correlation

To perform correlation with auto-discovery of rules:

1. Click **Correlation** in the **Home** tab of the ribbon and select **Correlate Script and Discover Rules** from the drop-down list.

   The Perform Script Correlation dialog box appears.



Perform Script Correlation

Performing correlation will discard all changes manually made to recorded Agenda nodes. Nodes that were inserted manually will be preserved. It is recommended that you save your Agenda prior to performing Script Correlation.

How do you wish to continue?

Figure 61: Perform Script Correlation Dialog Box

2. Click **Save and Continue** to save the changes in your script and perform correlation.

   -Or-

Click **Don't Save and Continue** to perform correlation without saving the changes in your script.

-Or-

Click **Cancel** to close the Perform Script Correlation dialog box without performing correlation in your script.

Performing Auto-discovery Correlation for Specific Values

WebLOAD enables you to perform correlation with auto-discovery of rules for any value you select in a script. This provides correlation when normal auto-discovery is not sufficient. Auto-discovery correlation for specific values can be used instead of normal auto-discovery in the following cases:

- **Partial values** – Since auto-discovery only searches for exact matches, if the dynamic value you wish to correlate is part of the sent value, it is not found. For example, in `wlHttp.FormData["data"]="session*1234"` or `wlHttp.FormData["data"]="<xml><data sessionid= 1234/><xml>"`, if only `1234` is dynamic, normal auto-discovery does not find this value.

Note: The only exception is in POST and GET commands, where the parameter name or values are replaced.

- **Dynamic URLs** – Similar to partial values, if a URL contains a dynamic value, it is not found. For example, in `http://www.mydomain.com/2131231/something.asp`, if only 2131231 is dynamic, normal auto-discovery does not find this value..
- **Non-standard query strings** – Similar to partial values and dynamic URLs, if a value appears to be a URL but uses a different encoding method, it is not found. For example, in `http://www.domain.com;strange=4342?normal=44&other=222`, normal auto-discovery finds normal=44 and other=222, but does not find strange=4342.
- **Using the referer header** – When available, auto-discovery uses the referer header and only searches for values there. Values that need to be correlated may be elsewhere.
- **Filter noise** – By default, auto-discovery filters out values that are too short or have a low score. In some cases, required rules may also be filtered out.

To perform correlation with auto-discovery of rules for a specific value:

1. Select the value you wish to correlate in the script.
2. Click **Correlation** in the **Home** tab of the ribbon and select **Correlation Script for Specific Value** from the drop-down list.

   -Or-

Right-click and select **Correlate Specific Value**.

The Correlation Specific Value dialog box appears, displaying the selected value.

3. Click **OK**.

   WebLOAD performs a regular correlation with auto-discovery of rules. The Perform Script Correlation dialog box appears (Figure 61).

4. Click **Save and Continue** to save the changes in your script and perform correlation.

   -Or-

   Click **Don't Save and Continue** to perform correlation without saving the changes in your script.

   -Or-

   Click **Cancel** to close the Perform Script Correlation dialog box without performing correlation in your script.

Setting the Default Correlation Action

You can control the default correlation action that WebLOAD should perform after recording.

To control which correlation action is performed after recording:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (Figure 114).

2. Select the **Correlation Options** tab.

   The Correlation Options tab moves to the front of the dialog box (Figure 120).

3. In the Correlation level drop-down, select one of the following:
   - **Do not run** – When recording is complete, go directly to the script without performing correlation. You can run the correlation engine at a later time.
   - **Use existing rules** – Run the correlation engine once the script recording is complete, only using the existing rules (do not try to auto-discover new rules).
   - **Discover rules** – Run the correlation engine using existing rules and try to discover new rules.
   - **Prompt** – A dialog that provides you with all of the options is displayed when the recording is complete. This is the default setting. For more information on using the Recording Complete dialog box, see *Recording a script* (on page 34).

Automatic Discovery of Correlation Rules

WebLOAD enables you to automatically discover potential correlation rules.

The discovery process is based on reverse scanning of the script. The auto-discovery module searches for the sent values in previous responses and tries to formulate rules to extract the values.

Notes:

- Not all of the suggested rules are usually needed. Select the rules that are appropriate for your application. You can run auto-discovery as many times as needed, and try different rules.
- There are some rule types that are not automatically discovered and you must manually define a rule for them.

When running correlation with auto-discovery, the correlation engine uses the existing defined rules and does not discover them again. Inactive rules are also not rediscovered or used. Making a rule inactive can be used to prevent discovering rules that you already know are unneeded.

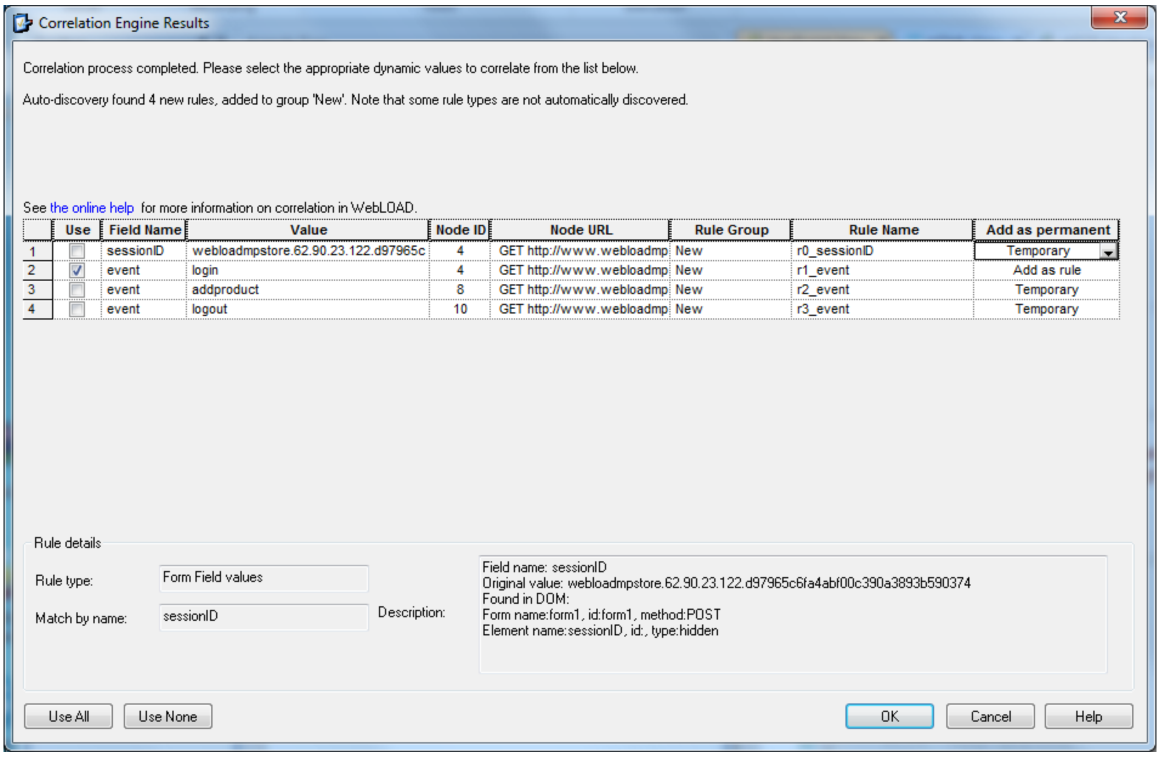
When the correlation process is complete, a review form is displayed for the user to choose which rules to use. For more information see *Approving the Correlation Engine Rules* on page 94.

Approving the Correlation Engine Rules

When performing Auto-discovery correlation, the correlation engine compiles a list of the suggested correlation rules according to the dynamic values that were recorded in the script. This enables you to determine which rules to approve and use during the correlation.

To approve the correlation engine rules:

1. After running correlation with auto-discovery of rules, the Correlation engine results dialog box appears.



Correlation process completed. Please select the appropriate dynamic values to correlate from the list below.

Auto-discovery found 4 new rules, added to group 'New'. Note that some rule types are not automatically discovered.

See the online help for more information on correlation in WebLOAD.

	Use	Field Name	Value	Node ID	Node URL	Rule Group	Rule Name	Add as permanent
1	☐	sessionID	webloadmpstore.62.90.23.122.d97965c	4	GET http://www.webloadmp	New	r0_sessionID	Temporary
2	✓	event	login	4	GET http://www.webloadmp	New	r1_event	Add as rule
3	☐	event	addproduct	8	GET http://www.webloadmp	New	r2_event	Temporary
4	☐	event	logout	10	GET http://www.webloadmp	New	r3_event	Temporary

Rule details

Rule type: Form Field values

Match by name: sessionID

Description:
Field name: sessionID
Original value: webloadmpstore.62.90.23.122.d97965c6fa4abf00c390a3893b590374
Found in DOM:
Form name:form1, id:form1, method:POST
Element name:sessionID, id:, type:hidden

Figure 62: Correlation Engine Results Dialog Box

2. Edit the rules in the Correlation Engine Results dialog box according to the following table and click **OK**.

Table 9: Correlation Engine Results Dialog Box Options

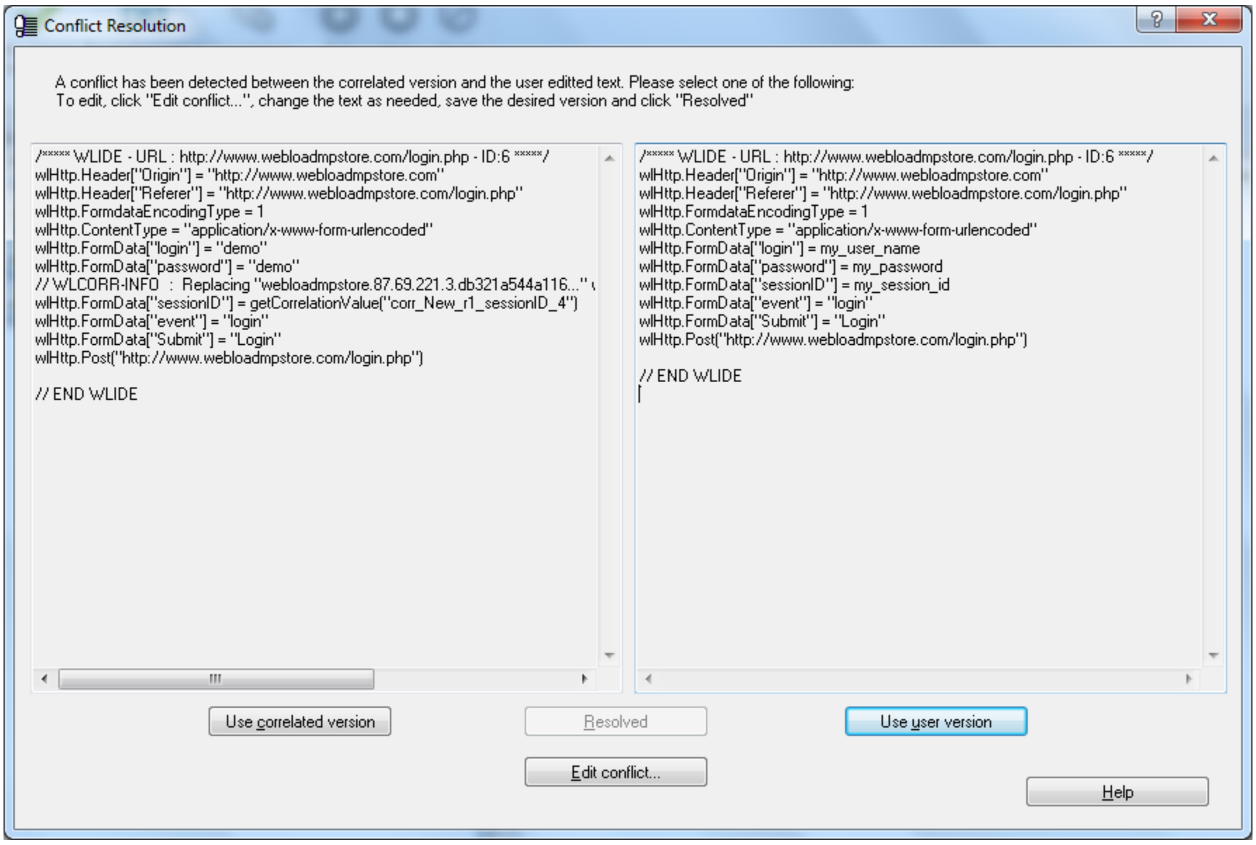
Column / Field	Description
Use	Select the rules to use in this script. You can click **Use All** to select all of the rules in the **Use** and **Add as permanent** columns, or **Use None** to deselect all of the rules in both columns. **Note:** You can use the rule in the current script, without adding it as a permanent rule.
Field Name	The field name that was used to send the value in the request. This field may be empty if it is defined in the rule.
Value	The value extracted or replaced.
Node ID	The node ID in the script. Each script node is marked with a comment indicating the node ID, for example: `/***** WLIDE - URL : http://ww.mydomain.com/ - ID:42 *****/`
Node URL	The GET or POST request of the value extracted or replaced.
Rule Group	The name of the group to which the correlation rule belongs.
Rule Name	The name of the correlation rule.

Column / Field	Description
Add as permanent	Specify how to use this rule: • Never use – Do not use this rule in any script • Add as rule – Add the rule to the permanent rules (always use) • Temporary – Use the rule only in this run
Rule details	
Rule Type	The method used to find the dynamic data to be correlated, according to the selected rule's definition. To modify the rule, see *Defining Correlation Rules* on page 101. Possible values are: • All body text • Form field values • User defined • Replace with expression • Search in cookies
…	Additional rule type fields. These fields change according to the rule type.
Description	A summary of the selected rule's details.

Resolving Conflicts between Manual Changes and Correlation Changes

Starting from WebLOAD 10.1, when you run correlation all the manual changes you may have made in the original JavaScript code are preserved by default (see *Configuring the Correlation Options* on page 189). However, the process of correlation also introduces some changes into the original JavaScript. Sometimes your manual changes conflict with the correlation changes. When this happens, a Conflict Resolution window appears in which you are asked to resolve the conflict.

Figure 63 shows a sample Conflict Resolution window.


Conflict Resolution

A conflict has been detected between the correlated version and the user editted text. Please select one of the following:
To edit, click "Edit conflict...", change the text as needed, save the desired version and click "Resolved"

```
/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:6 *****/
wlHttp.Header["Origin"] = "http://www.webloadmpstore.com"
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/login.php"
wlHttp.FormdataEncodingType = 1
wlHttp.ContentType = "application/x-www-form-urlencoded"
wlHttp.FormData["login"] = "demo"
wlHttp.FormData["password"] = "demo"
// WLCORR-INFO  :  Replacing "webloadmpstore.87.69.221.3.db321a544a116..." 
wlHttp.FormData["sessionID"] = getCorrelationValue("corr_New_r1_sessionID_4")
wlHttp.FormData["event"] = "login"
wlHttp.FormData["Submit"] = "Login"
wlHttp.Post("http://www.webloadmpstore.com/login.php")

// END WLIDE
```

```
/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:6 *****/
wlHttp.Header["Origin"] = "http://www.webloadmpstore.com"
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/login.php"
wlHttp.FormdataEncodingType = 1
wlHttp.ContentType = "application/x-www-form-urlencoded"
wlHttp.FormData["login"] = my_user_name
wlHttp.FormData["password"] = my_password
wlHttp.FormData["sessionID"] = my_session_id
wlHttp.FormData["event"] = "login"
wlHttp.FormData["Submit"] = "Login"
wlHttp.Post("http://www.webloadmpstore.com/login.php")

// END WLIDE
```

Figure 63: Conflict Resolution Window

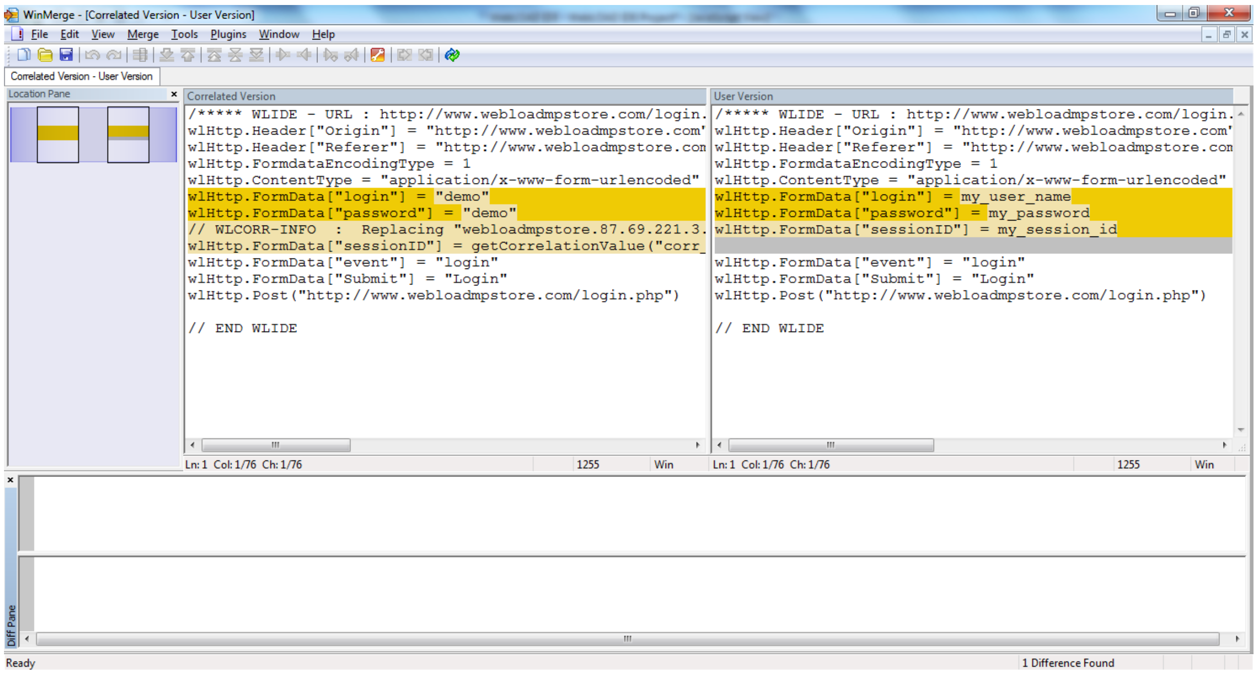
The left side of the Conflict Resolution window displays the correlated version without any user changes, and the right side displays the user version without any correlation changes. You must do one of the following:

- Click **Use correlated version** – This keeps all correlation changes and discards all user changes.
- Click **Use user version** – This keeps all user changes and discards all correlation changes.
- Click **Edit conflict** – This enables editing the JavaScript to your satisfaction. When you select this option, a WinMerge window appears by default. For information, refer to *Editing Conflicts between Manual Changes and Correlation Changes* below. When you finish editing, click **Resolved** in the Conflict Resolution window.

Editing Conflicts between Manual Changes and Correlation Changes

When you select to edit conflicts between manual changes and correlation changes, a merge tool is automatically launched, displaying the two conflicting versions.

The default merge tool is the WinMerge application. Note that you can optionally specify a different merge tool, as described in *Defining the Merge Tool Application* (on page 212).


WinMerge - [Correlated Version - User Version]
File Edit View Merge Tools Plugins Window Help
Correlated Version - User Version
Location Pane
Correlated Version
User Version
Ln: 1 Col: 1/76 Ch: 1/76
1255
Win
Diff Pane
Ready
1 Difference Found

Correlated Version:

```
/***** WLIDE - URL : http://www.webloadmpstore.com/login.
wlHttp.Header["Origin"] = "http://www.webloadmpstore.com'
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com
wlHttp.FormdataEncodingType = 1
wlHttp.ContentType = "application/x-www-form-urlencoded"
wlHttp.FormData["login"] = "demo"
wlHttp.FormData["password"] = "demo"
// WLCORR-INFO  :  Replacing "webloadmpstore.87.69.221.3.
wlHttp.FormData["sessionID"] = getCorrelationValue("corr_
wlHttp.FormData["event"] = "login"
wlHttp.FormData["Submit"] = "Login"
wlHttp.Post("http://www.webloadmpstore.com/login.php")

// END WLIDE
```

User Version:

```
/***** WLIDE - URL : http://www.webloadmpstore.com/login.
wlHttp.Header["Origin"] = "http://www.webloadmpstore.com'
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com
wlHttp.FormdataEncodingType = 1
wlHttp.ContentType = "application/x-www-form-urlencoded"
wlHttp.FormData["login"] = my_user_name
wlHttp.FormData["password"] = my_password
wlHttp.FormData["sessionID"] = my_session_id

wlHttp.FormData["event"] = "login"
wlHttp.FormData["Submit"] = "Login"
wlHttp.Post("http://www.webloadmpstore.com/login.php")

// END WLIDE
```

Figure 64: WinMerge Conflict Resolution Window

1. Select the lines you wish to edit, and edit them as desired.
2. Save your changes.
3. Exit the WinMerge application.

Configuring the Correlation Rules

WebLOAD Recorder enables you to configure the correlation rules used to define the correlation actions in your script with the Correlation Rules Editor. You can modify, create, and rename the correlation rules and groups.

Opening the Correlation Rules Editor

Open the Correlation Rules Editor to view, create or edit correlation rules.

To open the Correlation Rules Editor:

- Click **Correlation** in the **Home** tab of the ribbon and select **Correlation Rules Editor** from the drop-down list.

  The Correlation Rules Editor opens, displaying application- and development framework-specific correlation rules by groups.

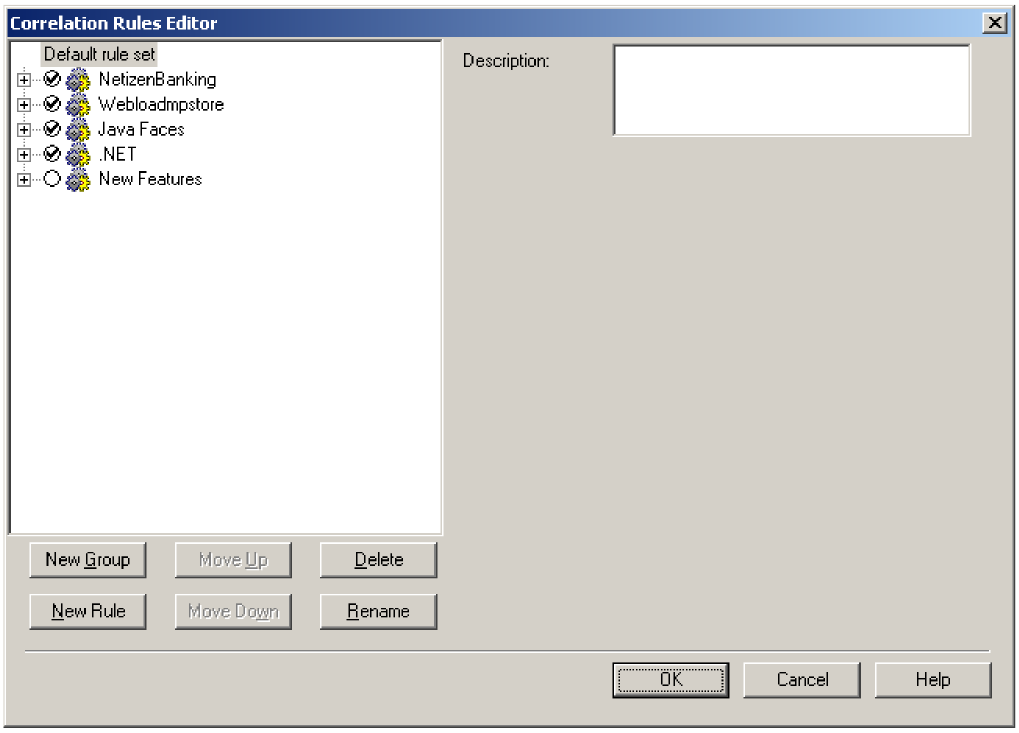


Figure 65: Correlation Rules Editor

The following table describes the options in the Correlation Rules Editor.

Table 10: Correlation Rules Editor Options

Field	Description
Default rule set	Displays a tree of the correlation rules. Each node represents a correlation rule group. You can expand the group nodes to view the associated correlation rules. The order of the correlation rules in the tree determines the order of their execution. You can configure the correlation rules by selecting the checkbox adjacent to the rule that you want to apply in the correlation. You can expand or compress the tree using the **+/-** buttons. If an upper level component is selected, all of the subcomponents in that tree will be selected. If only some subcomponents in a tree are selected, the upper level component is selected and greyed.
Description	Displays a description of the correlation group or rule. The type of information displayed in this area depends on the node selected in the Default rule set area.
New Group	Create a new correlation rule group below the selected group. If no group is selected, the new correlation rule group is created after the last group node.
New Rule	Create a new correlation rule below the selected rule. If no rule is selected, the new correlation rule is created after the last rule in the selected group.

Field	Description
Move Up	Move the selected correlation rule up inside its group or move the selected correlation group up in the tree.
Move Down	Move the selected correlation rule down inside its group or move the selected correlation group down in the tree.
Delete	Delete the selected correlation rule or group.
Rename	Rename the selected correlation rule or group.
OK	Accept your changes and close the Correlation Rules Editor dialog box.
Cancel	Discard your changes and close the Correlation Rules Editor dialog box.

Creating Correlation Rules

You can create correlation rules and groups to better suit your correlation requirements.

To create a correlation rule:

1. Click **Correlation** in the **Home** tab of the ribbon and select **Correlation Rules Editor** from the drop-down list.

   The Correlation Rules Editor dialog box opens (see Figure 65).

2. In the Default rule set area, select the correlation rule under which you wish to create your correlation rule and click **New Rule**,

   -Or-

   Right-click the correlation rule under which you wish to create your correlation rule and select **New Rule** from the pop-up menu.

   A new rule is created in the tree, at the specified location.

3. Modify the correlation rule parameters, as described in *Defining Correlation Rules* (on page 101).
4. Click **OK**. The new correlation rule is added to the Default rule set.

To create a correlation group:

1. Click **Correlation** in the **Home** tab of the ribbon and select **Correlation Rules Editor** from the drop-down list.

   The Correlation Rules Editor dialog box opens (see Figure 65).

2. In the Default rule set area, select the correlation group under which you wish to create your group and click **New Group**,

   -Or-

   Right-click the correlation group under which you wish to create your group and select **New Group** from the pop-up menu.

   A new group is created in the tree, at the specified location.

3. Click **OK**.

Defining Correlation Rules

You can modify the existing correlation rules and groups to better define your correlation requirements.

To modify an existing correlation rule:

1. Click **Correlation** in the **Home** tab of the ribbon and select **Correlation Rules Editor** from the drop-down list.

   The Correlation Rules Editor dialog box opens (see Figure 65).

2. In the Default rule set area, expand a correlation rule group.

   The correlation rules belonging to the group are displayed.

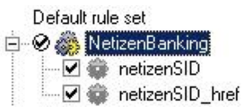
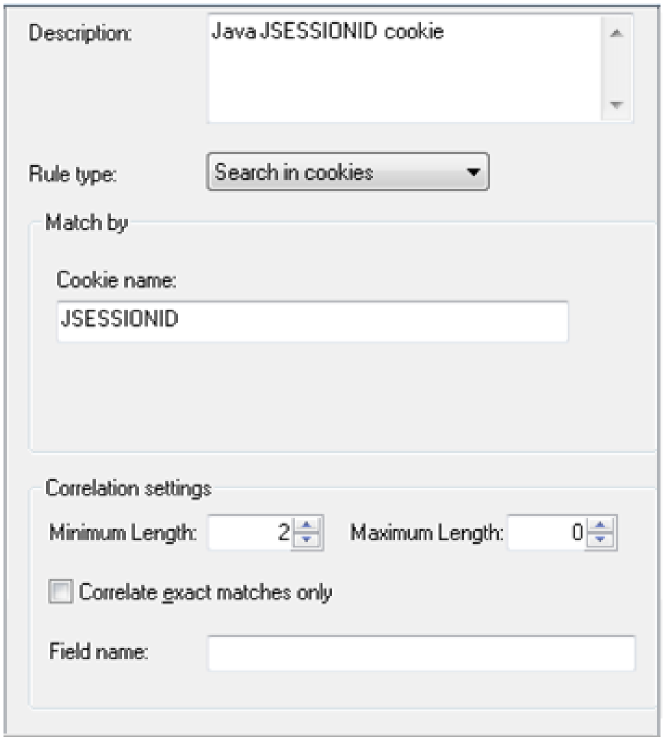


Figure 66: Correlation Rule Group – Expanded

3. Click a correlation rule. The Correlation Rules Properties appear.



Description: Java JSESSIONID cookie

Rule type: Search in cookies

Match by

Cookie name: JSESSIONID

Correlation settings

Minimum Length: 2 Maximum Length: 0

☐ Correlate exact matches only

Field name:

Figure 67: Correlation Rule Properties

Note: The **Match by** fields vary according to the value selected in the **Rule type** field.

4. Modify the correlation rule properties according to the information in the following table:

Table 11: Correlation Rule Properties

Field	Search Scope Value	Match by Field	Description
Description			A free text description of the selected rule.
Rule type			Determines the method used to find the dynamic data to be correlated. Possible rule type values are: • All body text • Form field values • User defined • Replace with expression • Search in cookies

Field	Search Scope Value	Match by Field	Description
	All body text		Search for dynamic data in the entire body of the HTTP response, not only in the links and forms. The dynamic data is uniquely identified by a combination of the Prefix and Suffix parameters. For example, if the dynamic data you wish to correlate appears as follows: `SessionID=1234&Day,` then `SessionID=` should be defined as the Prefix and `&` should be defined as the Suffix.
		Prefix	A text string that is used together with the Suffix parameter to uniquely identify the dynamic data string. The dynamic string should start immediately after the Prefix parameter.
		Suffix	A text string that is used together with the Prefix parameter to uniquely identify the dynamic data string. The dynamic string should end immediately before the Suffix parameter.
		Prefix Instance	The occurrence instance of the Prefix in the search scope. That is, if the Prefix parameter appears multiple times in the body text and you want to correlate only the second instance, select 2
		Reverse	Search for dynamic data from the end of the search scope. Note that if Reverse is selected, then Prefix Instance is also reversed. Thus, a value of 2 in Prefix Instance indicates the second from last occurrence instance.
		All instances	Search for all occurrences of the dynamic data. When All Instances is selected, both Prefix Instance and Reverse are not meaningful and are disabled.
	Form Field values		Search for dynamic data in specific form fields, regardless of if they are hidden. You can specify either a form field name or ID. For example, if you specify a form field ID, then the correlation engine will only search for form field IDs and not for form field names.

Field	Search Scope Value	Match by Field	Description
		By ID	Specify a form field ID to be searched.
		By Name	Specify a form field name to be searched.
	User defined		Search for dynamic data according to your own search criteria.
		JavaScript expression	Specify a JavaScript expression to be used as an extraction logic. This can be a valid regular expression, DOM access function, or any other function call. For example: `extractValue ("prefix", "suffix", document.wlSource )` To call a custom JavaScript function, the function must be accessible for both the execution engine and the correlation engine, so it should be included as an auto-discovered JavaScript code. For more information on the auto-discovery of JavaScript files, see *JavaScript Language Extension*, in the *WebLOAD Extensibility SDK*.
		Save Source	Select this option if your JavaScript expression refers to the response body (document.wlSource).

	Replace with expression	**Expression**	Replace data according to the Field name, regardless of the value. Replace the value with the value in Expression.
			For example:
			`Rule type = Replace with expression`
			`Expression = new Date().getTime()`
			`Field name = timestamp`
			During correlation, the timestamp value is replaced with "newDate().getTime()" instead of the date on which the script was recorded.

Field	Search Scope Value	Match by Field	Description
	Search in cookies	**Cookie name**	Search for dynamic data among the received cookies, according to the Cookie name. Replace the dynamic data in the query string or post data request. **Note:** In most cases, cookies sent by the server (in a Set-Cookie header) are echoed back by the client (in a Cookie header). WebLOAD's engine automatically handles these cases and they do not require correlation. Select this rule type when the value received in the cookie is used elsewhere in the request (using JavaScript) and correlation is necessary.
Correlation settings			
Minimum Length			Define the minimum length of the value to be considered for correlation. Shorter values, even if matched by a rule, are ignored.
Maximum Length			Define the maximum length of the value to be considered for correlation. Longer values, even if matched by a rule, are ignored.

Field	Search Scope Value	Match by Field	Description
Correlate exact matches only			Select this option to replace the identified dynamic value with a variable only when the values are a complete match. If the value found is only a part of the sent value, the rule will be ignored. For example, if the dynamic value found is "1234"and the variable replacing the dynamic value contains "123": • When **Correlate exact matches only** is unchecked, the value "1234" will be replaced by the variable and then a "4". • When **Correlate exact matches only** is checked, the value will not be replaced since it is not an exact match. **Note:** The values within a query string are also considered a complete match. For example, if the dynamic value is found in the following string, wlHttp.Get(http://domain?field=123), the dynamic value "123" will be replaced by the variable regardless of whether the Correlate exact matches only is checked. By default this is selected.
Field name			Replace the dynamic value only when the value is sent with this field name. **Note:** The **Field name** limits where the value can be replaced and is applicable to all rule types. Do not confuse this with the Search in: Form field values: by name field, which controls how the value is extracted This field is optional, unless **Replace with Expression** is selected, in which case, this field is mandatory.

5. Click **OK**.

   The correlation rule is modified.

Renaming Correlation Rules

You can rename correlation rules and groups to better organize the correlation rules according to your specific correlation requirements.

To rename a correlation rule:

1. Click **Correlation** in the **Home** tab of the ribbon and select **Correlation Rules Editor** from the drop-down list.

   The Correlation Rules Editor dialog box opens (see Figure 65).

2. Expand the correlation group to which your correlation rule is associated.
3. Slow double-click the correlation rule,

   -Or-

   Right-click the correlation rule and select **Rename** from the pop-up menu.

   The correlation rule's name becomes editable.

4. Rename the correlation rule and click anywhere in the Default rule set area.

   The correlation rule is renamed.

To rename a correlation group:

1. Click **Correlation** in the **Home** tab of the ribbon and select **Correlation Rules Editor** from the drop-down list.

   The Correlation Rules Editor dialog box opens (see Figure 65).

2. Slow double-click a correlation group,

   -Or-

   Right-click a correlation group and select **Rename** from the pop-up menu.

   The correlation group's name becomes editable.

3. Rename the correlation group and click anywhere in the Default rule set area.

   The correlation group is renamed.

Session Management

The HTTP protocol has no built-in method of uniquely identifying or tracking a particular user or session within an application, without transmitting some data between the client and the server. The most common method for an internet application developer to track a user's interaction with a website is by providing the user with a unique session ID. This process is referred to as *session management*. Most Web servers generate such session IDs for internet application developers. In such cases, the Web servers can communicate the session IDs between the user's internet browser and the server through:

- **Cookies** – When a server receives a request to create a session, it creates a session object and associates this object with a session ID. The session ID is then transmitted back to the browser as part of a response header and is stored with the rest of the cookies in the browser. On subsequent requests from the browser, the session ID is transmitted as part of the request header, which enables the application to associate each request for a session ID with the previous requests from that user. The entire interaction between the browser, application server, and the application is transparent to the end user.
- **URL rewriting** – The session ID information is embedded by the server in the URL and is then received by the application with the HTTP GET command (for example, when the client clicks on an embedded link within a page).
- **Hidden form fields** – The session ID information is stored within the fields of a form and submitted to the application. Typically, the session ID information is embedded within the form as a hidden field and is submitted with the HTTP GET/POST command.

The following sections provide information on how some of the most commonly used Web and application servers perform session management.

IBM WebSphere Application Server

The IBM WebSphere Application Server (WAS) supports all the session management methods listed in *Session Management* (on page 107), but works best with cookies (which is its default method). The WAS implementation of this method differs from a pure cookie-based method by using only one cookie, `JSESSIONID`, that contains only the session ID information. (A pure cookie-based method would use multiple cookies, containing possibly sensitive user state information, such as an account number or user ID.) `JSESSIONID` is used by the server to associate the request with the information already stored on the server for that session ID.

In an HTTP session, all the attributes associated with a user's request are stored on the server. Since the only information transmitted between the server and the browser is the session ID cookie, which has a limited lifetime, an HTTP session can provide a much more secure session management method than cookies, when configured in conjunction with SSL.

Microsoft ASP.NET

Microsoft ASP.NET uses HTTP cookies to send a user a unique session key. For example, an ASP.NET application that uses sessions can respond to a user's request with the following HTTP header:

```
Set-Cookie: ASPSESSIONID=PUYQGHUMEAAJPUYL; path=/Webapp
```

Any subsequent request made by this browser to the same server, in the virtual directory `/Webapp`, includes the following HTTP cookie header:

```
Cookie: ASPSESSIONID=PUYQGHUMEAAJPUYL
```

Each active ASP.NET session is identified and tracked using a 120-bit session ID string containing only the ASCII characters that are permitted in URLs. These session ID values are generated using an algorithm that guarantees a unique and random result. Such a guarantee is required to ensure that sessions do not collide and to prevent malicious user interference. These session ID strings are communicated between the client and server either by means of an HTTP cookie or a modified URL with the session ID string embedded, depending on how the application is configured.

Apache Server

An Apache server (version 1.3 onwards) uses cookies to identify a new user and then records access to the application identified by this unique ID, through the `mod_usertrack` module. When a user first visits the application, that user is sent a cookie with a unique ID. This unique ID is maintained until a predetermined timeout, thus enabling the server to track the user. This method enables the identification of different users even if they appear to originate from the same IP address. With Apache servers, the `CookieName` directive configures the name of the cookie that is stored.

Chapter 8

Running and Debugging Scripts

This section provides instructions for running and debugging scripts with WebLOAD Recorder.

About Running and Debugging Scripts with WebLOAD Recorder

When you run your script, WebLOAD Recorder interacts with your Web application just as a real user would. WebLOAD Recorder runs your script line by line. As your script executes, execution arrows are displayed in the left margin of the Script Tree and the JavaScript View pane, showing your progress.

Unless otherwise configured in the project options, the test session will log and continue on Minor Errors encountered during runtime. Severe Errors will cause WebLOAD Recorder to stop the entire test. If WebLOAD Recorder encounters HTTP errors that are undefined by WebLOAD, the test session logs them and continues running.

Messages, test failures, and differences are indicated by messages in the Log View Window.

After running a script, you can debug it. WebLOAD Recorder enables you to check that the script runs smoothly without errors, offers step controls to run through the script step-by-step, breakpoints, and various view and windows to monitor variables.

Running a Script

This section provides instructions for running a script.

Before running a script, you can do the following:

- Set the number of iterations to run, see *Setting Playback Options* (on page 209).
- Set the file locations for a test session, see *Setting File Locations* (on page 210).

- Set WebLOAD Recorder to ignore the recorded sleep time, see *Configuring Sleep Time Control Options* (on page 159).

Starting the Execution of a Script

To execute the script:

1. In the main window, click **Open** in the **File** tab of the ribbon and open the script you want to edit.
2. Click **Run** in the **Home** or **Debug** tab of the ribbon,

   -Or-

   Click **Step Into** in the **Debug** tab of the ribbon to run the script step-by-step.

   The script runs and displays the following:

   - A sequence of the events generated by the script in the Execution Tree.
   - The execution sequence in the JavaScript View pane and the Script Tree.
   - If the Page View tab is open, the pages returned from the Web site.

Note: If you specified more than one playback iteration, you are returned to the beginning of the script (for information on playback iteration, see *Setting Playback Options* on page 209).

Viewing the Execution Sequence in the Script Tree

When you run your script, WebLOAD Recorder interacts with your Web application just as a real user would. WebLOAD Recorder runs your script line by line. Execution arrows are displayed in the left margin of the Script Tree. When you select a node in the Script Tree, the corresponding information is displayed in each of the available views. For example, the Page View displays the page you have requested from the server, the HTML View displays the HTML of that page, and the HTTP Headers View displays the request and response's headers. For more information, see *Viewing and Analyzing the Test Results* (on page 130).

WebLOAD Recorder enables you to do the following:

- Run through the entire script line by line, and add breakpoints (see *Debugging a Script* on page 120).
- Display the Current Project Options by right-clicking the Agenda root node and clicking **Current Project Options** from the pop-up menu,

  -Or-

Click **Current Project Options** in the **Tools** tab of the ribbon (see *Configuring the Default and Current Project Options* on page 155).

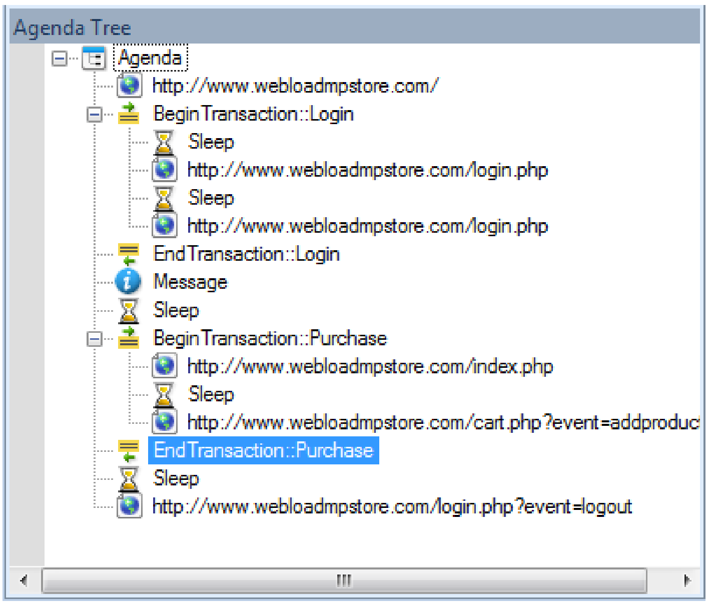
-Or-

Select **Current Project Options** from the WebLOAD Recorder **Home** tab of the ribbon (see *Configuring the Default and Current Project Options* on page 155).

To view the Script Tree:

- In the main window, click **Visual Mode** in the **Home** tab.

  By default, the Script Tree pane appears at the top left of the main window, to the right of the WebLOAD Recorder Toolbox pane.


Agenda Tree

```
Agenda
  http://www.webloadmpstore.com/
  BeginTransaction::Login
    Sleep
    http://www.webloadmpstore.com/login.php
    Sleep
    http://www.webloadmpstore.com/login.php
  EndTransaction::Login
  Message
  Sleep
  BeginTransaction::Purchase
    http://www.webloadmpstore.com/index.php
    Sleep
    http://www.webloadmpstore.com/cart.php?event=addproduc
  EndTransaction::Purchase
  Sleep
  http://www.webloadmpstore.com/login.php?event=logout
```

Figure 68: Script Tree Pane

Viewing the Execution Sequence in the JavaScript View Pane

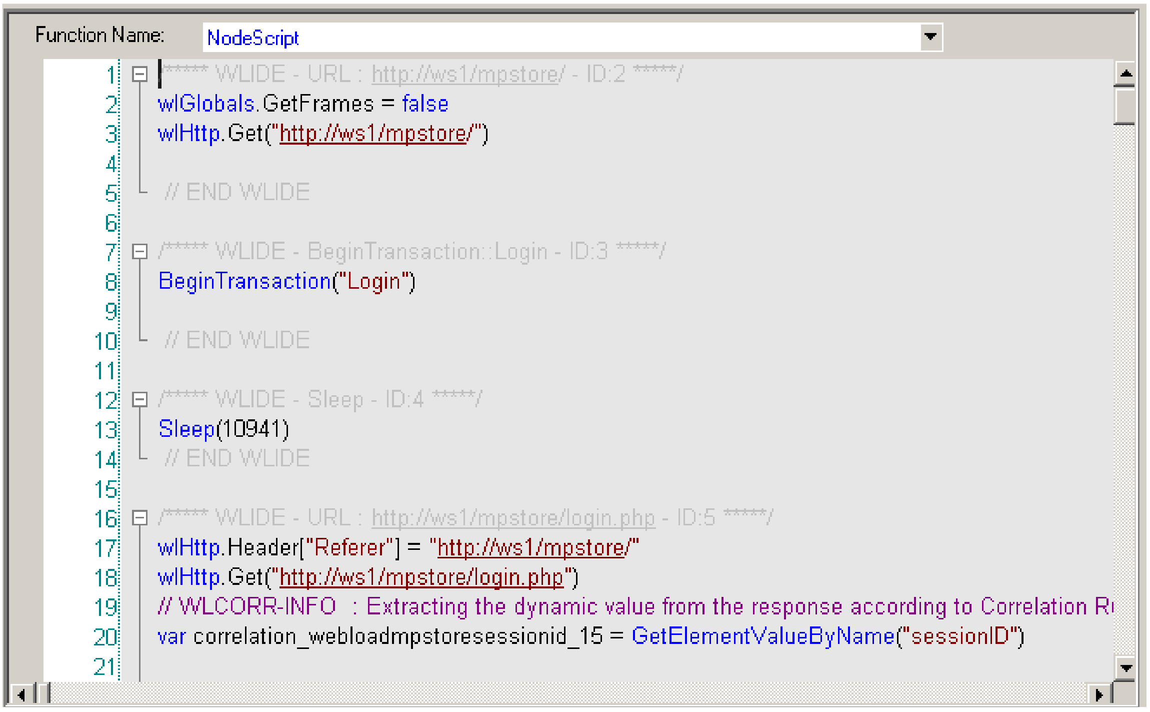
JavaScript View displays the complete JavaScript of your script with an execution arrow tracking its progress during runtime.

WebLOAD Recorder enables you to do the following:

- Run through the entire script line by line, add breakpoints, and add Watch variables (see *Debugging a Script* on page 120).
- Check the syntax by right-clicking in the script and clicking **Check Syntax** from the pop-up menu.

To view the JavaScript View:

- In the main window, select the **JavaScript View** checkbox in the **View** tab.



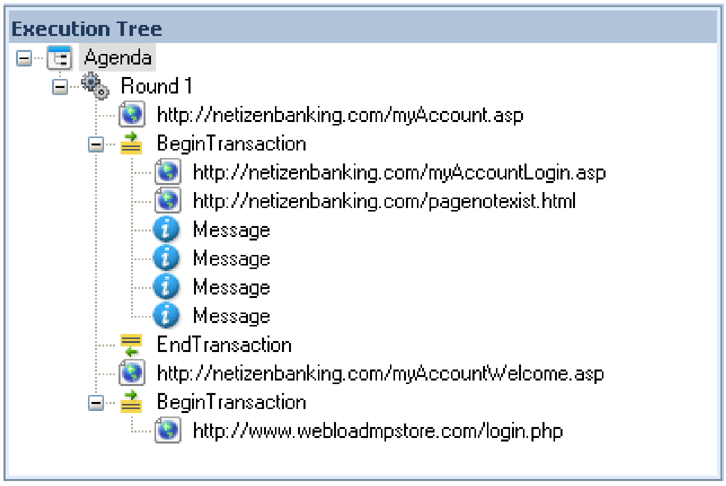
Function Name: NodeScript

```
/***** WLIDE - URL : http://ws1/mpstore/ - ID:2 *****/
wlGlobals.GetFrames = false
wlHttp.Get("http://ws1/mpstore/")

 // END WLIDE

/***** WLIDE - BeginTransaction::Login - ID:3 *****/
BeginTransaction("Login")

 // END WLIDE

/***** WLIDE - Sleep - ID:4 *****/
Sleep(10941)
 // END WLIDE

/***** WLIDE - URL : http://ws1/mpstore/login.php - ID:5 *****/
wlHttp.Header["Referer"] = "http://ws1/mpstore/"
wlHttp.Get("http://ws1/mpstore/login.php")
// WLCORR-INFO  : Extracting the dynamic value from the response according to Correlation Ru
var correlation_webloadmpstoresessionid_15 = GetElementValueByName("sessionID")
```

Figure 69: JavaScript View

Viewing the Response Data in the Execution Tree

As you execute a script, WebLOAD Recorder displays the actions performed during runtime in the Execution Tree. The Execution Tree is an interactive tree that you can use to examine the results.



```
Execution Tree
Agenda
  Round 1
    http://netizenbanking.com/myAccount.asp
    BeginTransaction
      http://netizenbanking.com/myAccountLogin.asp
      http://netizenbanking.com/pagenotexist.html
      Message
      Message
      Message
      Message
    EndTransaction
    http://netizenbanking.com/myAccountWelcome.asp
    BeginTransaction
      http://www.webloadmpstore.com/login.php
```

Figure 70: Execution Tree

Comparing Recorded Sequence and Execution Sequence Side by Side side by side

You can view the recorded sequence alongside the execution sequence using the side by side view feature. This can help you manually discover differences between the original recorded session and the playback.

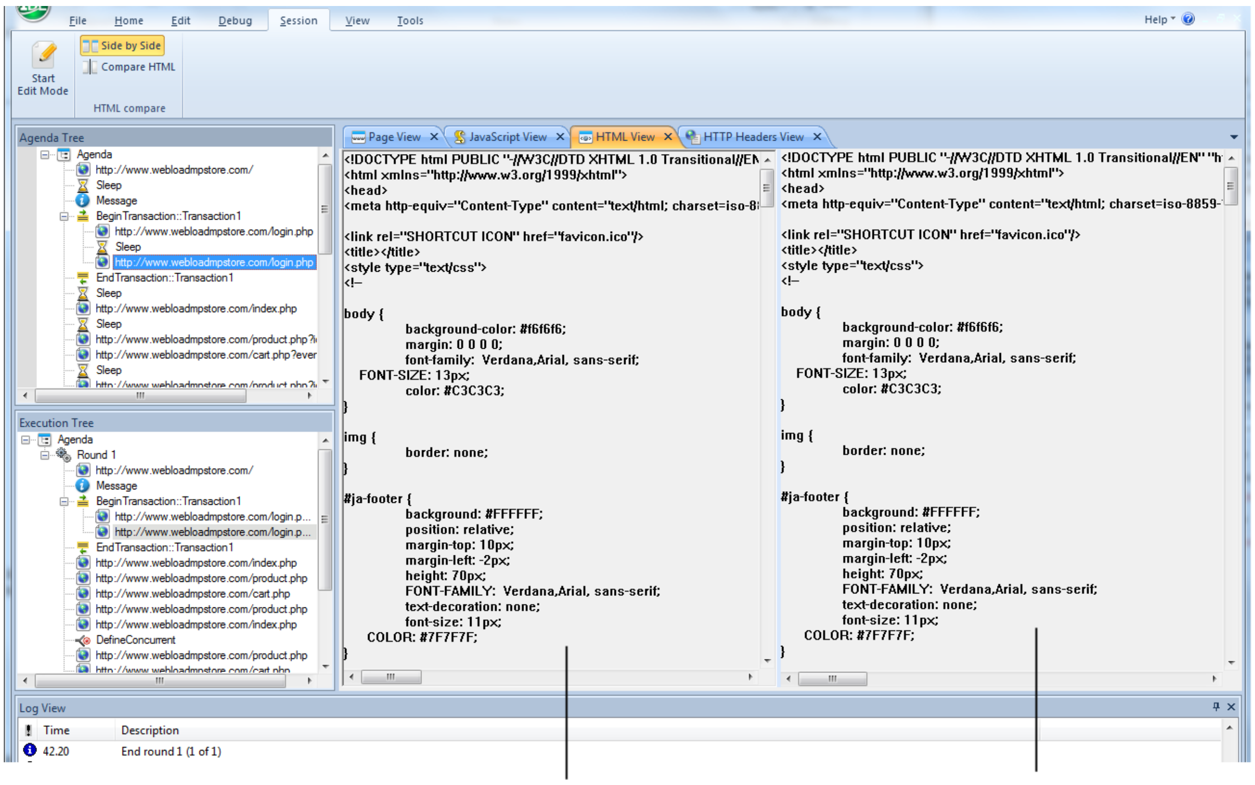
Side by side view is available for the Browser, HTTP Headers, and HTML Views.

To view the recorded sequence side by side with the execution sequence:

- Select **Side by Side** in the **Session** tab of the ribbon.

  The recorded sequence is displayed to the left of the execution sequence.

Note: This feature is only available after complete running an execution sequence, not at the beginning of an execution sequence.



Agenda Tree

```
Agenda
  http://www.webloadmpstore.com/
  Sleep
  Message
  BeginTransaction::Transaction1
    http://www.webloadmpstore.com/login.php
    Sleep
    http://www.webloadmpstore.com/login.php
  EndTransaction::Transaction1
  Sleep
  http://www.webloadmpstore.com/index.php
  Sleep
  http://www.webloadmpstore.com/product.php?i
  http://www.webloadmpstore.com/cart.php?ever
  Sleep
```

Execution Tree

```
Agenda
  Round 1
    http://www.webloadmpstore.com/
    Message
    BeginTransaction::Transaction1
      http://www.webloadmpstore.com/login.p...
      http://www.webloadmpstore.com/login.p...
    EndTransaction::Transaction1
    http://www.webloadmpstore.com/index.php
    http://www.webloadmpstore.com/product.php
    http://www.webloadmpstore.com/cart.php
    http://www.webloadmpstore.com/product.php
    http://www.webloadmpstore.com/index.php
    DefineConcurrent
    http://www.webloadmpstore.com/product.php
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8

<link rel="SHORTCUT ICON" href="favicon.ico"/>
<title></title>
<style type="text/css">
<!–

body {
        background-color: #f6f6f6;
        margin: 0 0 0 0;
        font-family:  Verdana,Arial, sans-serif;
  FONT-SIZE: 13px;
        color: #C3C3C3;
}

img {
        border: none;
}

#ja-footer {
        background: #FFFFFF;
        position: relative;
        margin-top: 10px;
        margin-left: -2px;
        height: 70px;
        FONT-FAMILY:  Verdana,Arial, sans-serif;
        text-decoration: none;
        font-size: 11px;
   COLOR: #7F7F7F;
}
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "h
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-

<link rel="SHORTCUT ICON" href="favicon.ico"/>
<title></title>
<style type="text/css">
<!–

body {
        background-color: #f6f6f6;
        margin: 0 0 0 0;
        font-family:  Verdana,Arial, sans-serif;
  FONT-SIZE: 13px;
        color: #C3C3C3;
}

img {
        border: none;
}

#ja-footer {
        background: #FFFFFF;
        position: relative;
        margin-top: 10px;
        margin-left: -2px;
        height: 70px;
        FONT-FAMILY:  Verdana,Arial, sans-serif;
        text-decoration: none;
        font-size: 11px;
   COLOR: #7F7F7F;
}
```

Log View

!	Time	Description
●	42.20	End round 1 (1 of 1)

Figure 71: Side by Side View

Comparing a Script Recording node to its Playback

After running a script test, you can perform a comparison of the original script recording and its playback for each node in the script.

To compare a script node's recording to its playback:

1. Select a script node.
2. Click **Compare HTML** in the **Session** tab of the ribbon.

   -Or-

   Right-click a node in the Script Tree or the Execution Tree and select **Compare html** from the pop-up menu.

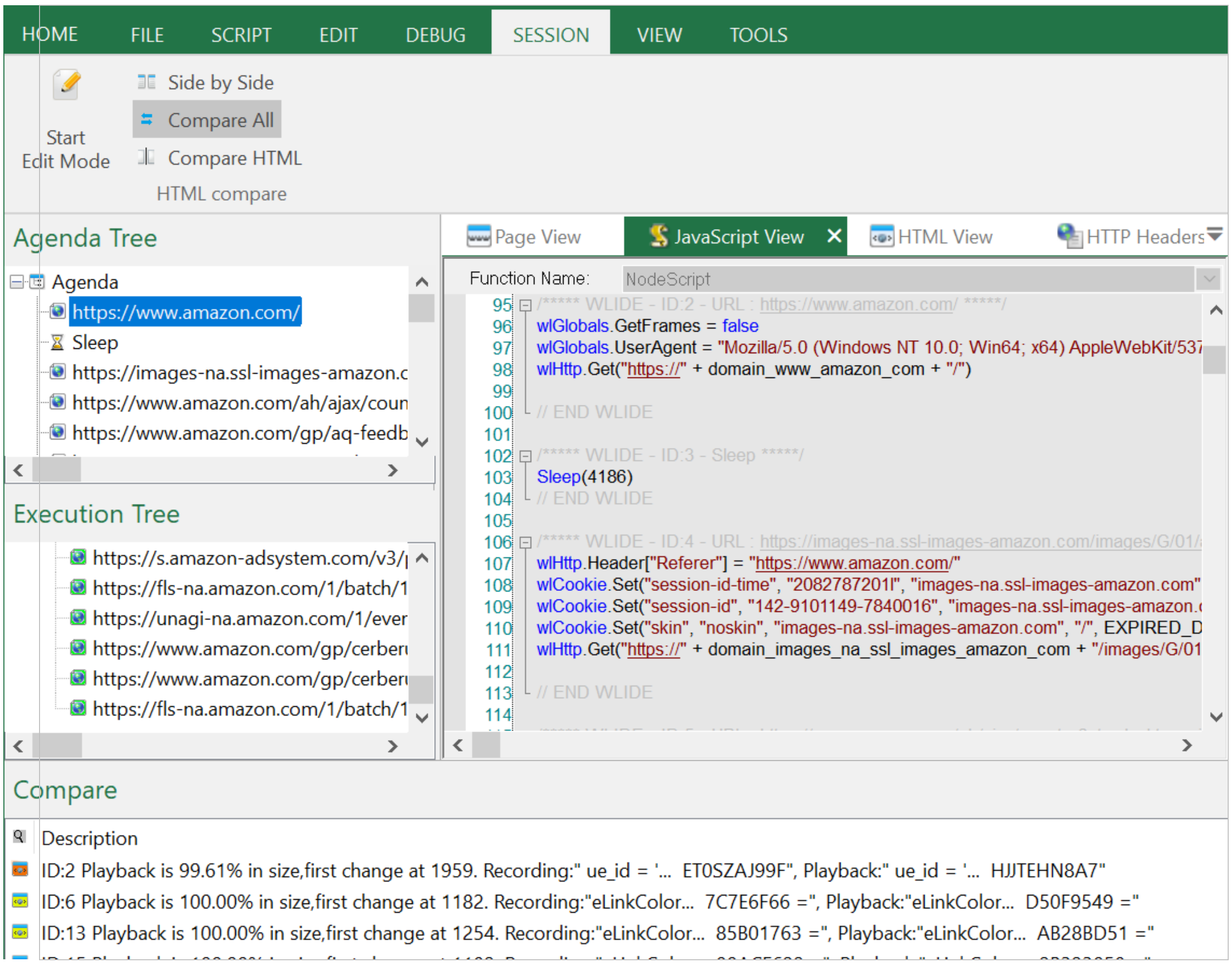
   The defined Difference Viewer application launches and automatically compares the selected node in the recording and in the playback. For information about defining the Difference Viewer application, see *Defining the Difference Viewer Application* (on page 211).

Comparing a Recorded Sequence with the Execution Sequence

You can compare an entire original recorded sequence with the current execution sequence using the Compare All feature. This can help you to check and fix any undesirable changes you made to the original recorded session.

To compare the current execution with the original sequence:

- Select **Compare All** in the **Session** tab of the ribbon.



Agenda
https://www.amazon.com/
Sleep
https://images-na.ssl-images-amazon.c
https://www.amazon.com/ah/ajax/coun
https://www.amazon.com/gp/aq-feedb

https://s.amazon-adsystem.com/v3/
https://fls-na.amazon.com/1/batch/1
https://unagi-na.amazon.com/1/ever
https://www.amazon.com/gp/cerber
https://www.amazon.com/gp/cerber
https://fls-na.amazon.com/1/batch/1

Function Name: NodeScript

```
/***** WLIDE - ID:2 - URL : https://www.amazon.com/ *****/
wlGlobals.GetFrames = false
wlGlobals.UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
wlHttp.Get("https://" + domain_www_amazon_com + "/")

// END WLIDE

/***** WLIDE - ID:3 - Sleep *****/
Sleep(4186)
// END WLIDE

/***** WLIDE - ID:4 - URL : https://images-na.ssl-images-amazon.com/images/G/01/
wlHttp.Header["Referer"] = "https://www.amazon.com/"
wlCookie.Set("session-id-time", "2082787201l", "images-na.ssl-images-amazon.com"
wlCookie.Set("session-id", "142-9101149-7840016", "images-na.ssl-images-amazon.
wlCookie.Set("skin", "noskin", "images-na.ssl-images-amazon.com", "/", EXPIRED_D
wlHttp.Get("https://" + domain_images_na_ssl_images_amazon_com + "/images/G/01

// END WLIDE
```

Description
ID:2 Playback is 99.61% in size,first change at 1959. Recording:" ue_id = '... ET0SZAJ99F", Playback:" ue_id = '... HJJTEHN8A7"
ID:6 Playback is 100.00% in size,first change at 1182. Recording:"eLinkColor... 7C7E6F66 =", Playback:"eLinkColor... D50F9549 ="
ID:13 Playback is 100.00% in size,first change at 1254. Recording:"eLinkColor... 85B01763 =", Playback:"eLinkColor... AB28BD51 ="

Figure 72: Compare All View

The following comparison information and options are provided:

Execution Tree pane in Compare All mode

In the **Execution Tree** pane, each node's background color indicates the degree to which the original execution was changed, as follows:

🟩 Green – The current execution is identical with the original execution

🟨 Yellow – There is a minor difference between the original and the current execution

🟥 Red – There is a major difference between the original and the current execution

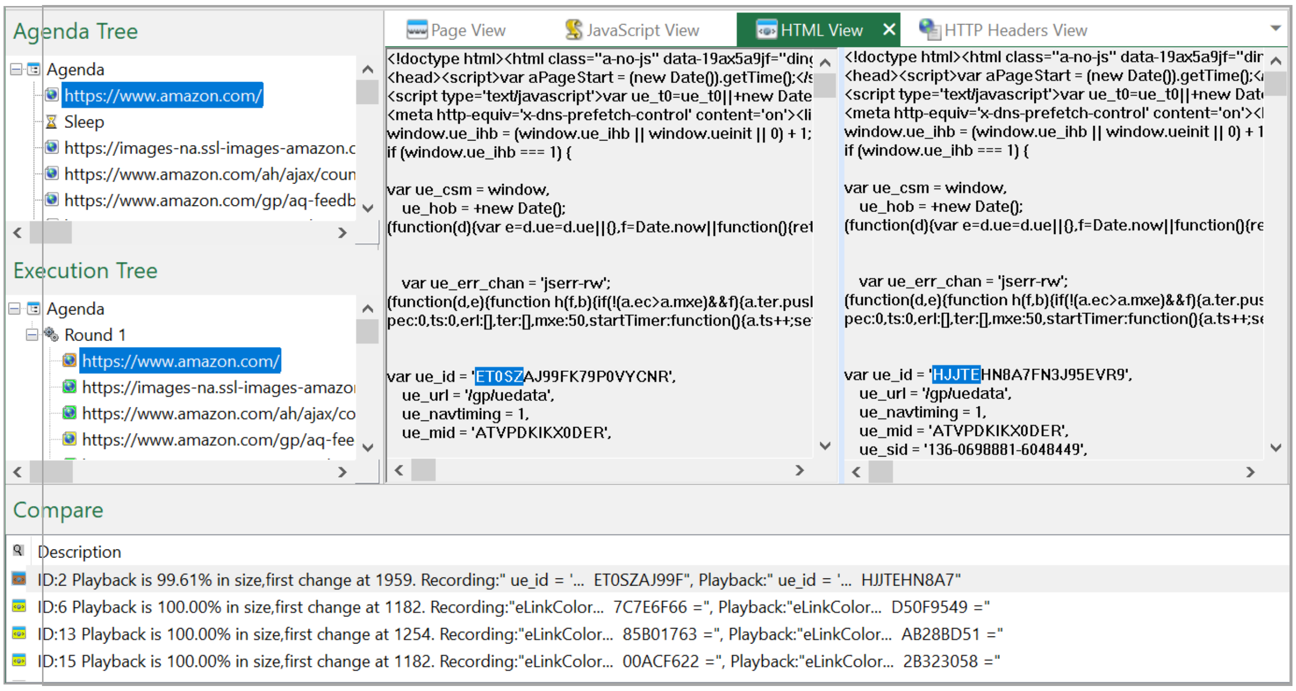
Compare pane

The **Compare** pane that appears at the bottom of the screen lists a digest of the differences between the original and the current execution. The icon at the beginning of each line indicates the level of difference:

🟨 Yellow – indicates a minor change.

🟥 Red – indicates a major change.

Double clicking a line, or right-clicking a line and selecting **Go to location**, opens a side-by-side view of the relevant code in the view pane, with the first difference highlighted in blue. The original recorded sequence is displayed on the left and the current execution is displayed on the right. The side by side comparison can be viewed in the Browser, HTTP Headers, and HTML Views.


Agenda Tree
Agenda
https://www.amazon.com/
Sleep
https://images-na.ssl-images-amazon.c
https://www.amazon.com/ah/ajax/coun
https://www.amazon.com/gp/aq-feedb
Execution Tree
Agenda
Round 1
https://www.amazon.com/
https://images-na.ssl-images-amazo
https://www.amazon.com/ah/ajax/co
https://www.amazon.com/gp/aq-fee
Page View
JavaScript View
HTML View
HTTP Headers View
Compare

```
<!doctype html><html class="a-no-js" data-19ax5a9jf="ding
<head><script>var aPageStart = (new Date()).getTime();</s
<script type='text/javascript'>var ue_t0=ue_t0||+new Date
<meta http-equiv='x-dns-prefetch-control' content='on'><li
window.ue_ihb = (window.ue_ihb || window.ueinit || 0) + 1;
if (window.ue_ihb === 1) {

var ue_csm = window,
   ue_hob = +new Date();
(function(d){var e=d.ue=d.ue||{},f=Date.now||function(){ret


   var ue_err_chan = 'jserr-rw';
(function(d,e){function h(f,b){if(!(a.ec>a.mxe)&&f){a.ter.pusl
pec:0,ts:0,erl:[],ter:[],mxe:50,startTimer:function(){a.ts++;se


var ue_id = 'ET0SZAJ99FK79P0VYCNR',
   ue_url = '/gp/uedata',
   ue_navtiming = 1,
   ue_mid = 'ATVPDKIKX0DER',
```

```
<!doctype html><html class="a-no-js" data-19ax5a9jf="dir
<head><script>var aPageStart = (new Date()).getTime();<
<script type='text/javascript'>var ue_t0=ue_t0||+new Dat
<meta http-equiv='x-dns-prefetch-control' content='on'><
window.ue_ihb = (window.ue_ihb || window.ueinit || 0) + 1
if (window.ue_ihb === 1) {

var ue_csm = window,
   ue_hob = +new Date();
(function(d){var e=d.ue=d.ue||{},f=Date.now||function(){re


   var ue_err_chan = 'jserr-rw';
(function(d,e){function h(f,b){if(!(a.ec>a.mxe)&&f){a.ter.pus
pec:0,ts:0,erl:[],ter:[],mxe:50,startTimer:function(){a.ts++;se


var ue_id = 'HJJTEHN8A7FN3J95EVR9',
   ue_url = '/gp/uedata',
   ue_navtiming = 1,
   ue_mid = 'ATVPDKIKX0DER',
   ue_sid = '136-0698881-6048449',
```

	Description
🟥	ID:2 Playback is 99.61% in size,first change at 1959. Recording:" ue_id = '... ET0SZAJ99F", Playback:" ue_id = '... HJJTEHN8A7"
🟨	ID:6 Playback is 100.00% in size,first change at 1182. Recording:"eLinkColor... 7C7E6F66 =", Playback:"eLinkColor... D50F9549 ="
🟨	ID:13 Playback is 100.00% in size,first change at 1254. Recording:"eLinkColor... 85B01763 =", Playback:"eLinkColor... AB28BD51 ="
🟨	ID:15 Playback is 100.00% in size,first change at 1182. Recording:"eLinkColor... 00ACF622 =", Playback:"eLinkColor... 2B323058 ="

Figure 73: Go to Location

In addition, right-clicking a line and selecting **Compare HTML**, launches the defined Difference Viewer application that compares the selected node in the original recording with the current execution. For more information see *Comparing a Script Recording node to its Playback* (on page 115).

Stopping the Execution of a Script

When debugging a script using a Step Into or breakpoint, the playback session stops immediately upon completion of the current WebLOAD Recorder protocol block.

To stop the execution of a script:

- Click **Stop** in the **Home** tab of the ribbon,

  -Or-

Use the hotkeys **Shift + F5**.

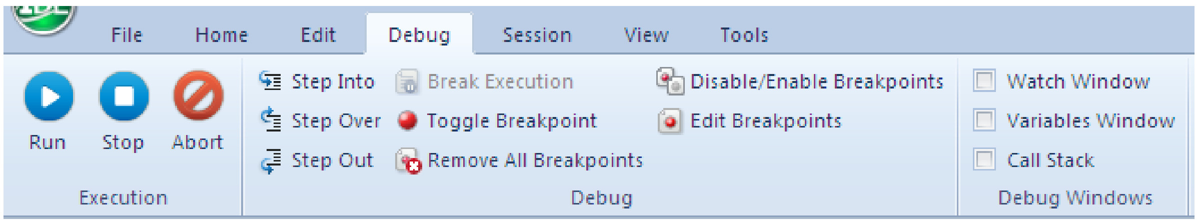
The playback session is stopped.

Debugging Scripts

WebLOAD Recorder provides an integrated debugger with a variety of tools to help locate bugs in your script. The debugger provides special menus, windows, dialog boxes, and grids of fields for debugging. You can pause the debugger and trigger WebLOAD Recorder to wait for user input before proceeding with running the script. In the script, you can set breakpoints and step into / over / out. While debugging your script, you can abort the debugger without executing the `TerminateClient` and `TerminateAgenda` functions, as opposed to stopping it completely.

Debug Tab Items

Commands for debugging can be found on the **Debug** tab of the ribbon.



The **Debug** tab contains commands to start the debugging process.

The following options are available through the **Debug** tab.

Table 12: Debug Tab Options

Tab Item	Description
Execution group	
Run	Starts playback of the script from the current statement until a breakpoint or the end of the script is reached.
Stop	Stops the playback of the script.
Abort	Stops the playback of the script without executing the `TerminateClient` or `TerminateAgenda` functions.
Debug group	
Step Into	Starts the play back of the script, a step at a time, entering each function encountered.

Tab Item	Description
Step Over	Starts the playback of the script, one step at a time. When a function is reached, it is executed without stepping through the function.
Step Out	Plays through the remaining steps of the called function, and stops on the line in the script immediately following the function call. Using this command you can quickly finish executing the current function after determining that a bug is not present in the function.
Break Execution	Stops the playback of the script at that point.
Toggle Breakpoint	Defines a line in the script where WebLOAD Recorder suspends execution.
Remove all Breakpoints	Eliminates all breakpoints.
Disable/Enable all Breakpoints	Disable or enable all breakpoints.
Edit Breakpoints	Displays the Breakpoints dialog box, enabling the setting of breakpoints.
Debug Windows group	
Watch Window	Toggles the displays of the Watch window (available only during runtime in debug mode), which displays the names and values of variables and expressions..
Variables Window	Toggles the display of the Variables window (available only during runtime in debug mode) which displays information about variables used in the current and previous statements and functions..
Call Stack	Toggles the display of the Call Stack window which lists the function calls that led to the current statement, with the current function on the top of the stack..

Debugging a Script

When debugging a script, you can set the script to run in the following ways:

- Step-by-step – The execution starts at the first line of the script and stops at each subsequent line.
- Breakpoints – The execution starts at the first line of the script and stops when it reaches a breakpoint.
- A combination of step-by-step and breakpoints.

To debug a script:

1. Click **Run** or **Step Into** in the **Debug** tab of the ribbon.
2. When you reach the end of the script you can:
   a. Click **Step Into** in the **Debug** tab to return to the beginning of the script.
   b. View results (see *Viewing and Analyzing the Test Results* on page 130).
   c. Add breakpoints (see *Setting Breakpoints* on page 121).
3. Return to Edit mode and revise your script.

Starting the Debugger

To start debugging:

- Click **Run** in the **Debug** tab to run the script continuously,

  -Or-

  Click **Step Into** in the **Debug** tab to run the script step-by-step.

Setting Breakpoints

Use breakpoints to define places in the script to suspend execution. Breakpoints can be set in Edit mode and in Debug mode. The breakpoints you set will be saved as a part of your WebLOAD Recorder project.

To set multiple breakpoints to a script:

1. Display the entire script.
2. Select the line of code.
3. Click **Edit Breakpoints** in the **Debug** tab.

   The Breakpoints dialog box opens.

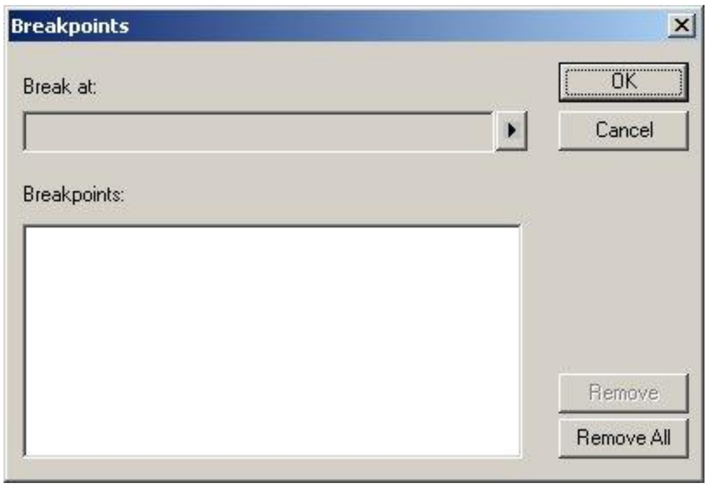


Figure 74: Breakpoints Dialog Box

4. Click the arrow next to the **Break at** field.

   The Breakpoint options appear.

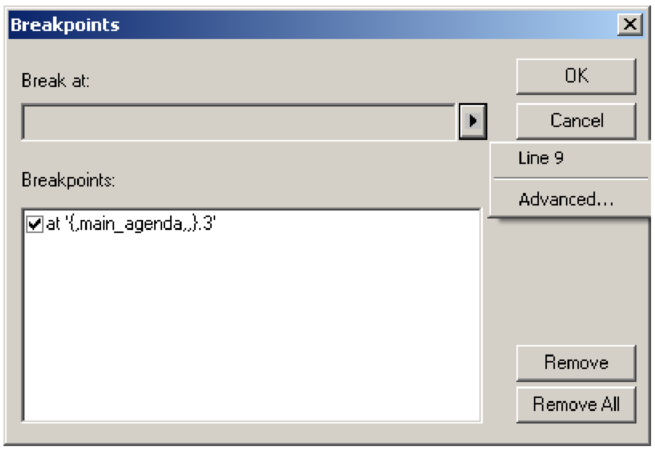


Figure 75: Breakpoint Dialog Box Options

5. Click the Line number.

   The Line number is added to the list of breakpoints.

6. To add context to the breakpoint, click the arrow again, and click **Advanced**.

   The Advanced Breakpoint dialog box opens.

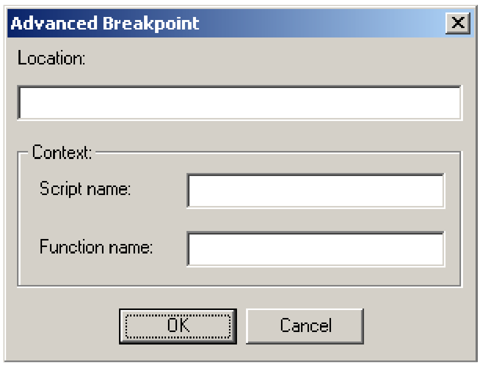


Figure 76: Advanced Breakpoint Dialog Box

7. Fill in the fields, and click **OK**.

To set a breakpoint in the Script Tree:

1. Right-click an item in the Script Tree.
2. From the pop-up menu, click **Toggle Breakpoint**.

   A red dot appears in the left margin of the JavaScript View pane adjacent to the selected code and in the Script Tree adjacent to the visual script element for which the breakpoint is defined, indicating that the breakpoint is set.

To set a breakpoint in the JavaScript View pane:

1. Select the **JavaScript View** checkbox in the **View** tab to open the JavaScript View pane.
2. In the Script Tree, click the **Agenda** root node to display the entire script in the JavaScript View pane.
3. In the JavaScript View pane, select the line of code where you want the script to wait.
4. Right-click and select **Toggle Breakpoint** from the pop-up menu,

   -Or-

   Click **Toggle Breakpoint** in the **Debug** tab.

   A red dot appears in the left margin of the JavaScript View pane adjacent to the selected code and in the Script Tree adjacent to the visual script element for which the breakpoint is defined, indicating that the breakpoint is set.

To set a breakpoint in Debug mode:

1. Run the script by clicking **Step Into** in the **Debug** tab.
2. Continue stepping through the script until reaching the point you want to insert the breakpoint.

3. In the JavaScript View pane, select the code where you want to insert in breakpoint.
4. Click **Toggle Breakpoint** in the **Debug** tab.

   While in debug mode a red dot appears in the left margin of your script code, indicating that the breakpoint is set.

Running to a Breakpoint

To run until a breakpoint is reached:

1. Set a breakpoint (see *Setting Breakpoints* on page 121).
2. Click **Run** in the **Debug** tab

   Click **Step Into** in the **Debug** tab to run the script step-by-step.

Removing Breakpoints

You can remove individual breakpoints or remove all breakpoints in the script.

To remove a breakpoint:

1. Select the **JavaScript View** checkbox in the **View** tab to open the JavaScript View pane.
2. In the Script Tree, click the **Agenda** root node to display the entire script in the JavaScript View pane.
3. In the JavaScript View pane, select the line containing the breakpoint you want to remove.
4. Click **Toggle Breakpoint** in the **Debug** tab.

   The red dot in the left margin disappears.

To remove all breakpoints:

1. Select the **JavaScript View** checkbox in the **View** tab to open the JavaScript View pane.
2. In the Script Tree, click the **Agenda** root node to display the entire script in the JavaScript View pane.
3. Click **Remove all Breakpoints** in the **Debug** tab

   The red dot in the left margin disappears.

Disabling and Enabling All Breakpoints

You can disable or enable all breakpoints in the script.

To disable or enable all breakpoints:

1. Select the **JavaScript View** checkbox in the **View** tab to open the JavaScript View pane.
2. In the Script Tree, click the **Agenda** root node to display the entire script in the JavaScript View pane.
3. Click **Disable/Enable Breakpoints** in the **Debug** tab.
   - When all of the breakpoints are disabled, the red dots in the left margin turn white.
   - When all of the breakpoints are enabled, the white dots in the left margin turn red.

Stepping Into the Script

To run the script and execute one statement at a time (Step Into):

1. Click **Run** or **Step Into** in the **Debug** tab.

   The debugger executes the next statement and then it pauses execution. If you step into a nested function call, the debugger steps into the most deeply nested function.

2. Repeat step **1** to continue executing the script one statement at a time.

To step into a specific function:

1. Set a breakpoint just before the function call or use the **Step Into** command to advance the script execution to that point.

   For information on setting breakpoints see *Setting Breakpoints* (on page 121).

2. Click **Step Into** in the **Debug** tab.

Stepping Out or Over a Function

To step over a function:

1. Click **Run** or **Step Into** in the **Debug** tab.
2. Execute the script to the function call.
3. Click **Step Over** in the **Debug** tab.

The debugger executes the next function, but pauses after the function returns.

4. Continue executing the program.

To step out of a function:

1. Click **Run** or **Step Into** in the **Debug** tab and execute the program to some point inside the function.
2. Click **Step Out** in the **Debug** tab.

   The debugger continues until it has completed execution of the return from the function, then pauses.

Stopping the Playback of the Script

You can stop the playback of the script at a specific point. Stopping a script executes the `TerminateClient` or `TerminateAgenda` functions.

To stop the playback of the script:

1. Start debugging. Click **Run** or **Step Into** in the **Debug** tab.
2. Click **Break Execution** in the **Debug** tab.

   The script stops running. You can continue the playback from this point, at a later time.

3. Click **Stop** in the **Debug** tab.

   The script stops running. Continuing the playback from this point is not possible.

Aborting the Playback of the Script

You can abort the playback of the script at a specific point. Aborting a script does not execute the `TerminateClient` or `TerminateAgenda` functions.

To abort the execution of a script:

- Click **Abort** in the **Debug** tab.

  The playback session is aborted.

Using the Watch Window

The Watch window is used for debugging your application, and is only available when you are running your script. The Watch window displays the values of selected variables or watch expressions that you specify while debugging your script. The values of the variables and expressions in the Watch window are only updated when execution is stopped at a breakpoint.

Use the Watch window to specify variables and expressions that you want to watch while debugging your program. You can also modify the value of a variable using the Watch window. To add a watch variable, see *Adding a Watch Variable or Expression* (on page 128).

To open the Watch window:

- Select the **Watch Window** checkbox in the **Debug** tab.

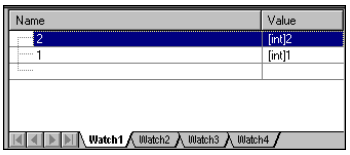


Figure 77: Watch Window

The Watch window contains four tabs:

- Watch1
- Watch2
- Watch3
- Watch4

Each tab displays a user-specified list of variables and expressions in a grid field. You can group variables that you want to watch together onto the same tab. For example, you could put variables related to a specific page on one tab and variables related to second page on another tab. You could watch the first tab when debugging the first page and the second tab when debugging the second page.

If you add an array variable to the Watch window, plus sign (+) or minus sign (-) boxes appear in the Name column. You can use these boxes to expand or collapse your view of the variable.

Viewing the Value of a Variable in the Watch Window

You can view the value of a variable in the Watch window.

To view a variable or expression in the Watch window:

1. Start debugging. Click **Run** or **Step Into** in the **Debug** tab of the ribbon.
2. Select the **Watch Window** checkbox in the **Debug** tab to open the Watch window.

In the Name column, plus sign (+) or minus sign (-) boxes may appear. These appear if you added an array or object variable to the Watch window. Use these boxes to expand or collapse your view of the variable.

Adding a Watch Variable or Expression

You can add a watch variable or expression to the Watch window, while you are running your script. Valid expressions accepted in the Watch window include any valid JavaScript expression that can be added to the script.

The Watch dialog box is equivalent to using the JavaScript `eval` function. Using the `eval` function you can define a variable and its value. In the same way, you can use the watch dialog box to define values for variables used throughout a script.

For example, if your script contains the variable `a`, when you type `a=10` in the Watch window, the engine evaluates the expression as though it were written within the script. The result of the expression `a=10` would be setting the variable `a` to 10. Then when you type `a=a+1` in the watch window, the variable `a` would be set to 11. The value of the variable is always according to the last definition of the variable. So, if you type `a=2`, the variable `a` would be set to 2 regardless of what the variable's value was beforehand.

To add a Watch variable or expression in the JavaScript View pane:

1. Start debugging. Click **Run** or **Step Into** in the **Debug** tab.
2. Select the **JavaScript View** option in the **View** tab to open the JavaScript View pane.
3. In the Script Tree, click the **Agenda** root node to display the entire script in the JavaScript View pane.
4. In the JavaScript View pane, select the line where you want to add the Watch variable or expression.
5. Right-click the variable in the JavaScript View pane, and click **Add Watch** from the pop-up menu.

   The Add Watch dialog box opens.

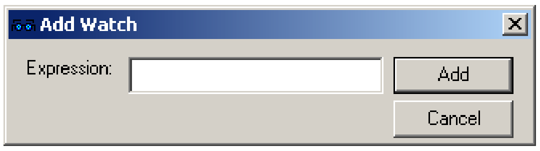


Figure 78: Add Watch Dialog Box

6. In the Expression field, type a variable or expression.
7. Click **Add**.

The variable or expression is added to the Watch window. The Watch window evaluates the variable or expression immediately and displays the value or an error message.

If you added an array or object variable to the Watch window, plus sign (+) or minus sign (-) boxes appear in the Name column. Use these boxes to expand or collapse your view of the variable.

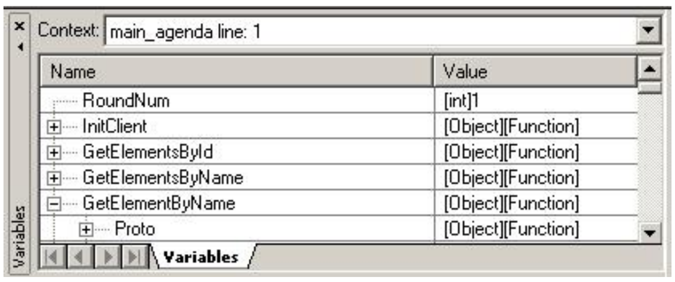
8. You can optionally edit the name or value of the variable or expression by double-clicking the name or value that you want to edit.

Viewing the Variables Window

The Variables window provides quick access to variables that are important in the scripts current context.

To open the Variables Window:

1. Start debugging. Click **Run** or **Step Into** in the **Debug** tab.
2. Select the **Variables Window** checkbox in the **Debug** tab.



Context: main_agenda line: 1

Name	Value
RoundNum	[int]1
⊞ InitClient	[Object][Function]
⊞ GetElementsById	[Object][Function]
⊞ GetElementsByName	[Object][Function]
⊟ GetElementByName	[Object][Function]
⊞ Proto	[Object][Function]

Figure 79: Variables Window

The Variables window displays variables used in the current statement and in the previous statement. It also displays return values when you step over or out of a function.

The Variables window contains a grid with fields for the variable name and value. The debugger automatically fills in these fields. You cannot add variables or expressions to the Variables window (you must use the Watch window, see *Adding a Watch Variable or Expression* on page 128), but you can expand or collapse the variables shown. You can expand an array, object, or structure variable in the Variables window if it has a plus sign (+) box in the Name field. If an array, object, or structure variable has a minus sign (-) box in the Name field, the variable is already fully expanded.

The Variables window also has a Context dropdown list that displays the current scope of the variables displayed. To view variables in a different scope, select the scope from the drop-down list box.

Viewing the Value of a Variable

You can view the value of a variable in the Variables window.

To view a variable in the Variables window:

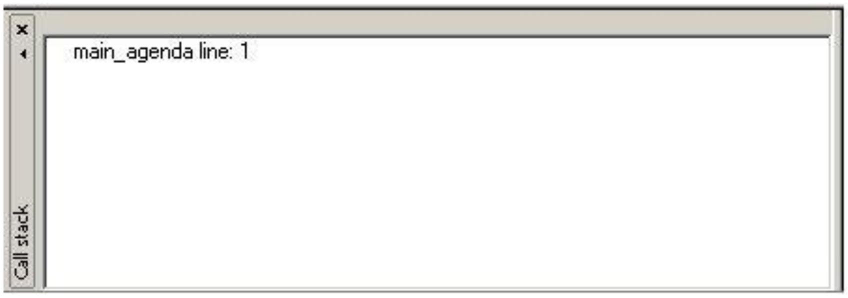
1. Start debugging. Click **Run** or **Step Into** in the **Debug** tab.
2. Select the **Variables Window** checkbox in the **Debug** tab to open the Variables window.

Viewing the Call Stack Window

The Call Stack window lists the function calls that led to the current statement, with the current function on the top of the stack.

To open the Call Stack Window:

1. Start debugging. Click **Run** or **Step Into** in the **Debug** tab.
2. Select the **Call Stack** checkbox in the **Debug** tab.



main_agenda line: 1

Figure 80: Call Stack Window

Viewing and Analyzing the Test Results

While recording, editing, and running your script, WebLOAD Recorder provides information on all major events that occurred during runtime such as failures and error messages. You can navigate through the Execution Tree to view the results of your test at increasing levels of detail. This technique lets you view detailed information on any errors.

Using the Execution Tree to View Results

As you execute a script, WebLOAD Recorder displays the Web pages accessed in the Web application in the Execution Tree.

When working with a file JavaScript file that has not been converted to a WebLOAD Recorder project file, WebLOAD Recorder displays a playback node for each HTTP request of the JavaScript.

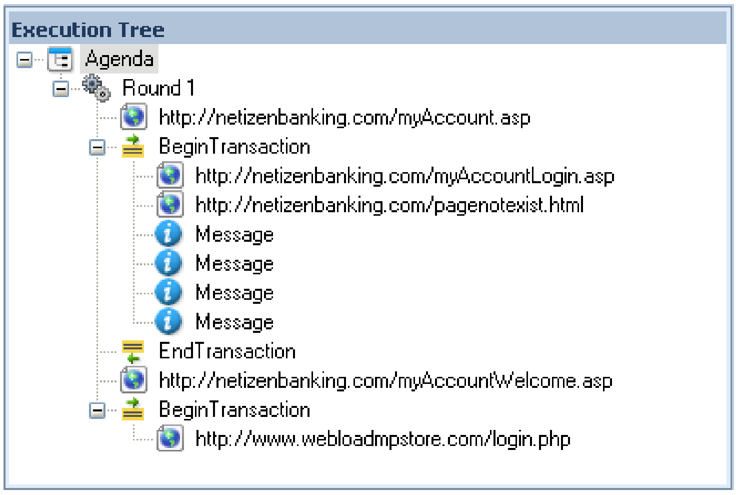

```
Execution Tree
Agenda
  Round 1
    http://netizenbanking.com/myAccount.asp
    BeginTransaction
      http://netizenbanking.com/myAccountLogin.asp
      http://netizenbanking.com/pagenotexist.html
      Message
      Message
      Message
      Message
    EndTransaction
    http://netizenbanking.com/myAccountWelcome.asp
    BeginTransaction
      http://www.webloadmpstore.com/login.php
```

Figure 81: Execution Tree View

Using the Page View to View Results

The Page View displays a visual representation of the baseline set of Web pages in your script. This view is available while recording, editing, or running your script.

To open the Page View:

1. Select the **Page View** checkbox in the **View** tab of the ribbon.
2. Select the **Page View** tab.

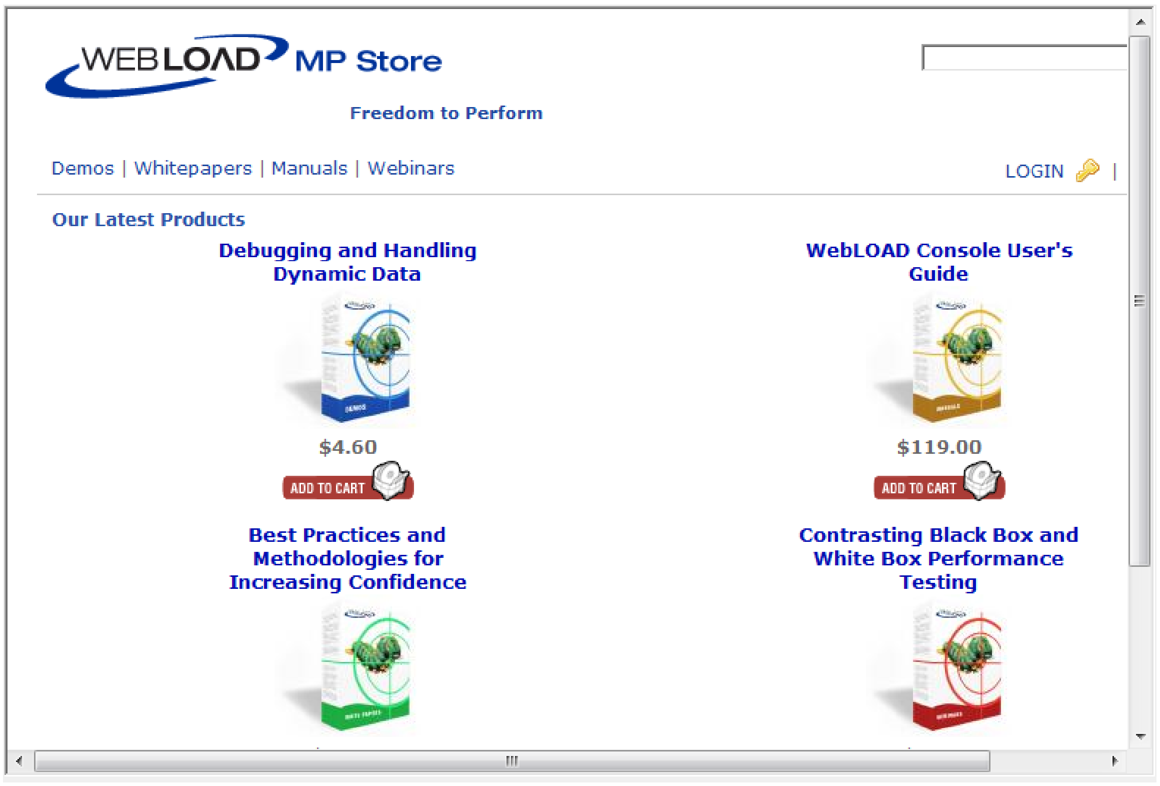

WEBLOAD MP Store
Freedom to Perform
Demos | Whitepapers | Manuals | Webinars
LOGIN |
Our Latest Products
Debugging and Handling Dynamic Data
$4.60
ADD TO CART
WebLOAD Console User's Guide
$119.00
ADD TO CART
Best Practices and Methodologies for Increasing Confidence
Contrasting Black Box and White Box Performance Testing

Figure 82: Page View

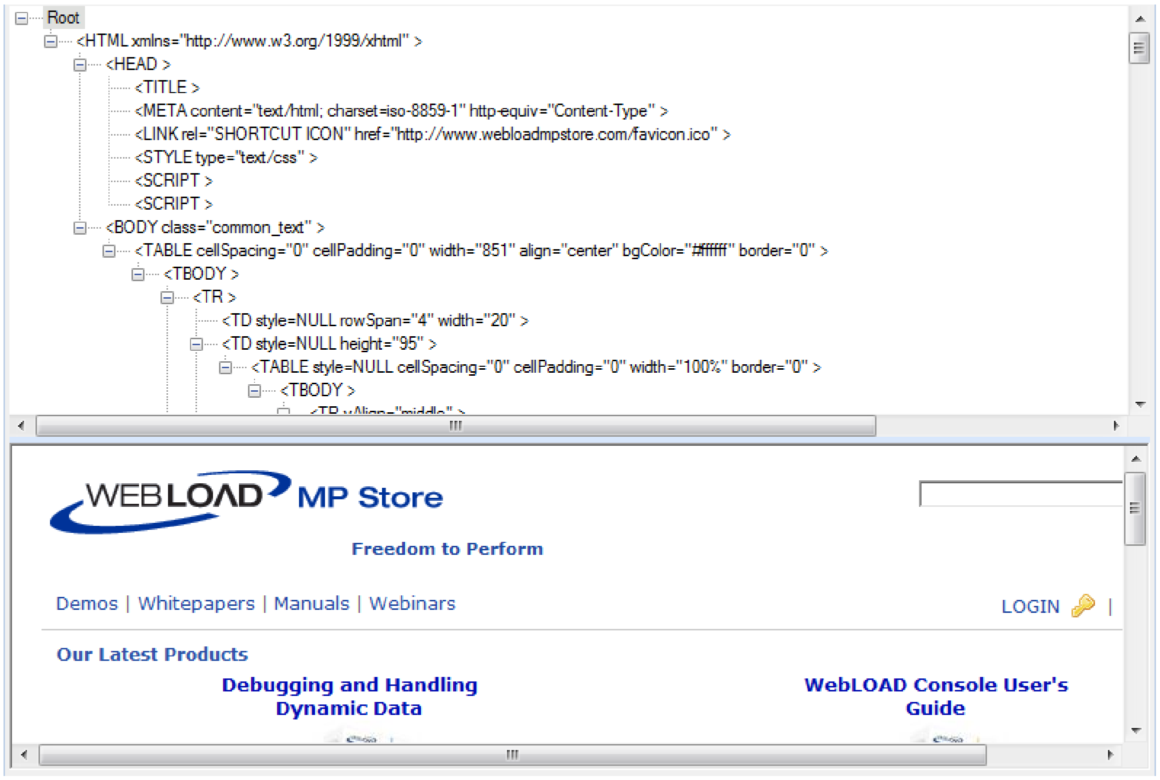
Using the DOM View to View Results

DOM View displays all of the objects and the structure of the Web page displayed in Page View, giving you access to objects not visible in the pages presentation layer.

DOM View is available when Page View is open, while recording, editing, or running your script. When an element is selected in the DOM View, the object is highlighted in the Page View.

To open the DOM View:

1. Select the **DOM View** checkbox in the **View** tab of the ribbon.
2. Select the **DOM View** tab.



```
Root
  <HTML xmlns="http://www.w3.org/1999/xhtml" >
    <HEAD >
      <TITLE >
      <META content="text/html; charset=iso-8859-1" http-equiv="Content-Type" >
      <LINK rel="SHORTCUT ICON" href="http://www.webloadmpstore.com/favicon.ico" >
      <STYLE type="text/css" >
      <SCRIPT >
      <SCRIPT >
    <BODY class="common_text" >
      <TABLE cellSpacing="0" cellPadding="0" width="851" align="center" bgColor="#ffffff" border="0" >
        <TBODY >
          <TR >
            <TD style=NULL rowSpan="4" width="20" >
            <TD style=NULL height="95" >
              <TABLE style=NULL cellSpacing="0" cellPadding="0" width="100%" border="0" >
                <TBODY >
```

WEBLOAD MP Store
Freedom to Perform
Demos | Whitepapers | Manuals | Webinars
LOGIN |
Our Latest Products
Debugging and Handling Dynamic Data
WebLOAD Console User's Guide

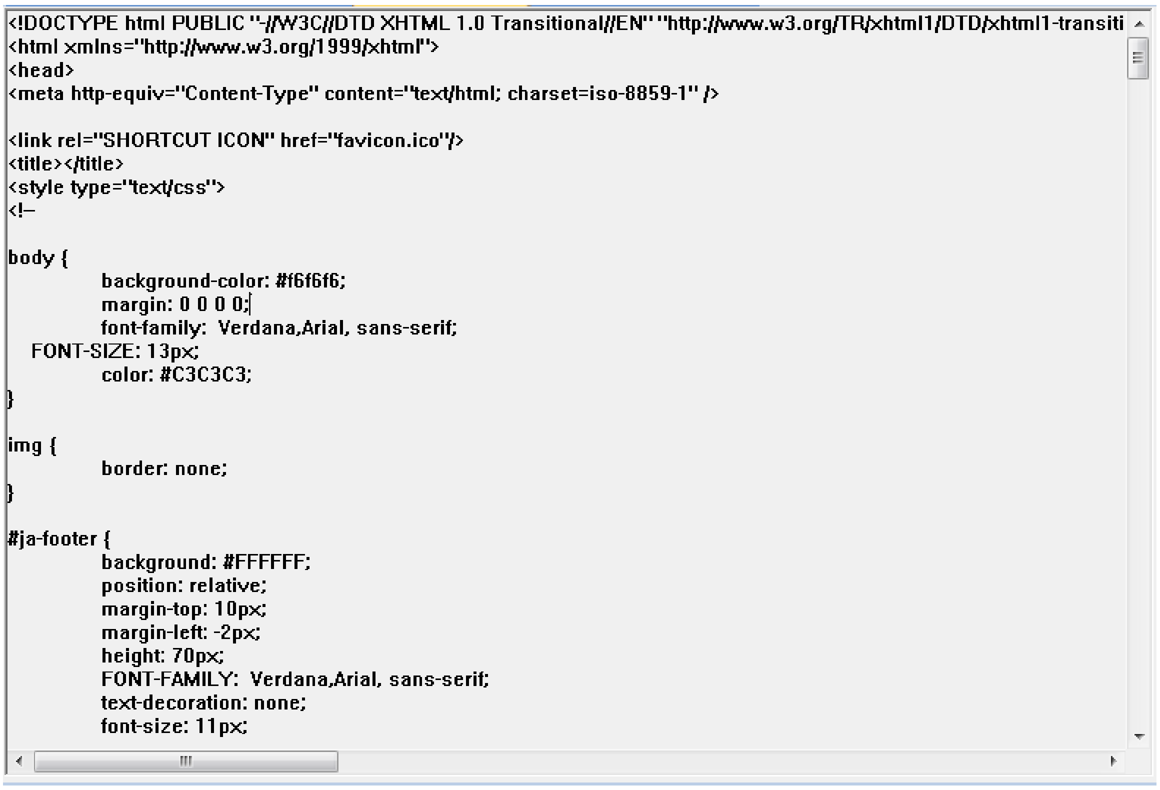
Figure 83: DOM View

Using the HTML View to View Results

HTML view displays an HTML preview of each page and frame requested in the script. When switching between the JavaScript, HTTP Headers, Browser, and HTML Views, the new view displays the node that is selected in the Script Tree (during edit mode) or Execution Tree (during debug mode). These views are available while recording, after the recording is finished, and after opening a saved script.

To open the HTML View:

1. Select the **HTML View** checkbox in the **View** tab of the ribbon.
2. Select the **HTML View** tab.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transiti
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<link rel="SHORTCUT ICON" href="favicon.ico"/>
<title></title>
<style type="text/css">
<!--

body {
            background-color: #f6f6f6;
            margin: 0 0 0 0;
            font-family:  Verdana,Arial, sans-serif;
   FONT-SIZE: 13px;
            color: #C3C3C3;
}

img {
            border: none;
}

#ja-footer {
            background: #FFFFFF;
            position: relative;
            margin-top: 10px;
            margin-left: -2px;
            height: 70px;
            FONT-FAMILY:  Verdana,Arial, sans-serif;
            text-decoration: none;
            font-size: 11px;
```

Figure 84: HTML View

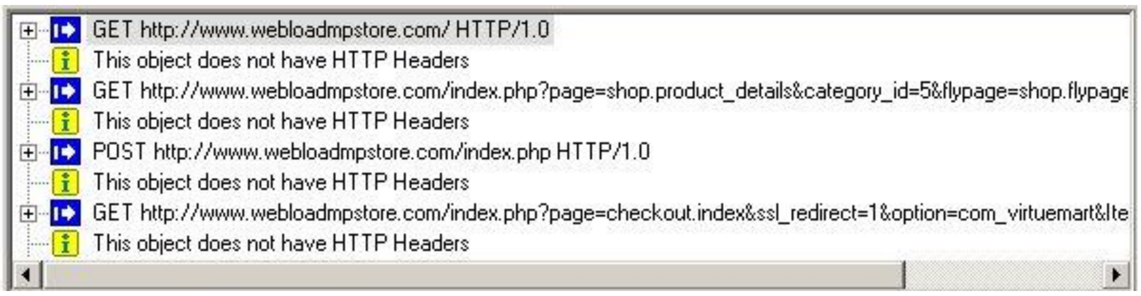
3. To search for text:
   a. Right-click and click **Find** from-the pop-up menu.
   b. Type the text you want to find, and click **Find Next**.
4. To copy text:
   a. Select the text you want to copy.
   b. Right-click and click **Copy** from-the pop-up menu.

Using the HTTP Headers View to View Results

The HTTP Headers View displays the GET and POST HTTP protocol commands. Other commands can also be displayed, such as CONNECT. When switching between the various views, the new view displays the node that is selected in the Script Tree. These views are available while recording, after the recording is finished, during playback and debugging, and after opening a saved script.

To open the HTTP Headers View:

1. Select the **HTTP Headers View** checkbox in the **View** tab of the ribbon.
2. Select the **HTTP Headers View** tab.

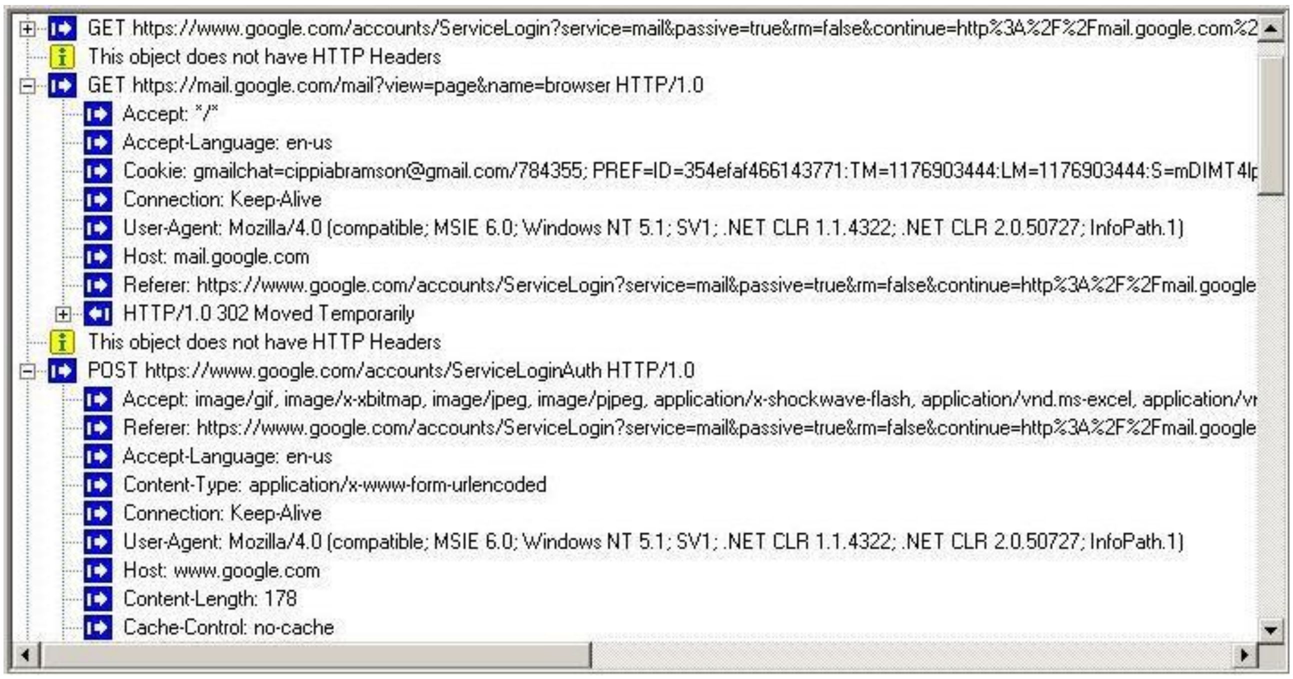


```
GET http://www.webloadmpstore.com/ HTTP/1.0
  This object does not have HTTP Headers
GET http://www.webloadmpstore.com/index.php?page=shop.product_details&category_id=5&flypage=shop.flypage
  This object does not have HTTP Headers
POST http://www.webloadmpstore.com/index.php HTTP/1.0
  This object does not have HTTP Headers
GET http://www.webloadmpstore.com/index.php?page=checkout.index&ssl_redirect=1&option=com_virtuemart&Ite
  This object does not have HTTP Headers
```

Figure 85: HTTP Headers View

The headers are divided into groups of headers per playback request. For each request, only the relevant headers are displayed.

You can expand the headers to show the form data and all other content.



```
GET https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&continue=http%3A%2F%2Fmail.google.com%2
  This object does not have HTTP Headers
GET https://mail.google.com/mail?view=page&name=browser HTTP/1.0
    Accept: */*
    Accept-Language: en-us
    Cookie: gmailchat=cippiabramson@gmail.com/784355; PREF=ID=354efaf466143771:TM=1176903444:LM=1176903444:S=mDIMT4lp
    Connection: Keep-Alive
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)
    Host: mail.google.com
    Referer: https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&continue=http%3A%2F%2Fmail.google
    HTTP/1.0 302 Moved Temporarily
  This object does not have HTTP Headers
POST https://www.google.com/accounts/ServiceLoginAuth HTTP/1.0
    Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vr
    Referer: https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&continue=http%3A%2F%2Fmail.google
    Accept-Language: en-us
    Content-Type: application/x-www-form-urlencoded
    Connection: Keep-Alive
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)
    Host: www.google.com
    Content-Length: 178
    Cache-Control: no-cache
```

Figure 86: HTTP Headers

3. To view all of the headers on the script, click the Agenda root node.
4. To view headers of a specific round, click the Round node in the Execution tree.
5. To search for text:
   a. Right-click and click **Find...** from-the pop-up menu.

b. In the **Find what** field, type the text you want to find.

   The **Find what** field is case sensitive.

c. Click **Find Next**.

   The entire text of the selected node is selected.

6. To copy text:

   a. Select the text you want to copy.

   b. Right-click and click **Copy** from-the pop-up menu.

      The entire text of the selected node is copied.

Using the Log View Window to View Results

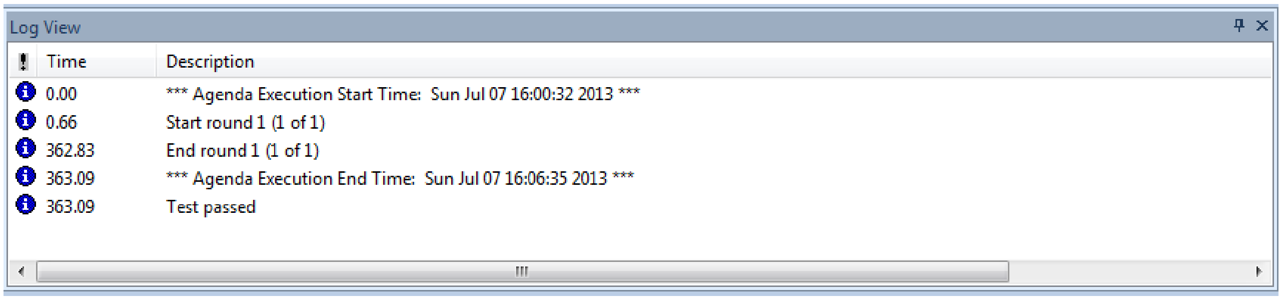
In addition to the results available through viewing the Script Tree and the Execution Tree, the Log View Window displays the errors encountered during playback and additional information about your test session results.

An Info Message or a minor error will not cause the playback to stop. Similarly, a generic message, issued when WebLOAD Recorder encounters HTTP errors that are undefined by WebLOAD, will not cause playback to stop. A higher level of severity (Error or Severe Error) ends the playback upon completion of the WebLOAD Recorder protocol block.

To open the Log View Window:

- Select the **Log View** checkbox in the **View** tab of the ribbon.

  By default, the Log View pane appears at the bottom of the main window.



Log View

!	Time	Description
ℹ	0.00	*** Agenda Execution Start Time: Sun Jul 07 16:00:32 2013 ***
ℹ	0.66	Start round 1 (1 of 1)
ℹ	362.83	End round 1 (1 of 1)
ℹ	363.09	*** Agenda Execution End Time: Sun Jul 07 16:06:35 2013 ***
ℹ	363.09	Test passed

Figure 87: Log View Pane

The following information is displayed:

- **!** – The result and severity of each message:
  - Information message
  - Minor error message

  - Error message
  - Severe error message
- **Time** – The amount of runtime.
- **Description** – The runtime action and information about failed actions, including the date and time the action occurred.

Performing a Full Search in the Test Results

You can search for a specific string in all views at once.

To search for text in all views:

1. Select **Find All** in the **Edit** tab of the ribbon.

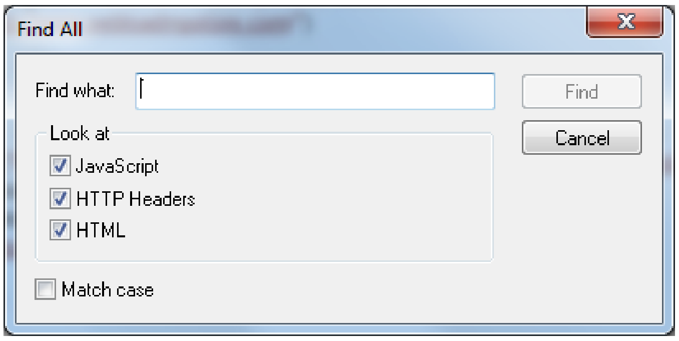
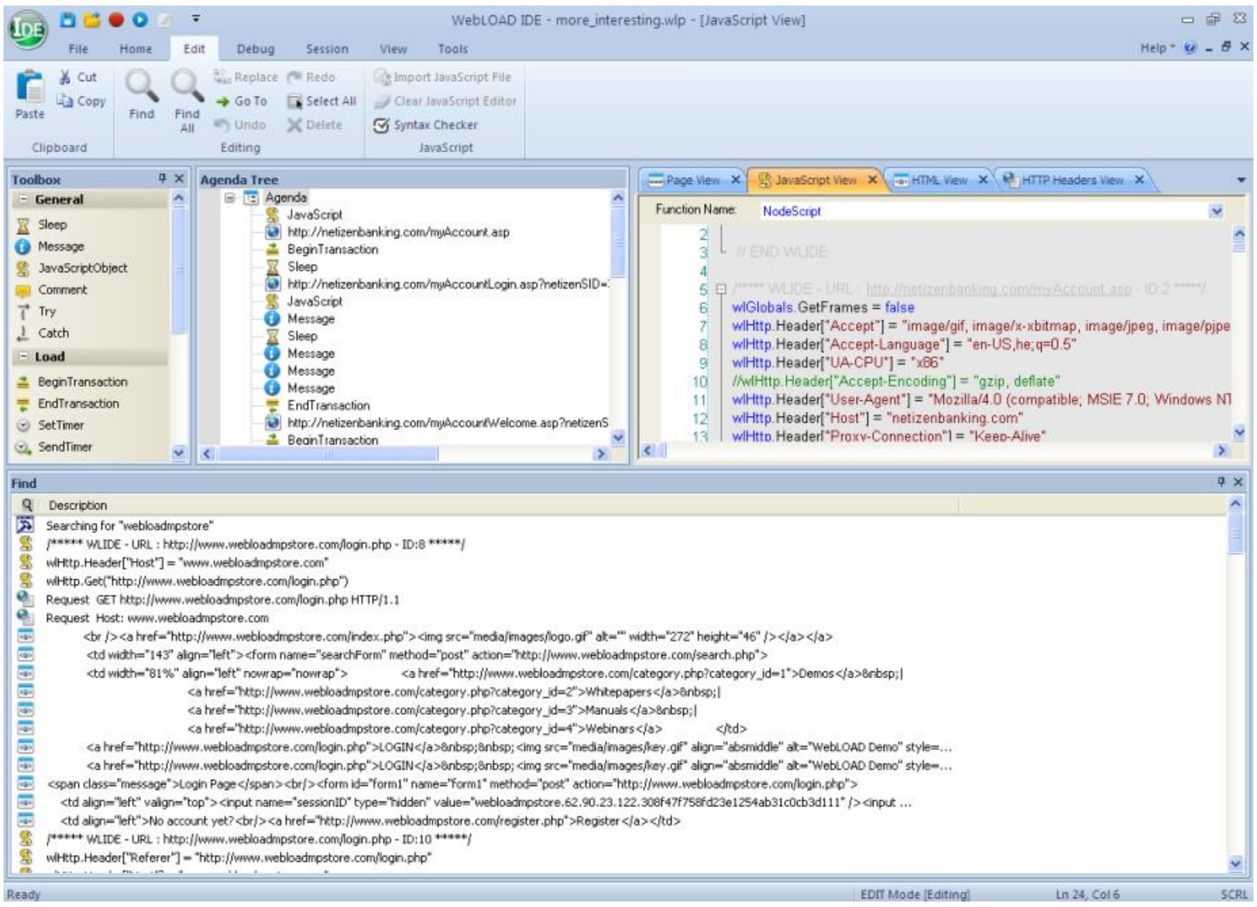


Figure 88: Find All Dialog Box

2. In the **Find what** field, type the text you want to find.
3. Under **Look at**, specify in which view you wish to search. You can select any combination of the following:
   - JavaScript
   - HTTP Headers
   - HTML
4. Optionally check the **Match case** checkbox.
5. Click **Find**.

   A results pane appears in the bottom half of the screen, displaying all the search results.



```
// END WLIDE

/***** WLIDE - URL : http://netizenbanking.com/myAccount.asp - ID:2 *****/
wlGlobals.GetFrames = false
wlHttp.Header["Accept"] = "image/gif, image/x-xbitmap, image/jpeg, image/pjpe
wlHttp.Header["Accept-Language"] = "en-US,he;q=0.5"
wlHttp.Header["UA-CPU"] = "x86"
//wlHttp.Header["Accept-Encoding"] = "gzip, deflate"
wlHttp.Header["User-Agent"] = "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
wlHttp.Header["Host"] = "netizenbanking.com"
wlHttp.Header["Proxy-Connection"] = "Keep-Alive"
```

```
Searching for "webloadmpstore"
/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:8 *****/
wlHttp.Header["Host"] = "www.webloadmpstore.com"
wlHttp.Get("http://www.webloadmpstore.com/login.php")
Request  GET http://www.webloadmpstore.com/login.php HTTP/1.1
Request  Host: www.webloadmpstore.com
        <br /><a href="http://www.webloadmpstore.com/index.php"><img src="media/images/logo.gif" alt="" width="272" height="46" /></a></a>
         <td width="143" align="left"><form name="searchForm" method="post" action="http://www.webloadmpstore.com/search.php">
         <td width="81%" align="left" nowrap="nowrap">            <a href="http://www.webloadmpstore.com/category.php?category_id=1">Demos</a>&nbsp;|
                          <a href="http://www.webloadmpstore.com/category.php?category_id=2">Whitepapers</a>&nbsp;|
                          <a href="http://www.webloadmpstore.com/category.php?category_id=3">Manuals</a>&nbsp;|
                          <a href="http://www.webloadmpstore.com/category.php?category_id=4">Webinars</a>            </td>
         <a href="http://www.webloadmpstore.com/login.php">LOGIN</a>&nbsp;&nbsp;<img src="media/images/key.gif" align="absmiddle" alt="WebLOAD Demo" style=...
         <a href="http://www.webloadmpstore.com/login.php">LOGIN</a>&nbsp;&nbsp;<img src="media/images/key.gif" align="absmiddle" alt="WebLOAD Demo" style=...
<span class="message">Login Page</span><br/><form id="form1" name="form1" method="post" action="http://www.webloadmpstore.com/login.php">
   <td align="left" valign="top"><input name="sessionID" type="hidden" value="webloadmpstore.62.90.23.122.308f47f758fd23e1254ab31c0cb3d111" /><input ...
   <td align="left">No account yet?<br/><a href="http://www.webloadmpstore.com/register.php">Register</a></td>
/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:10 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/login.php"
```

Figure 89: Search Results of the Find All Function

- The icon to the left of each search result indicates in which page view the result appears.
- Double clicking a search result highlights the result both in the Script Tree and in the corresponding page view.

Printing the Contents of the Log View Window

To print the contents of the Log View Window:

1. Right-click inside the Log View window.
2. Select **Print** from the right-click menu.

   The Print Setup dialog displays.

3. Select a printer and click **OK**.

Saving the Contents of the Log View Window

To save the contents of the Log View Window:

1. Right-click inside the Log View window.
2. Select **Save** from the right-click menu.

   The Save As dialog displays.

3. In the **File Name** field, type in the name for the file.
4. Click **Save**.

   The file is saved with the extension `*.log`.

   You can view the saved log file with any text editor.

Viewing a Log Message

To view the complete log message:

1. Right-click an entry in the Log View window.
2. Select **Display Message** from the right-click menu.

   The Log Message window with detailed information on the selected entry appears.

Validating Responses

WebLOAD Recorder enables you to validate a response in a script by adding a response validation function. You can validate a Web page's title, the maximum time taken to load the Web page, its content, and the length of its content. You can also determine WebLOAD's behavior if validation fails. During playback, the results of the validation process (failure or success) are displayed in the Log View window.

To add a response validation function:

1. Select a node in the Script Tree.
2. Click **Response Validation** in the **Home** tab of the ribbon.

   -Or-

   Right-click the node and select **Response Validation**.

   -Or-

   Perform the following:

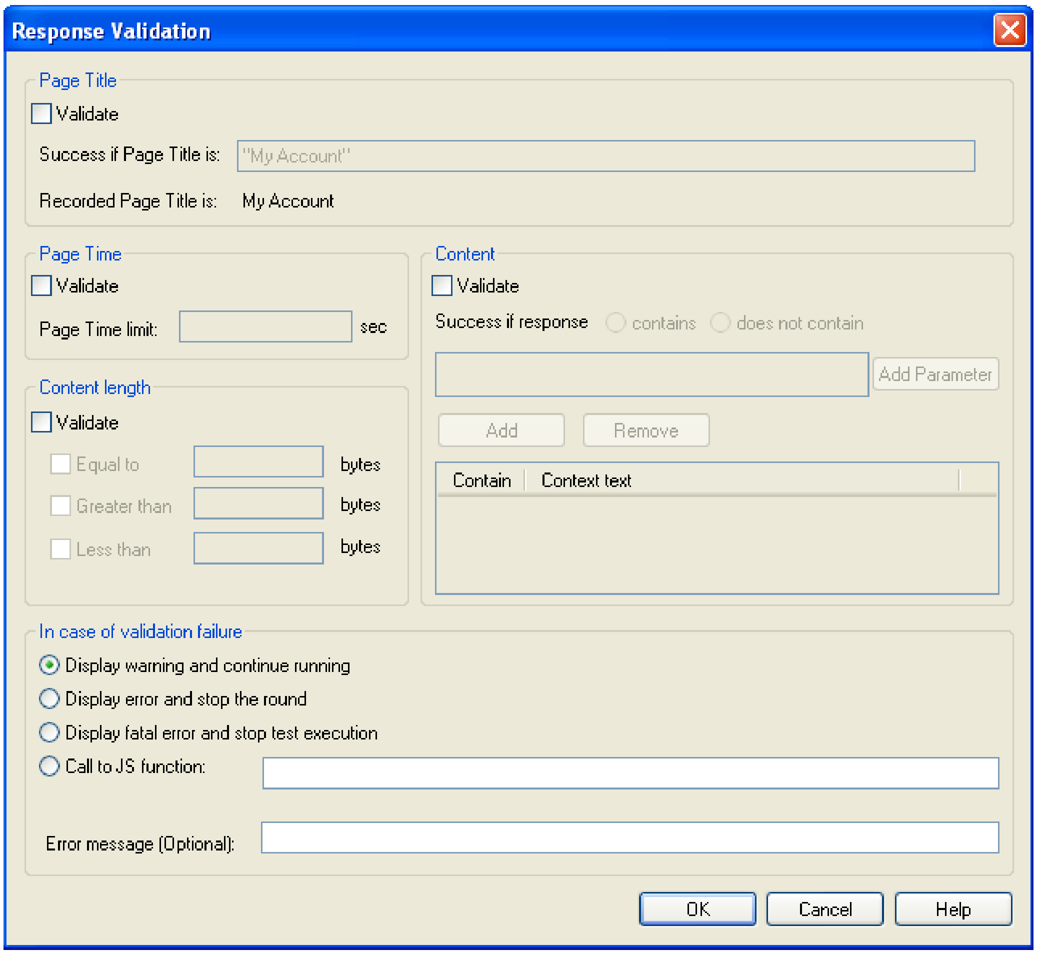
a. Click the **HTML View** tab to view the node in HTML View.

b. Select HTML text within the node.

c. Right-click the selection and click **Response Validation**.

The Response Validation dialog box appears.

Note: When accessing the Response Validation dialog box from **HTML View**, the dialog box appears automatically configured with the selected content.


Response Validation

Page Title
☐ Validate
Success if Page Title is: "My Account"
Recorded Page Title is: My Account

Page Time
☐ Validate
Page Time limit: sec

Content length
☐ Validate
☐ Equal to bytes
☐ Greater than bytes
☐ Less than bytes

Content
☐ Validate
Success if response ○ contains ○ does not contain

Contain	Context text

In case of validation failure
[✓ Display warning and continue running]
○ Display error and stop the round
○ Display fatal error and stop test execution
○ Call to JS function:
Error message (Optional):

Figure 90: Response Validation Dialog Box

3. Configure the responses you wish to validate during playback, according to the information displayed in Table 13, and click **OK**.

   The Response Validation function is added to your script.

Table 13: Response Validation Dialog Box Options

Field	Description
Page Title	
Validate	Select to validate the page title.
Success if Page Title is	The title of the Web page. During playback, if the title of the Web page matches the text entered in this field, the validation is successful.
Recorded page title is	The page title as defined in the HTML <title> tag.
Page Time	
Validate	Select to validate the page time.
Page Time limit x sec	The maximum number of seconds that may elapse while waiting for the Web page to open for the validation to be successful.
Content length	
Validate	Select to validate the content length.
Equal to x bytes	The size of the Web page content, in bytes, must equal the specified value for the validation to be successful.
Greater than x bytes	The size of the Web page content, in bytes, must be greater than the specified value for the validation to be successful.
Lower than x bytes	The size of the Web page content, in bytes, must be less than the specified value for the validation to be successful.
Recorded Content Length is	The size of the response, in bytes.
Content	
Validate	Select to validate the content. For a full explanation, refer to *Performing Multiple Text Validations of Web Page Content* (on page 142).
Success if response contains/does not contain x	For each JavaScript expression you include in your validation check, specify whether it must or must not appear in the Web page for the validation to be successful.
Add	Click this button to add a new JavaScript expression to the list of validations that must or must not appear in the Web page. The string "<text to find>" appears in the box above the button. Delete this string and instead do either or both of the following: • Enter a text string in quote marks. For example, "**Welcome**". • Enter a parameter without quote marks. For example, **TodaysDate()**. You can click **Add Parameter** and select a parameter from the list. Note that you can concatenate strings and/or parameters to create a JavaScript expression. For example: "**Welcome**" + **params_user.getValue()**.

Field	Description
Remove	Click this button to delete a selected JavaScript expression from the list of validations that must or must not appear in the Web page.
Add Parameter	Opens a list of parameters you can include in the **contains/does not contain** text. This list is identical to the list available in the Insert Variable menu (*Figure 54*).
In case of validation failure	
Display warning and continue running	Select to display a warning during playback and continue running the script, if the verification fails.
Display error and stop the round	Select to display an error during playback and stop the round, if the verification fails.
Display fatal error and stop test execution	Select to display a fatal error and stop running the script, if the verification fails.
Call to JS function	Select to run a specified JavaScript function, if the verification fails.
Error message (Optional)	Enter an error message to be displayed if the verification fails (optional).

4. Click **OK**. The Response Validation function is added to your script.

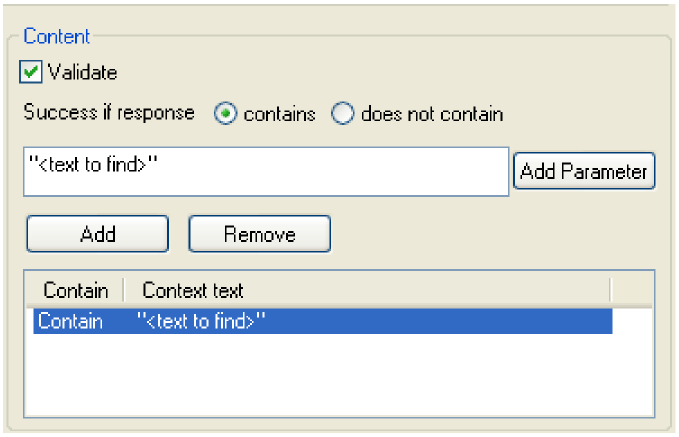
Performing Multiple Text Validations of Web Page Content

You can use the **Response Validation** feature to validate a Web page's content.

To validate the content of a Web page:

1. Follow the instructions in *Validating Responses* (on page 139) to access the Response Validation dialog box (Figure 90).
2. In the **Content** section, check the **Validate** checkbox.
3. Click **Add**.

   The box above the Add button displays **"<text to find>"**.



[✓] Validate

Success if response [✓ contains] does not contain

"<text to find>"

Contain	Context text
Contain	"<text to find>"

Figure 91: Defining Content Validation

4. Define a JavaScript expression and whether it must or must not appear in the Web page, as follows:

   a. Delete the string **"<text to find>"** and instead do either or both of the following:

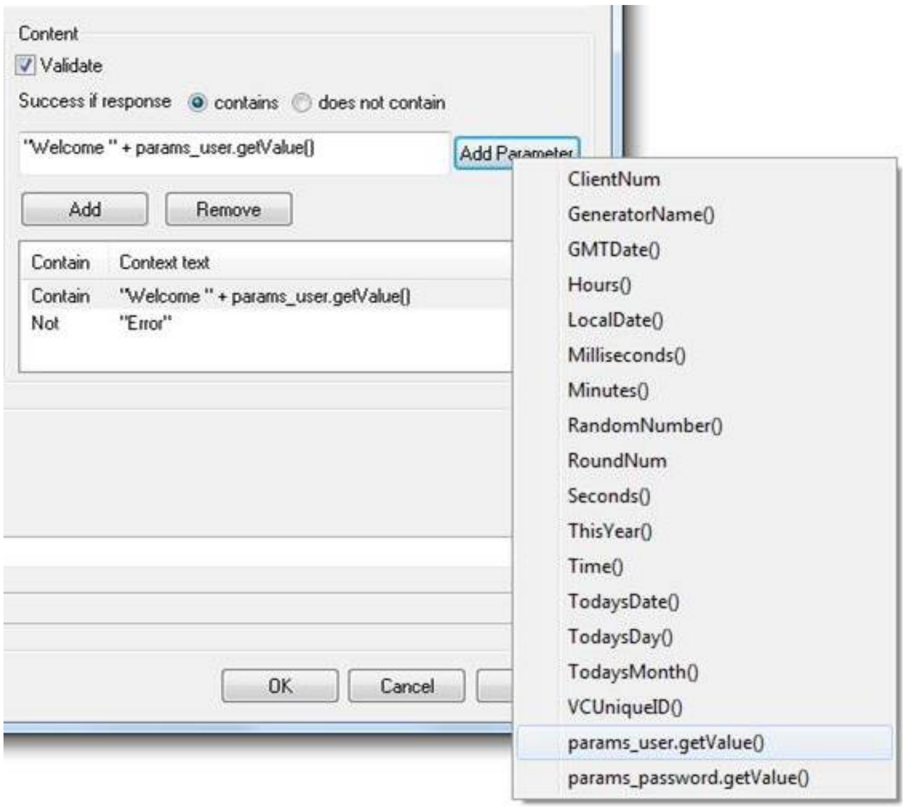
      - Enter a text string enclosed in quote marks. For example, "**Welcome**".
      - Enter a parameter without quote marks. For example, **TodaysDate()**. Alternatively, you can click **Add Parameter** and select a parameter from the list of predefined parameters.

      Note that you can concatenate strings and/or parameters to create a JavaScript expression. For example: "**Welcome**" **+ params_user.getValue()**.

   b. Select **contains** if the expression must appear in the Web page; select **does not contain** if it should not appear in the Web page.

5. Repeat the previous step for every additional expression you wish to define.

Figure 92 shows a content validation example. In this example, the page content will be validated only if it contains the string `Welcome` followed by a user name, and does not contain the string `Error`.



Contain	Context text
Contain	"Welcome " + params_user.getValue()
Not	"Error"

Figure 92: Content Validation Example

Editing a Script for Dynamic HTML Pages

When you record an HTML page in the WebLOAD Recorder, there can be dynamic values that WebLOAD adds to the script, which are recorded in the WebLOAD Recorder. Such dynamic values can contain state management information, such as the session-id, which is usually passed as URL encoded parameters or hidden form fields. The dynamic values that are recorded in the WebLOAD Recorder are different during each run. Since the value that was recorded in the WebLOAD Recorder and the dynamic value do not match, you will receive an error.

To overcome this situation, you need to edit the script and perform correlation. This can be done manually, or by using WebLOAD's Smart Copy feature. This feature enables you to convert the dynamic value into the correct value for the specific session.

Note: The Smart Copy feature supports converting the dynamic value of the following HTML objects: images, links, and form elements.

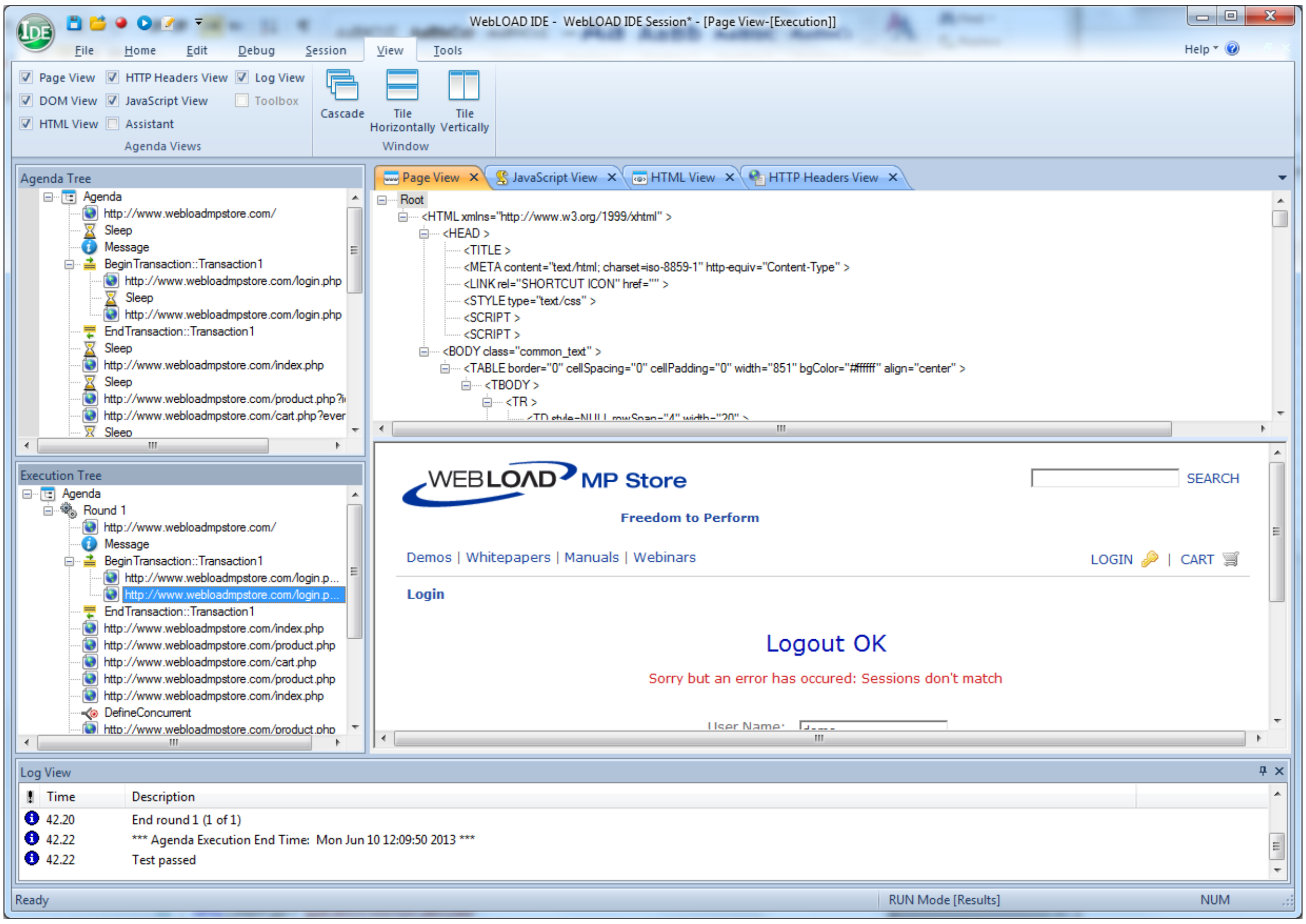
Note: Editing the script and performing correlation is not necessary for static HTML pages, since they do not contain dynamic values. In this case, the script executes smoothly with no need for initial editing.

To edit a script using Smart Copy:

1. After recording and running your script, open the Page View with the DOM View. Select the **DOM View** checkbox in the **View** tab of the ribbon.

   The Page and DOM View appear.

2. In the Execution Tree, select the first node.
3. In the Page View, search for an error message. If there is no message, select the next node in the Execution Tree and search for a message there.



Agenda Tree:

```
Agenda
  http://www.webloadmpstore.com/
  Sleep
  Message
  BeginTransaction::Transaction1
    http://www.webloadmpstore.com/login.php
    Sleep
    http://www.webloadmpstore.com/login.php
  EndTransaction::Transaction1
  Sleep
  http://www.webloadmpstore.com/index.php
  Sleep
  http://www.webloadmpstore.com/product.php?i
  http://www.webloadmpstore.com/cart.php?ever
  Sleep
```

Execution Tree:

```
Agenda
  Round 1
    http://www.webloadmpstore.com/
    Message
    BeginTransaction::Transaction1
      http://www.webloadmpstore.com/login.p...
      http://www.webloadmpstore.com/login.p...
    EndTransaction::Transaction1
    http://www.webloadmpstore.com/index.php
    http://www.webloadmpstore.com/product.php
    http://www.webloadmpstore.com/cart.php
    http://www.webloadmpstore.com/product.php
    http://www.webloadmpstore.com/index.php
    DefineConcurrent
    http://www.webloadmpstore.com/product.php
```

DOM View:

```
Root
  <HTML xmlns="http://www.w3.org/1999/xhtml" >
    <HEAD >
      <TITLE >
      <META content="text/html; charset=iso-8859-1" http-equiv="Content-Type" >
      <LINK rel="SHORTCUT ICON" href="" >
      <STYLE type="text/css" >
      <SCRIPT >
      <SCRIPT >
    <BODY class="common_text" >
      <TABLE border="0" cellSpacing="0" cellPadding="0" width="851" bgColor="#ffffff" align="center" >
        <TBODY >
          <TR >
```

Page View:

WEBLOAD MP Store SEARCH
Freedom to Perform
Demos | Whitepapers | Manuals | Webinars LOGIN | CART
Login
Logout OK
Sorry but an error has occured: Sessions don't match
User Name:

Log View:

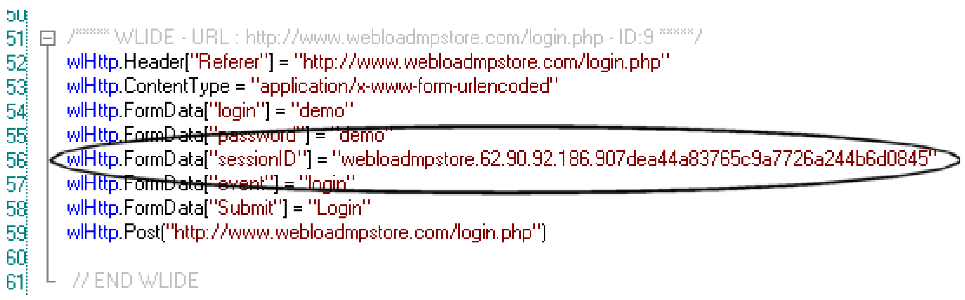
Time	Description
42.20	End round 1 (1 of 1)
42.22	*** Agenda Execution End Time: Mon Jun 10 12:09:50 2013 ***
42.22	Test passed

Figure 93: Page View Displaying Error Message

4. Once you locate the message, open the JavaScript View. Select the **JavaScript View** checkbox in the **View** tab of the ribbon.

   The JavaScript View appears with the requested block of code selected.

5. Within the selected block of code, locate the dynamic value (for example, the session-id field). This field must be retrieved from the previous block of code.

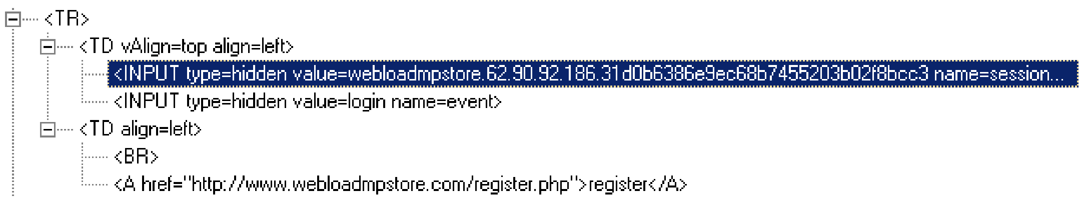


```
/***** WLIDE - URL : http://www.webloadmpstore.com/login.php - ID:9 *****/
wlHttp.Header["Referer"] = "http://www.webloadmpstore.com/login.php"
wlHttp.ContentType = "application/x-www-form-urlencoded"
wlHttp.FormData["login"] = "demo"
wlHttp.FormData["password"] = "demo"
wlHttp.FormData["sessionID"] = "webloadmpstore.62.90.92.186.907dea44a83765c9a7726a244b6d0845"
wlHttp.FormData["event"] = "login"
wlHttp.FormData["Submit"] = "Login"
wlHttp.Post("http://www.webloadmpstore.com/login.php")

// END WLIDE
```

Figure 94: Dynamic Value in the JavaScript View

6. Click the previous node in the Execution Tree to search for the element that contains the dynamic value. Make sure the Browser and DOM Views are open. Select the **Page View** and **DOM View** checkboxes in the **View** tab of the ribbon.

7. In the DOM View, locate the element that contains the dynamic value. This is usually a hidden input field.

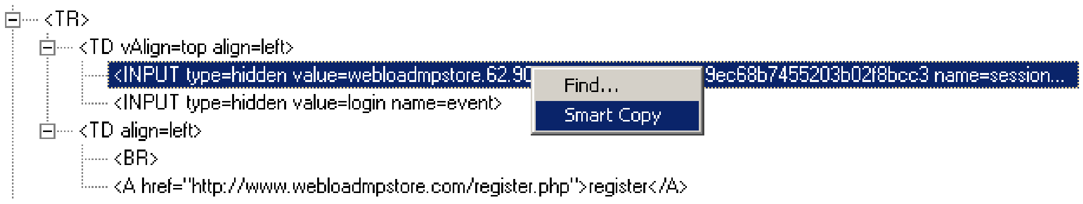


```
<TR>
  <TD vAlign=top align=left>
    <INPUT type=hidden value=webloadmpstore.62.90.92.186.31d0b6386e9ec68b7455203b02f8bcc3 name=session...
    <INPUT type=hidden value=login name=event>
  <TD align=left>
    <BR>
    <A href="http://www.webloadmpstore.com/register.php">register</A>
```

Figure 95: Dynamic Value in DOM View

Note: You cannot use the value recorded in the script, since the value that was recorded was dynamic, and will not match the new value that is given when you run the script.

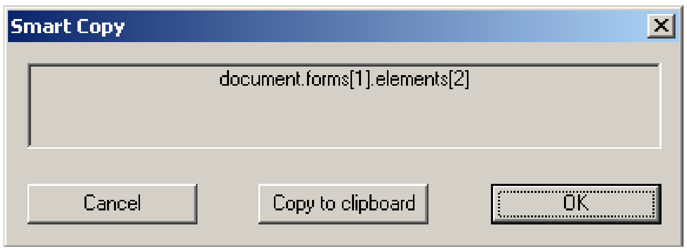
8. Right-click the element and select **Smart Copy** from the pop-up menu.



```
<TR>
  <TD vAlign=top align=left>
    <INPUT type=hidden value=webloadmpstore.62.90          9ec68b7455203b02f8bcc3 name=session...
    <INPUT type=hidden value=login name=event>
  <TD align=left>
    <BR>
    <A href="http://www.webloadmpstore.com/register.php">register</A>
```

Figure 96: Smart Copy Pop-up Menu

The Smart Copy dialog box appears.



document.forms[1].elements[2]

Figure 97: Smart Copy Dialog Box

9. Click **Copy to clipboard** and click **OK**.
10. To edit the JavaScript, click **Full Script** in the **Home** tab of the ribbon.
11. Create a variable for the dynamic field by typing the following at the end of the selected block of code:

    ```
    Session_id =
    ```

12. Paste the clipboard text (using **Paste** in the **Edit** tab of the ribbon) after the equal sign.

    For example:

    ```
    Session_id = document.forms[1].elements[2].value
    ```

13. In the subsequent block of code, replace

    ```
    wlHttp.FormData[“session_id”] = <static session id>
    ```

    with

    ```
    wlHttp.FormData[“session_id”] = session_id
    ```

    The script is edited. You can now run the script successfully without receiving error messages.

Chapter 9

Working with Git

Git is a free and open source distributed version control system for tracking changes in computer files and coordinating work on those files among multiple people.

WebLOAD supports basic Git operations, with easy access to Git operations from within WebLOAD Recorder and WebLOAD Console. WebLOAD assumes that you have downloaded and installed Git, and are familiar with the way it works.

Prerequisites to working with Git

As prerequisites to working with Git, you need to:

1. Specify once in WebLOAD how to access the Git Local and Remote Repositories. Refer to *Setting up Access to Git Local and Remote Repositories* (on page 149).
2. Make sure your WebLOAD files are saved in the directory defined as the Git local repository. This can be done in one of two ways:
   - Define the Git local repository directory to be the WebLOAD default directory.
   - Change the WebLOAD default directory to be the directory of the Git local repository.

   To do either of the above, you need to access the window where you can view or change the location of the WebLOAD default directory:

   - In the WebLOAD Console, navigate to **Global Options** > **File Locations**.
   - In the WebLOAD Recorder, navigate to **Settings** > **File Locations**.

   Note that changing the WebLOAD default directory in either the Console UI or the WebLOAD Recorder UI, changes it for both.

Setting up Access to Git Local and Remote Repositories

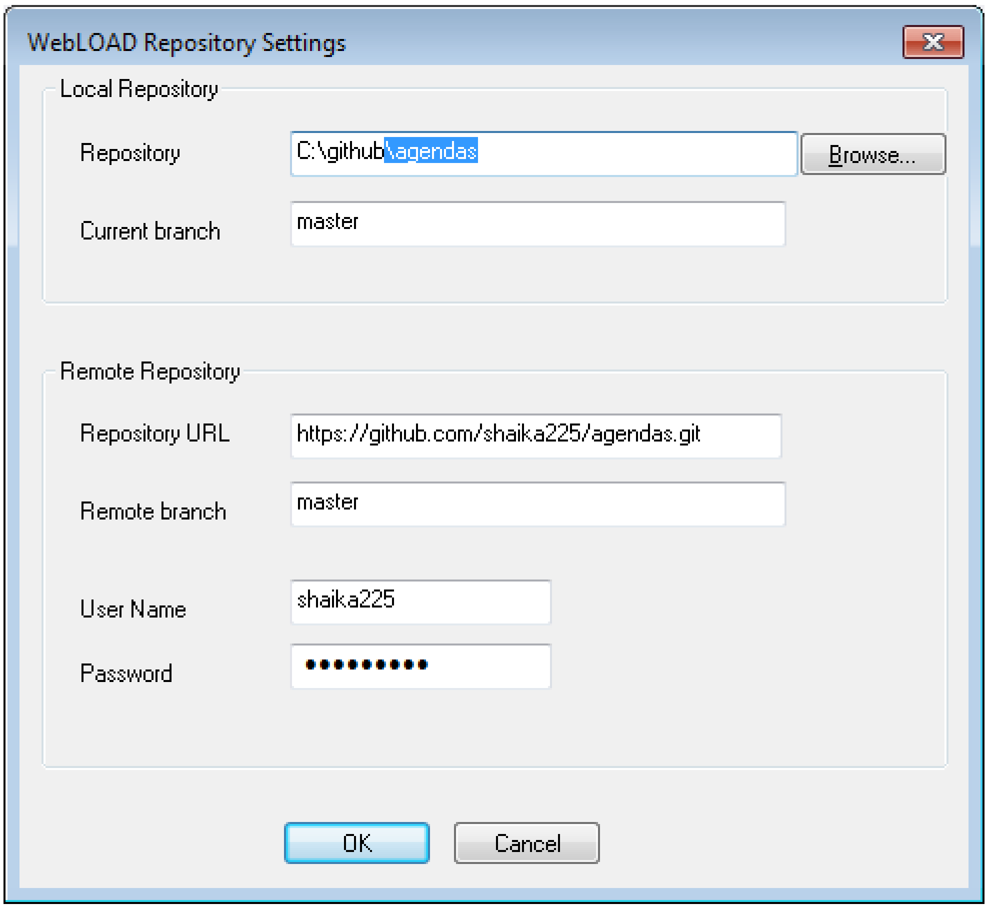
As a prerequisite to using Git operation from within WebLOAD, you need to once specify in WebLOAD how to access the Git Local Repository and Remote Repository.

Note: You can perform this prerequisite task either from the WebLOAD Recorder, or from the WebLOAD Console. Once you perform it in one of them, Git operations will be accessible from both.

To specify the Git local and remote repositories:

1. Click **Repository Settings** in the **Tools** tab of the ribbon.

   The WebLOAD Repository Settings window appears.



WebLOAD Repository Settings

Local Repository

Repository: C:\github\agendas

Current branch: master

Remote Repository

Repository URL: https://github.com/shaika225/agendas.git

Remote branch: master

User Name: shaika225

Password: ●●●●●●●●●●

Figure 98: WebLOAD Repository Settings

2. In the **Local Repository** section:
   1. Specify the location of the local repository.
   2. Specify the **current branch**.
3. In the **Remote Repository** section:
   1. Specify the location of the remote repository (either a URL or a directory).
   2. Specify the **remote branch**.
   3. Specify the **User Name** and **Password** for accessing the remote repository.
4. Click **OK**.

Performing a Commit action

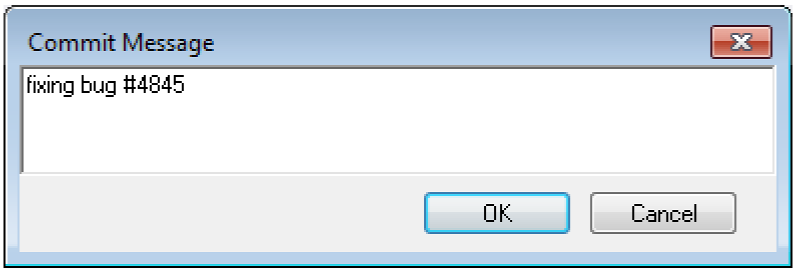
If you setup WebLOAD to support Git, you can perform a Commit, that is, save to the Git local repository, any WebLOAD objects you are currently working on. This functionality is available in both the WebLOAD Recorder and the WebLOAD Console.

- When you select Commit in the WebLOAD Recorder, the script you are currently working on as well as all its additional information, are saved to the Git local directory.
- When you select Commit in the WebLOAD Console, the test definition of the test you are currently working on, as well as the session created by test execution, are saved to the Git local directory

To Commit a WebLOAD object:

1. Click **Commit** in the **Tools** tab of the ribbon.

   A window appears in which you can enter a Commit comment.



fixing bug #4845

Figure 99: Entering a Commit Comment

2. Optionally enter a comment, and click **OK**.

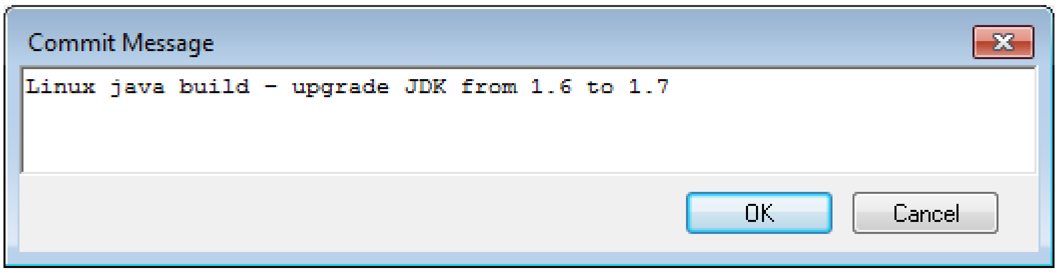
Performing a Commit Dir action

If you setup WebLOAD to support Git, you perform a Commit dir, that is, save to the Git local repository an entire folder (with all its descendants). This folder must be located under the local repository folder. This functionality is available in both the WebLOAD Recorder and the WebLOAD Console.

To perform a Commit dir:

1. Click **Commit dir** in the **Tools** tab of the ribbon.
2. In the File Explorer window that appears, specify the directory you wish to commit.

   A window appears in which you can enter a Commit dir comment.



```
Linux java build - upgrade JDK from 1.6 to 1.7
```

Figure 100: Entering a Commit Dir Comment

3. Optionally enter a comment, and click **OK**.

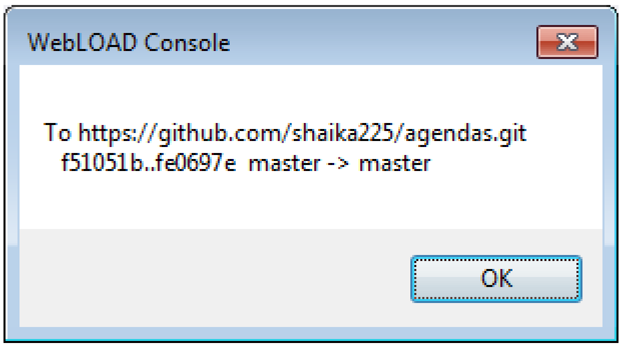
Performing a Push action

If you setup WebLOAD to support Git, you can perform a Push, which causes all Commits you had made to the Git local repository, to be pushed to the Git remote repository. This functionality is available in both the WebLOAD Recorder and the WebLOAD Console.

To perform a Push:

1. Click **Push** in the **Tools** tab of the ribbon.

   A window appears, informing you of the Git Push operation and its success status.



```
To https://github.com/shaika225/agendas.git
  f51051b..fe0697e  master -> master
```

Figure 101: Git Push Operation Message

2. Click **OK**.
3. If the Push operation is rejected, for example because a conflict has arisen due to changes made by two different users, a window appears, displaying the Git rejection message. Follow the instructions in the message, and resolve the rejection using your chosen Git UI.

Performing a Pull action

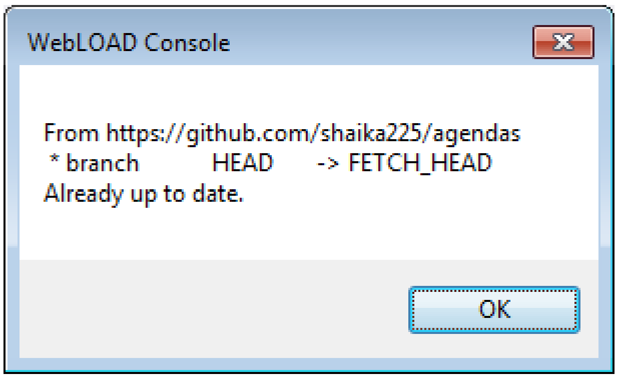
If you setup WebLOAD to support Git, you can perform a Pull, which pulls data from the remote repository to the local repository. This functionality is available in both the WebLOAD Recorder and the WebLOAD Console.

A Pull operation pulls the entire branch from the remote repository and updates the local repository and directories.

To perform a Pull:

1. Click **Pull** in the **Tools** tab of the ribbon.

   A window appears, informing you of the Git Pull operation and its success status.



```
From https://github.com/shaika225/agendas
 * branch         HEAD      -> FETCH_HEAD
Already up to date.
```

Figure 102: Git Pull Operation Message

2. Click **OK**.

Opening the Git UI

If you setup WebLOAD to support Git, you can launch your Git UI from within WebLOAD. This functionality is available in both the WebLOAD Recorder and the WebLOAD Console.

To launch the Git UI:

1. Click **Open Git Gui** in the **Tools** tab of the ribbon.

The Git UI is launched.

Chapter 10

Configuring the WebLOAD Recorder Options

You can set the following WebLOAD Recorder configuration options:

- **Default Project Options** – Settings for WebLOAD Recorder that will be in effect for each script you create. These options are for the playback.
- **Current Project Options** – Settings that will override the Default Project Options settings.
- **Recording and Script Generation Options** – Settings that define the behavior of WebLOAD Recorder during the recording and script generation of a Web session.
- **Settings** – Settings for WebLOAD Recorder.
- **Customize** – Settings for the toolbar.
- **Parameterization Manager** – Settings for replacing a recorded static value in a script with a random value from a pool of values, or with a whole set of values from a file.

Configuring the Default and Current Project Options

The Project Options are settings for WebLOAD Recorder that will be in effect for each script you create.

- **Default Project Options** are settings that will be in effect for each script you create. Each script created or edited in WebLOAD Recorder is automatically assigned these defaults. You can modify these settings to your specifications.
- **Current Project Options** are settings that will override the Default Project Options settings and affect only the currently open script. You can modify these settings to your specifications.

Notes:
The Current Project Options dialog boxes are the same as the Default Project Options dialog boxes *except* for the title.

You must be in Edit mode to modify the options.

Opening the Default and Current Project Options

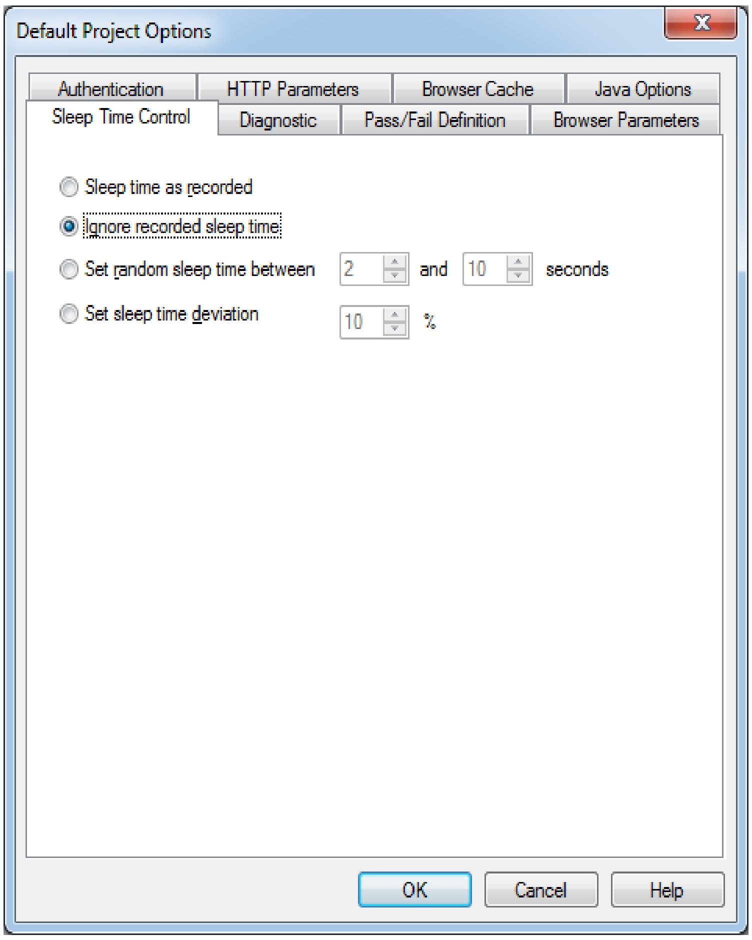
To open the Default Project Options dialog box:

- Click **Default Project Options** in the **Tools** tab of the ribbon,

  -Or-

  Select **Default Project Options** from the WebLOAD Recorder **Home** tab of the ribbon.

  The Default Project Options dialog box opens with the Sleep Time Control tab displayed.



Sleep time as recorded [✓ Ignore recorded sleep time]

Set random sleep time between 2 and 10 seconds

Set sleep time deviation 10 %

Figure 103: Default Project Option Dialog Box

To open the Current Project Options dialog box:

- Click **Current Project Options** in the **Tools** tab of the ribbon,

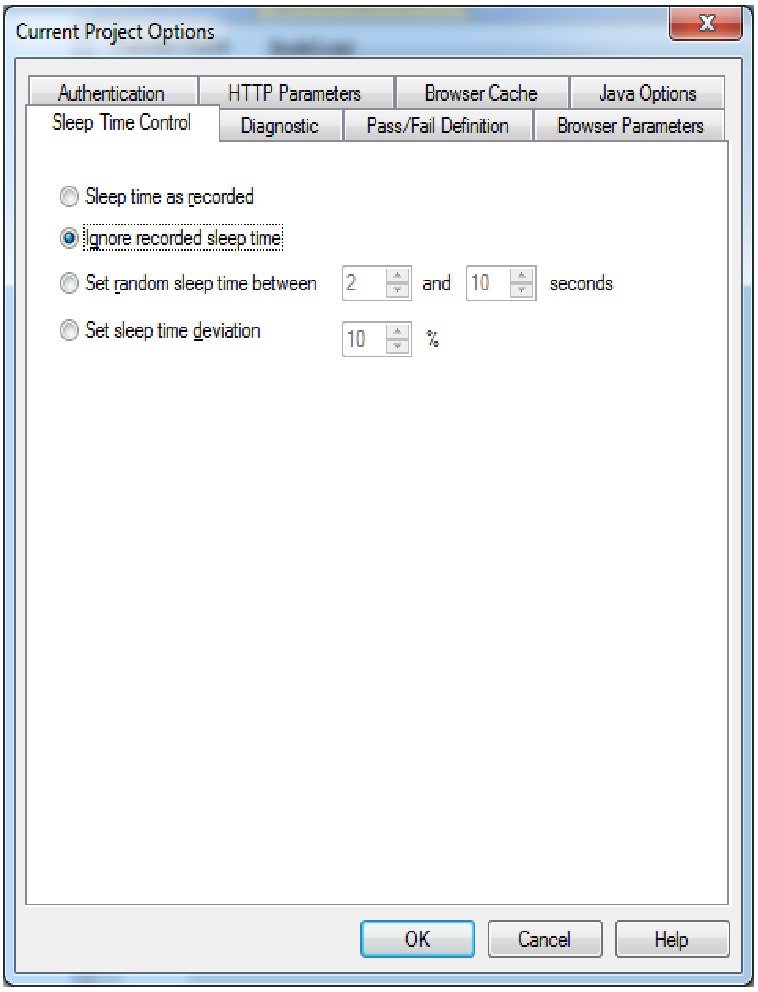
  -Or-

Select **Current Project Options** from the WebLOAD Recorder **Home** tab of the ribbon,

-Or-

Right-click the Agenda root node in the Script Tree and select **Current Project Options**.

The Current Project Options dialog box opens with the Sleep Time Control tab displayed.



Sleep time as recorded [✓ Ignore recorded sleep time]

Set random sleep time between 2 and 10 seconds

Set sleep time deviation 10 %

Figure 104: Current Project Options Dialog Box

The following table describes the tabs in the Default and Current Project Options dialog box.

Table 14: Default and Current Project Options Dialog Box Tabs

Tab	Description
Sleep Time Control (default)	Define the behavior of Sleep time during script playback and debug. Sleep time is a pause to simulate the intermittent activity of real users.

Tab	Description
Pass/Fail Definition	Define the test failure criteria in WebLOAD Recorder.
Browser Parameters	Define the Virtual Client behavior.
Authentication	Define the Global and Proxy authentication settings.
HTTP Parameters	Define the HTTP Client behavior on the HTTP protocol level.
Browser Cache	Define the type of cache and when the cache is cleared.
Diagnostic	Define the Diagnostic options that can be enabled while developing a script or for tracking problems in existing scripts.
Java Options	Define the Virtual Machine to be used.

Setting Pass/Fail Definitions

Use the Pass/Fail Definition options to define test failure criteria in WebLOAD Recorder.

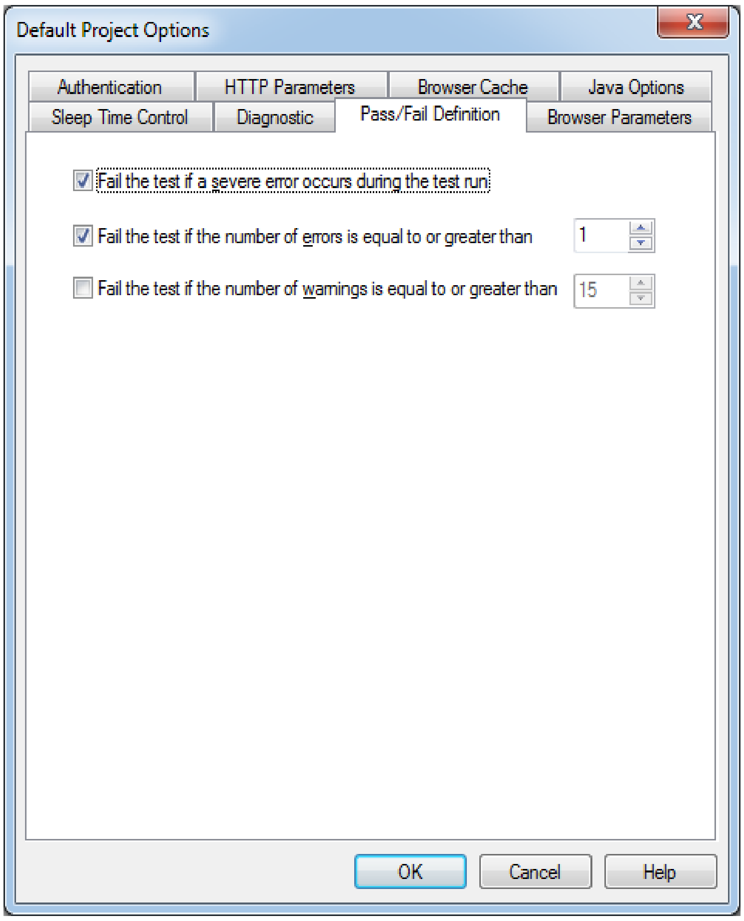
To set the Pass/Fail Definition options:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.

   The Default/Current Project Options dialog box opens.

2. Select the **Pass/Fail Definition** tab.

   The Pass/Fail Definition tab moves to the front of the dialog box.



☑ Fail the test if a severe error occurs during the test run

☑ Fail the test if the number of errors is equal to or greater than 1

☐ Fail the test if the number of warnings is equal to or greater than 15

Figure 105: Pass/Fail Definition Tab

3. Set test failure criteria. By default, WebLOAD Recorder will fail a test if a severe error occurs during the test run. You can also set WebLOAD Recorder to fail the test if a set numbers of errors or warnings are encountered.
4. Click **OK**.

Configuring Sleep Time Control Options

Sleep time is a pause to simulate the intermittent activity of real users. WebLOAD Recorder enables you to control the sleep time during run-time and set a script to execute with the sleep times recorded in the script, random sleep times, or no sleep times.

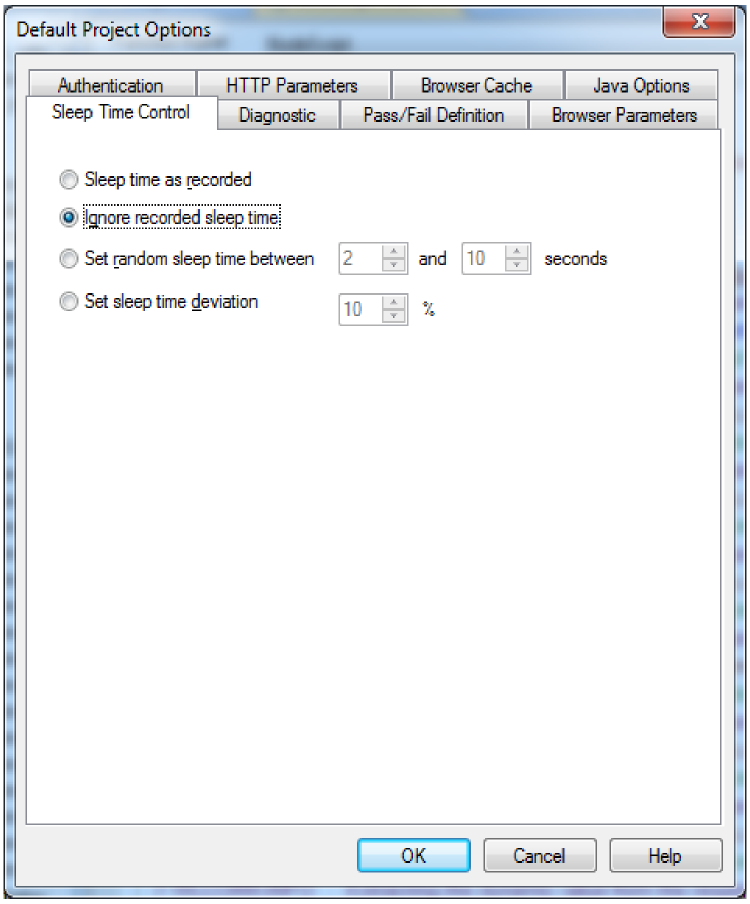
To configure Sleep Time Control options:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.

   The Default/Current Project Options dialog box opens.

2. Select the **Sleep Time Control** (default) tab.

   The Sleep Time Control tab moves to the front of the dialog box.



( ) Sleep time as recorded
(●) Ignore recorded sleep time
( ) Set random sleep time between 2 and 10 seconds
( ) Set sleep time deviation 10 %

Figure 106: Sleep Time Control Tab

3. Specify the type of sleep to use when playing the script.

   There are four options:

   - Select **Sleep time as recorded** to run the script with the delays corresponding to the natural pauses that occurred when recording the script.
   - Select **Ignore recorded sleep time** (default) to eliminate any pauses when running the script and run a worst-case stress test.
   - Select **Set random sleep time** and set the range of delays to represent a range of users.
   - Select **Set sleep time deviation** and set the percentage of deviate from the recorded value to represent a range of users.

4. Click **OK**.

Setting the Browser Parameters

The Browser Parameters option enables you to define Virtual Client behavior.

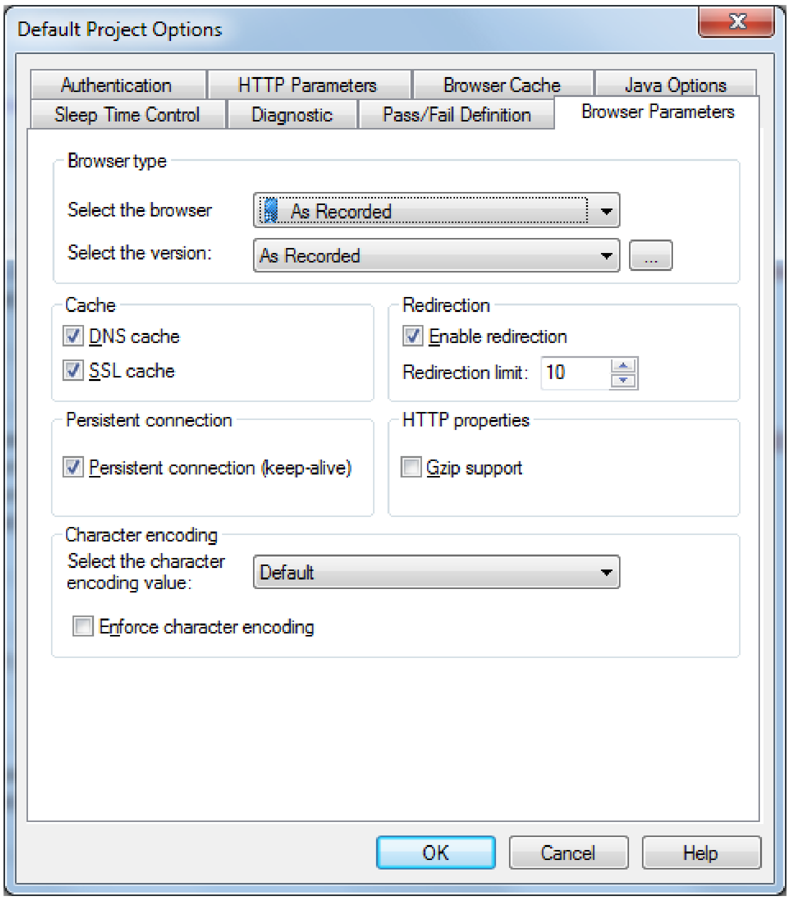
To set the Browser Parameters options:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.

   The Default/Current Project Options dialog box opens.

2. Select the **Browser Parameters** tab.

   The Browser Parameters tab moves to the front of the dialog box.



Browser type

Select the browser: As Recorded

Select the version: As Recorded

Cache

- [✓] DNS cache
- [✓] SSL cache

Redirection

- [✓] Enable redirection

Redirection limit: 10

Persistent connection

- [✓] Persistent connection (keep-alive)

HTTP properties

- [ ] Gzip support

Character encoding

Select the character encoding value: Default

- [ ] Enforce character encoding

Figure 107: Browser Parameters Tab of Default/Current Project Options Dialog Box

3. To set the Browser and Version:

   a. Select the browser from the **Select the browser** drop-down list. An appropriate version automatically appears in the **Select the version** field.

   b. You can select an alternative version from the drop-down list, or click the Change button ... to edit the version definition (see *Editing Browser Version* Definitions (on page 164)).

4. To simulate specific cache behaviors, select the **DNS cache** checkbox and **SSL cache** checkbox.
5. To set Redirection limits:
   a. Select the **Enable redirection** checkbox.
   b. In the Redirection limit field, type or select the desired redirection limit. The default limit is 10.
6. To enable a persistent connection to the server, select the **Persistent connection** checkbox.
7. To set Gzip, select the **Gzip support** checkbox.
8. To set character encoding:
   a. Select a character encoding type from the drop-down list.
   b. To enforce character encoding, select the **Enforce character encoding** checkbox.
9. Click **OK**.

The following table describes the fields and buttons in the Browser Parameters tab.

Table 15: Browser Parameters Tab Fields and Buttons

Field	Description
Browser Type	The browser type and user-agent setting specify the type of browser the Virtual Clients should emulate. By default, all Virtual Clients use the WebLOAD Recorder default browser agent.
Select the browser	You can set WebLOAD Recorder to emulate any of the standard browsers.
Select the user-agent	You can specify any specific application by supplying a custom user-agent that is included in all HTTP headers.
Cache	The types of cache to simulate.
DNS Cache	Caches the IP addresses received from the domain name server, reducing the domain name resolution time. Disable DNS caching if you want to include the time for domain name resolution in the WebLOAD performance statistics.
SSL Cache	Caches the SSL decoding keys received from an SSL server, so that WebLOAD Recorder only receives the key on the first SSL connection in each round. In subsequent connections, WebLOAD Recorder retrieves the key from cache. Enabling SSL Cache also reduces transmission time during SSL communication. Disable SSL caching if you want to measure the transmission time of the decoding key in the WebLOAD performance statistics for each SSL connection.

Field	Description
Redirection	
Enable redirection	Enables the redirection.
Redirection limit	Sets the redirection limit.
Persistent Connection	
Persistent connection (keep-alive)	When enabled, WebLOAD Recorder keeps an HTTP connection alive between successive accesses in the same round of the main script. Keeping a connection alive saves time between accesses. WebLOAD attempts to keep the connection alive unless you switch to a different server. However, some HTTP servers may refuse to keep a connection alive. You should not keep a connection alive if establishing the connection is part of the performance test.
HTTP properties	
Gzip support	Sets the `wlGlobals.AcceptEncodingGzip` flag. When this flag is set, WebLOAD Recorder behaves as follows: 1. For each request, sends the header "Accept-Encoding: gzip, deflate". This informs the server that the client can accept zipped content. 2. When this header is turned on, the server MAY send a response with the header "content-encoding: gzip" or "content-encoding: deflate". If either of these headers is sent, it means that the response is zipped/deflated and WebLOAD Recorder will unzip/inflate the content. **Note:** Most servers will work correctly even if the client does not send the "Accept-Encoding: gzip, deflate" header. Therefore, unless needed, it is recommended not to set the `wlGlobals.AcceptEncodingGzip` flag because it is performance heavy. However, some servers will fail if it is not sent. Microsoft Internet Explorer/Mozilla sends it by default. You can manually code the script to send the "Accept-Encoding: gzip, deflate" header even if the `wlGlobals.AcceptEncodingGzip` flag is not set. In this case, if the server returns zipped content, WebLOAD Recorder will not unzip it.
Character Encoding	
Select the character encoding value	Contains the value corresponding to the character set being used. The default value is Default (0), the regional settings of the computer.
Enforce character encoding	Indicates whether the parser should use the character set it parses in the HTML pages or override it using the value specified in the Select Character Encoding drop-down list. The default value is `false` (use the encoding from the HTML pages).

Editing Browser Version Definitions

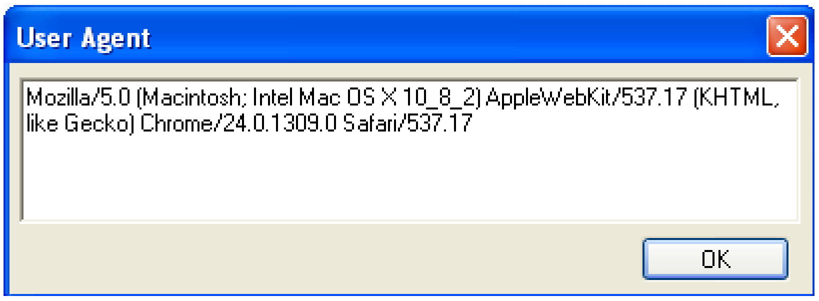
The available browser version list is appropriate for the browser type you select. You can add to the browser version list.

Note: If you are working in the Current Project options dialog box, adding a browser version to the list only affects the current session. When you restart the application, the original browser version list is used. If you are working in the Default Project options dialog box, the updated browser version list is saved for future sessions as well.

To add a browser version:

1. Click the browse button in the **Browser Type** area on the Browser Parameters tab.

   The User Agent dialog box opens.



Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/537.17 (KHTML, like Gecko) Chrome/24.0.1309.0 Safari/537.17

Figure 108: User Agent Dialog Box

2. Manually edit the version definition.
3. Click **OK**.

Setting the HTTP Parameters

The HTTP Parameters option enables you to define HTTP client behavior on the HTTP protocol level.

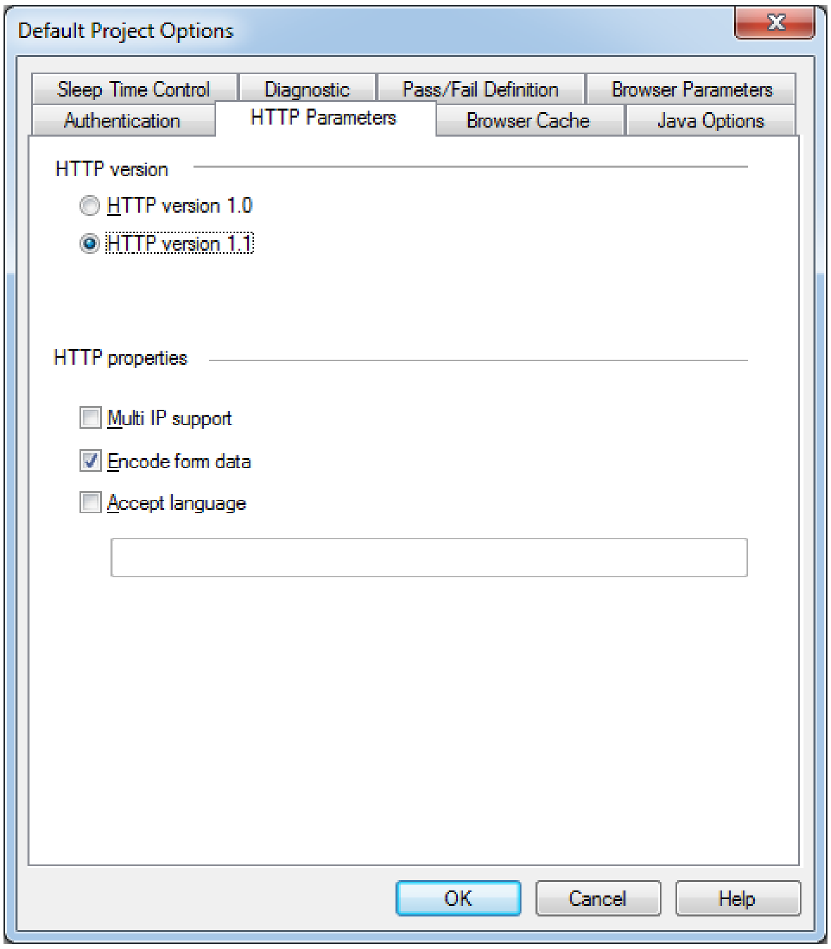
To set the HTTP Parameters options:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.

   The Default/Current Project Options dialog box opens.

2. Select the **HTTP Parameters** tab.

   The HTTP Parameters tab appears at the front of the dialog box.


Default Project Options

Sleep Time Control | Diagnostic | Pass/Fail Definition | Browser Parameters | Authentication | HTTP Parameters | Browser Cache | Java Options

HTTP version

- ( ) HTTP version 1.0
- (●) HTTP version 1.1

HTTP properties

- [ ] Multi IP support
- [✓] Encode form data
- [ ] Accept language

Figure 109: HTTP Parameters Dialog Box

3. Set the HTTP version by clicking **HTTP version 1.0** or **HTTP version 1.1**.
4. Select one or more **HTTP properties** checkboxes.
5. Click **OK**.

The following table describes the fields and buttons in the HTTP Parameters dialog box.

Table 16: HTTP Parameters Dialog Box Fields and Buttons

Field	**Description**
HTTP version	The appropriate HTTP protocol version (for example "HTTP/1.1").
HTTP version 1.0	Sets the HTTP protocol version to 1.0.
HTTP version 1.1	Sets the HTTP protocol version to 1.1.

Field	Description
HTTP properties	

Multi IP support	Sets the `wlGlobals.MultiIPSupport` flag to indicate that the HTTP protocol version supports more than one IP address.
	
	If this option is selected, WebLOAD takes all IP addresses defined on the Load Generator machine. However, you can exclude specific IP addresses by modifying the WebLOAD.ini configuration file.
	
	To exclude certain IP addresses, open the WebLOAD.ini file in `<RadView directory>\bin` and locate the following line:
	
	`EXCLUDED_IPS=" "`
	
	Enter the IP addresses you wish to exclude. If you enter multiple addresses, use a pipe delimiter between addresses as in the following example:
	
	`EXCLUDED_IPS="127.0.0.1|192.168.113.16|10.254.8.88"`

Encode form data	Sets the `wlGlobals.EncodeFormdata` flag. In general, when an HTTP client (Microsoft Internet Explorer/Firefox or WebLOAD Recorder) sends a post request to the server, the data must be HTTP encoded. Special characters like blanks, ">" signs, and so on, are replaced by "%xx". For example, space is encoded as "%20". This encoding can be performance heavy for large data, so WebLOAD Recorder enables you to turn it off. This should ONLY be done if you are sure that the data does not contain special characters. If it does, and the customer wants to improve performance via this flag, then you should replace the special characters within the script or use `wlHttp.EncodeFormdata` to set the flag for specific requests.
Accept language	Sets the `wlGlobals.AcceptLanguage` flag. This flag defines a global value for the "Accept-Language" header which will be sent with each request. Some applications/servers will behave differently depending on the setting. It is a simple string and WebLOAD Recorder does not enforce any checks on the values. It is similar to the property in the sense that it is a `wlGlobals/wlHttp` setting that affects the value of request headers.

Setting the Browser Cache

WebLOAD Recorder enables you to define the behavior of the cache that WebLOAD Console uses in order to simulate the behavior of a browser's cache. WebLOAD can cache JavaScript files, style sheets, images, applets, and XML files.

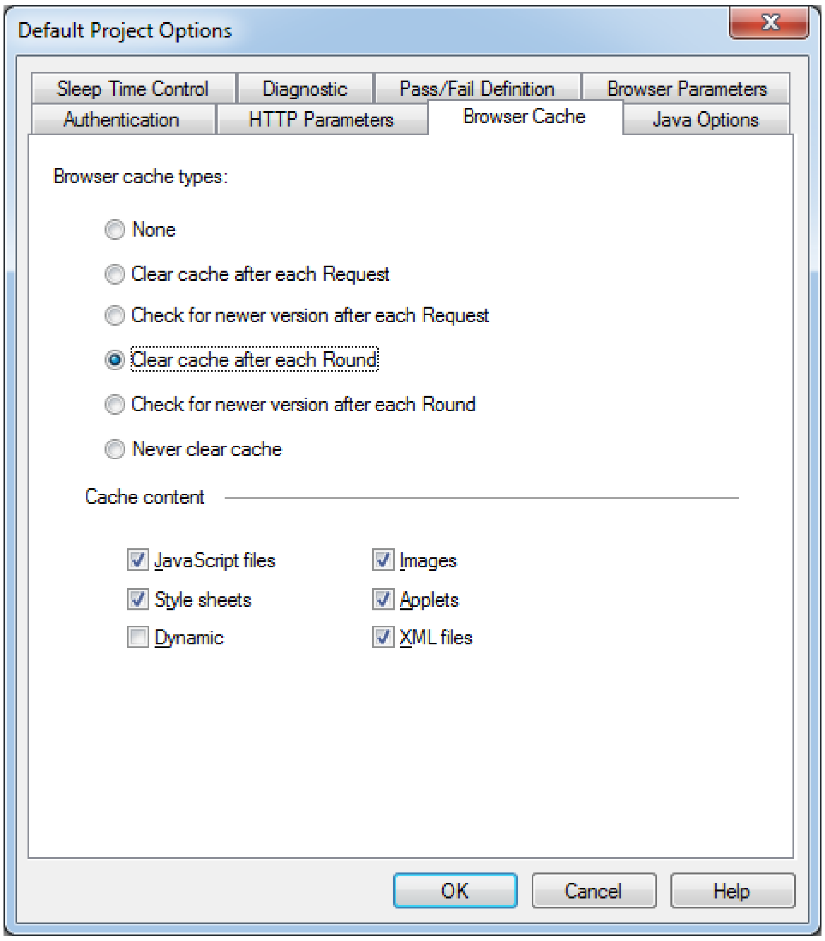
To define the browser cache behavior:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.

   The Default/Current Project Options dialog box opens.

2. Select the **Browser Cache** tab.

   The Browser Cache tab moves to the front of the dialog box.



Browser cache types:

- ○ None
- ○ Clear cache after each Request
- ○ Check for newer version after each Request
- ● Clear cache after each Round
- ○ Check for newer version after each Round
- ○ Never clear cache

Cache content

- ☑ JavaScript files
- ☑ Style sheets
- ☐ Dynamic
- ☑ Images
- ☑ Applets
- ☑ XML files

Figure 110: Browser Cache Tab

The following table describes the fields in the Browser Cache tab.

Table 17: Browser Cache Tab Fields

Field	Description
Browser cache types	
None	All Virtual Clients simulate a browser with no available cache.
Clear cache after each Request	Defines that the cache will be cleared after each request.

Field	Description
Check for newer version after each Request	Defines that WebLOAD will check for a newer version of the cached item after every request. Whenever the engine has a request for a cached resource, the engine sends the request with an “if-modified-since” header. If the server responds with a 200 status, the engine will refresh the cache.
Clear cache after each Round	Defines that the cache will be cleared after each script execution round. This is the default setting.
Check for newer version after each Round	Defines that WebLOAD will check for a newer version of the cached item after each round. Whenever the engine has a request for a cached resource, the engine sends the request with an “if-modified-since” header. If the server responds with a 200 status, the engine will refresh the cache.
Never clear cache	Defines that the cache will never be cleared. Each client maintains its own cache.
Cache content	You can select a filter, enabling you to indicate specific content types to be cached in the script. The available filters are: • JavaScript files • Style sheets • Images • Applets • XML files • Dynamic

Configuring Authentication Settings

WebLOAD Recorder enables you to define the global and proxy authentication settings.

WebLOAD Recorder enables you to configure a double proxy configuration, which instructs the recorder to use two application proxies, one for regular HTTP traffic and another for secure (SSL) traffic. To configure the two proxies, see *Configuring a Double Proxy* (on page 203).

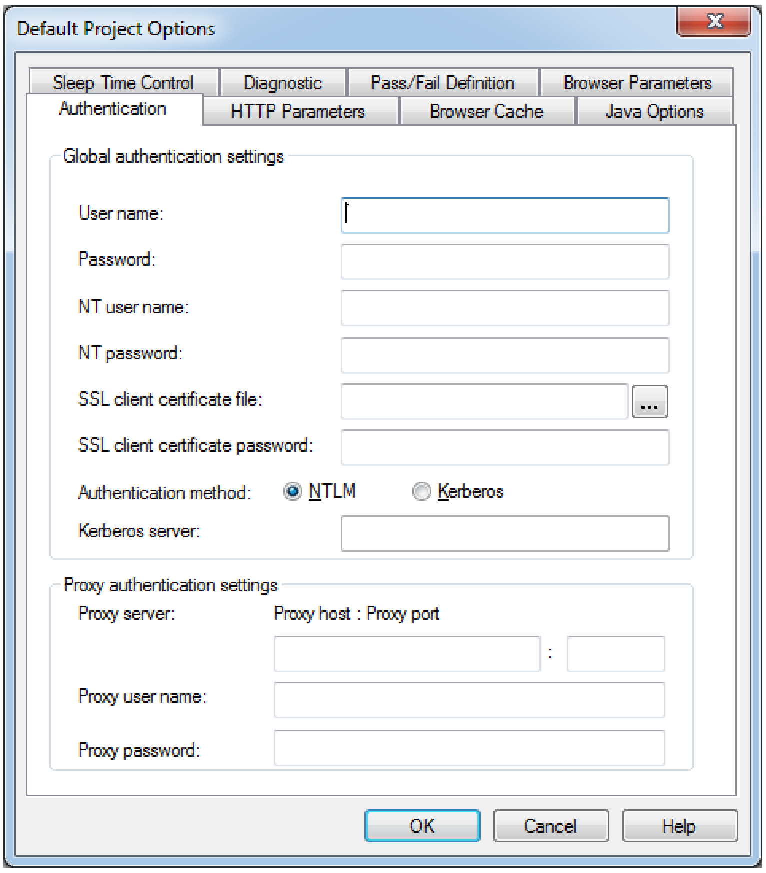
To configure Authentication settings:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.

   The Default/Current Project Options dialog box appears.

2. Select the **Authentication** tab.

The Authentication tab moves to the front of the dialog box.



Global authentication settings

User name:

Password:

NT user name:

NT password:

SSL client certificate file:

SSL client certificate password:

Authentication method: [✓ NTLM] Kerberos

Kerberos server:

Proxy authentication settings

Proxy server: Proxy host : Proxy port

Proxy user name:

Proxy password:

Figure 111: Authentication Tab

3. Fill in the fields, as described in Table 18.
4. Click **OK**.

The following table defines all the fields and options in the Authentication tab.

Table 18: Authentications Tab Fields and Options

Field	**Description**
Global Authentication Settings	
User name and **Password**	The user name and password that the script should use to log onto restricted HTTP sites.
NT user name and **NT password**	The user name the script should use for Windows NT Challenge response authentication.
SSL client certificate file and **SSL client certificate password**	The file name (optionally including a directory path) of the certificate WebLOAD makes available to SSL servers and type the password. Click **Browse** to search for the file.

Field	Description
Authentication method	The authentication method supported by the server: • NTLM (default). • Kerberos.
Kerberos server	The Kerberos server Fully Qualified Domain Name (FQDN). For example, specify "qa4" rather than "qa4.radview.co.il". This field is only enabled when the authentication method is set to Kerberos.
Proxy authentication settings	
Proxy server: **Proxy host** and **Proxy port**	The host name and port number for the proxy server.
Proxy user name and **Proxy password**	The user name and password for the proxy server.

Setting Diagnostic Options

Diagnostic options can be enabled while developing a script or for tracking problems in existing scripts.

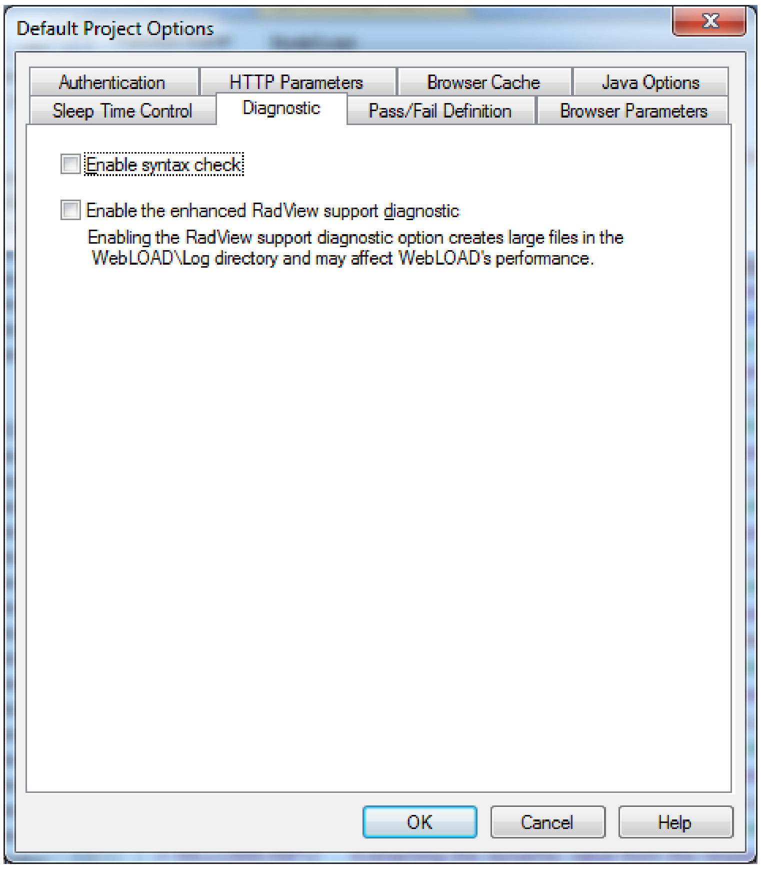
To set Diagnostic options:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.

   The Default/Current Project Options dialog box opens.

2. Select the **Diagnostic** tab.

   The Diagnostic tab moves to the front of the dialog box.


Default Project Options

Authentication | HTTP Parameters | Browser Cache | Java Options | Sleep Time Control | Diagnostic | Pass/Fail Definition | Browser Parameters

☐ Enable syntax check

☐ Enable the enhanced RadView support diagnostic

Enabling the RadView support diagnostic option creates large files in the WebLOAD\Log directory and may affect WebLOAD's performance.

Figure 112: Diagnostic Tab

3. Set the **Enable syntax check** option, see *Enabling Syntax Checking* (on page 171).
4. Set the **Enable the enhanced RadView support diagnostic** option, see *Enabling RadView Support Diagnostic* (on page 172).
5. Click **OK**.

Enabling Syntax Checking

Enable syntax checking to perform the following tests on a script while it is running.

Table 19: While-Running Script Tests

Test	Description
Type Inspection	WebLOAD Recorder checks that each property receives the correct type. For example, `wlLocals.ParseForms = 14` prompts the following log message: `"Wrong type for the property ParseForms. The correct type is Boolean. Legal values are: "Yes"/"No" or "true"/"false".`

Test	Description
Value Inspections	WebLOAD Recorder checks to ensure that each property is assigned a legal value. For example, `wlHttp.Version = "2.1"` prompts the following log message: `"2.1 is an illegal value for the property Version. Legal values are: 1.0, 1.1."`
Scope Inspections	WebLOAD Recorder checks to ensure that each property is assigned a permitted scope. For example, `wlLocals.ConnectionSpeed = 28800` prompts the following log message: `"The property ConnectionSpeed is not valid for the object wlLocals."`
Case Inspections	WebLOAD Recorder objects and properties are case sensitive. When syntax checking is enabled, WebLOAD Recorder checks to ensure that all objects and properties are written correctly. For example, `wlLocals.parse = "No"` prompts the following message: `"The property parse should be written Parse."`

We recommend that syntax checking be run at least once while developing a script.

To enable syntax checking:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.
2. Select the **Diagnostic** tab.
3. Select the **Enable syntax check** checkbox.

Enabling RadView Support Diagnostic

Enabling the RadView support diagnostic option creates large files in the `WebLOAD Recorder\User\Log` directory that may affect Load Generator performance.

To enable RadView Support diagnostic:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.
2. Select the **Diagnostic** tab.
3. Select the **Enable the enhanced RadView support diagnostic** checkbox.

Configuring the Java Options

The Java options enable you to define the Java Virtual Machine to be used by WebLOAD Recorder, for executing Java classes.

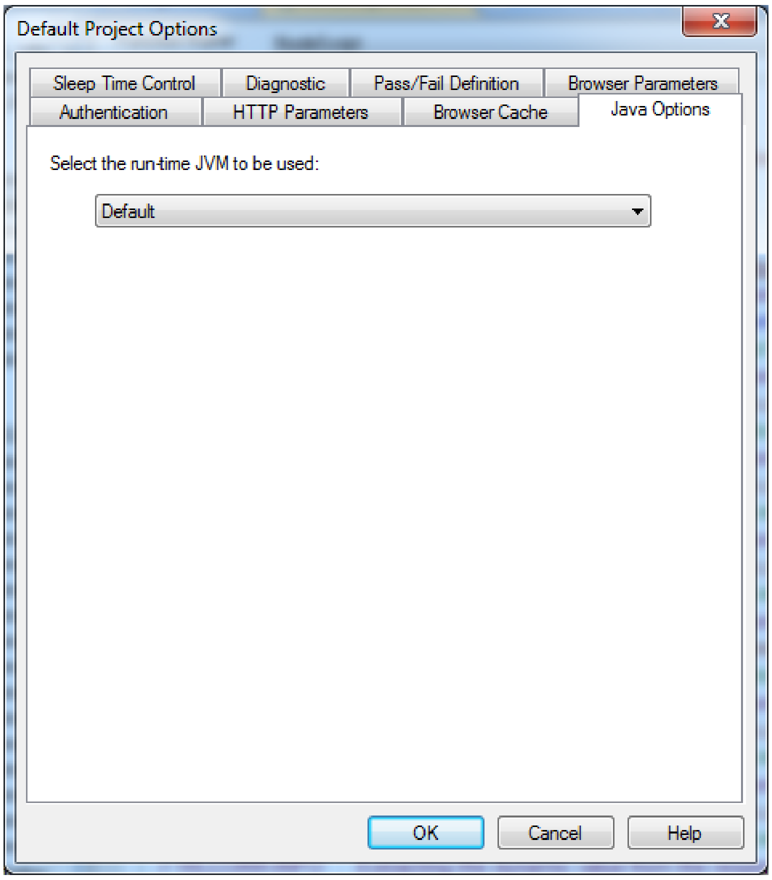
To configure Java Option settings:

1. Click **Default/Current Project Options** in the **Tools** tab of the ribbon.

   The Default/Current Project Options dialog box appears.

2. Select the **Java Options** tab.

   The Java Options tab moves to the front of the dialog box.



Select the run-time JVM to be used:

Default

Figure 113: Java Options Tab

3. In the Select run-time JVM to be used drop down, select one of the available Java Virtual Machines. The default setting is the WebLOAD standard Java Virtual Machine.

The selected value is passed to `wlGlobals.JVMtype` and is the key for `WLJVMs.xml`. This `XML` file (located on every WebLOAD Machine in the `...\extensions\JVMs` directory) contains the following parameters for each JVM:

- Type (the value from the flag)
- Path (should be machine-agnostic)
- Options

When Type is "Default", the RadView default (installed) JVM will be used. The default JVM's path is defined in `webload.ini`, as it depends on the WebLOAD installation path.

4. Click **OK**.

   The Java Options are saved.

Configuring the Recording and Script Generation Options

The Recording and Script Generation Options enable you to define the behavior of the WebLOAD Recorder during the recording and script generation of a Web session.

Opening the Recording and Script Generation Options

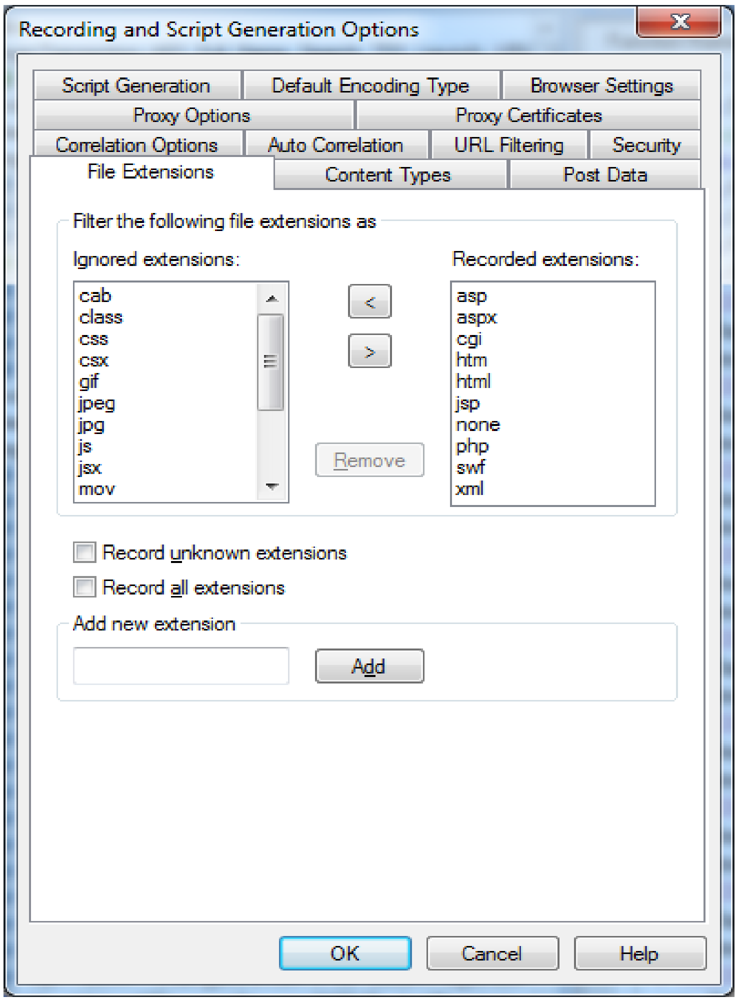
To open the Recording and Script Generation Options dialog box:

- Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

  Or-

  Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

  The Recording and Script Generation Options dialog box opens with the File Extensions tab displayed.



Filter the following file extensions as

Ignored extensions: cab, class, css, csx, gif, jpeg, jpg, js, jsx, mov

Recorded extensions: asp, aspx, cgi, htm, html, jsp, none, php, swf, xml

☐ Record unknown extensions

☐ Record all extensions

Add new extension

Figure 114: Recording and Script Generation Options Dialog Box – File Extensions Tab

The following table describes the tabs in the Recording and Script Generation Options dialog box.

Table 20: Recording and Script Generation Options Dialog Box Tabs

Tab	Description
Proxy Certificates	Configure the Server Side and Client Side certificates.
Script Generation	Define how the WebLOAD Recorder should handle various HTTP elements.
Default Encoding Type	Select the default encoding type.
Browser Settings	Select the default browser. If you selected either Microsoft Internet Explorer or Netscape Navigator, you can also request that the program configure the proxy value automatically (default). If you want to configure the proxy value manually, see *Configuring the Proxy Value for Your Browser* (on page 14).

Tab	Description
File Extensions (default)	Select which file types should be recorded and which ones should not.
Content Types	Select which objects should be recorded and which ones should not.
Post Data	Define how the WebLOAD Recorder should handle Post Data.
Proxy Options	Configure the proxy values for WebLOAD Recorder and the application (if necessary).
Correlation Options	Define correlation and logging options.
Auto Correlation	Define the auto-discovery correlation options.
URL Filtering	Configure which types of URLs the WebLOAD Recorder records.

Specifying the Script Content to be Generated

Use the Script Generation tab in the Recording and Script Generation Options dialog box to specify what the WebLOAD Recorder should record in your script. The Script Generation tab lists all the HTTP objects that can be automatically identified by the WebLOAD Recorder and recorded in the script so you do not have to enter them manually. For example, you can instruct WebLOAD Recorder whether or not to record and display the headers.

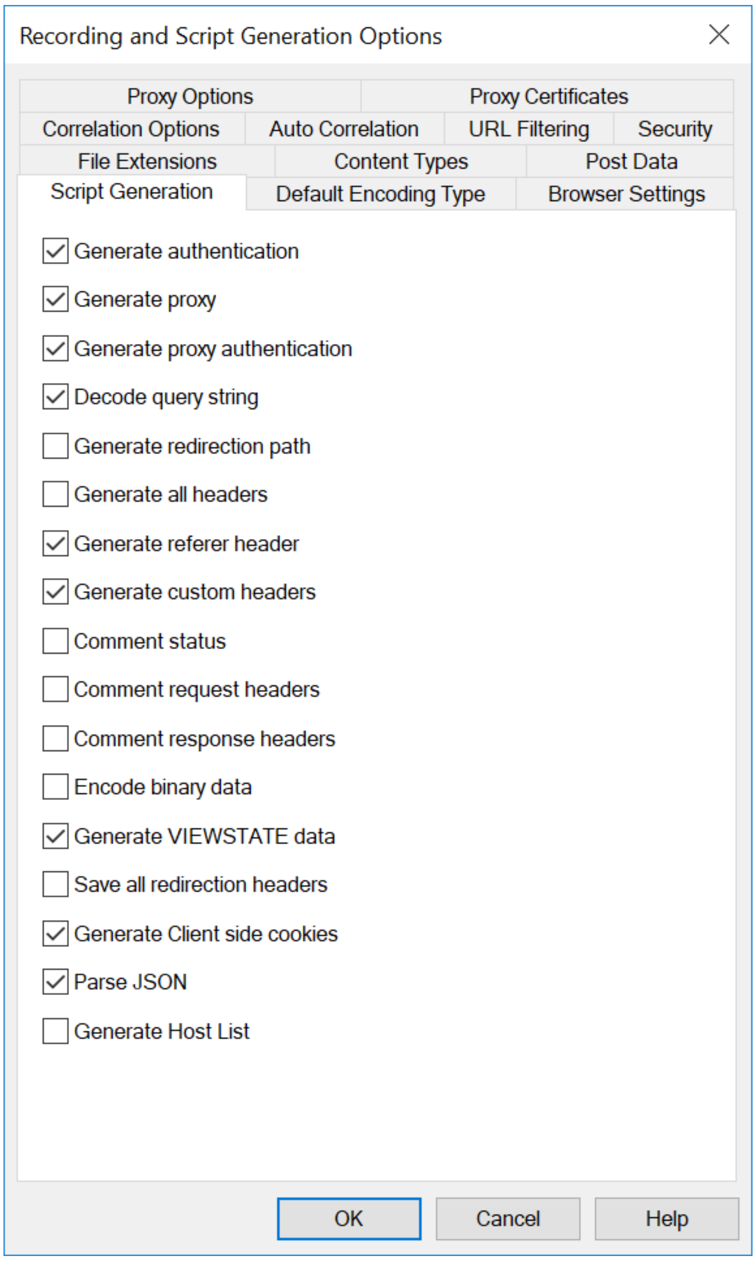
To specify the HTTP objects to be recorded:

1. Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Script Generation** tab.

   The Script Generation tab moves to the front of the dialog box.



- [✓] Generate authentication
- [✓] Generate proxy
- [✓] Generate proxy authentication
- [✓] Decode query string
- [ ] Generate redirection path
- [ ] Generate all headers
- [✓] Generate referer header
- [✓] Generate custom headers
- [ ] Comment status
- [ ] Comment request headers
- [ ] Comment response headers
- [ ] Encode binary data
- [✓] Generate VIEWSTATE data
- [ ] Save all redirection headers
- [✓] Generate Client side cookies
- [✓] Parse JSON
- [ ] Generate Host List

Figure 115: Script Generation Tab

3. Select or clear the options to specify what the WebLOAD Recorder should record in your script.
4. Click **OK**.

The following table describes the options in the Script Generation tab.

Table 21: Script Generation Tab Options

HTTP Object	Description	Example

Generate Authentication	Generates the name and password that appear in the header of a request.
	
	This option is selected by default.
		`wlHttp.UserName="John"`
		
		`wlHttp.Password="Blue"`

Generate proxy	Generates the proxy setting. This option is selected by default.	`wlHttp.Proxy=`*<ProxyName>*`.`*<ProxyPort>*
Generate proxy authentication	Generates the name and password used to identify you to the proxy. This option is selected by default.	`wlHttp.ProxyUserName=`*<UserName>* `wlHttp.ProxyPassword=`*<Password>*

Decode query string	Records the query string that is the part of the URL after the "?" sign and is used for sending parameters for the specific server item which is targeted by this URL.	When this option is not selected, the GET command will displayed as follows:
		
		`wlHttp.Get("http://localhost/netizenbank/myAccountWelcome.asp?netizenSID=31341549426&ssn=1234&password=1231&I1.x=21&I1.y=10")`
		
		That is all the parameters that will appear as part of the URL.
		
		When this option is selected, the URL will be parsed and displayed as follows:
		
		`wlHttp.FormData["netizenSID"] = "31341549618"`
		
		`wlHttp.FormData["ssn"] = "124"`
		
		`wlHttp.FormData["password"] = "3424"`
		
		`wlHttp.FormData["I1.x"] = "29"`
		
		`wlHttp.FormData["I1.y"] = "14"`
		
		`wlHttp.Get("http://localhost/netizenbank/myAccountWelcome.asp")`
		
		That is a block of parameters that will be easier to parameterize.

HTTP Object	Description	Example

Generate redirection path	WebLOAD Recorder records (in the script) only the first GET statement; the rest of the URLs visited when redirected are inserted into the script as comments. The script does not revisit all the URLs during playback.	`wlHttp.Get ("http://www.abcdef.com")`
		`// Redirections:`
		`http://www.ghijkl.com, etc.`

Generate all headers	Generates **all** HTTP headers.
	The headers `If-Modified-Since`, `If-None-Matched`, and `Keep Alive` will be commented out to overcome the situation where recorded links were fetched from the browser's cache during the recording.
	The request header `Accept-Encode: gzip` will also be commented out, to ensure correct behavior.
	When Generate All Headers is selected, Generate Referer Header and Generate Custom Header are automatically checked and disabled so that they cannot be unchecked.
		`wlHttp.Header["user-agent"] ="Mozilla/4.04 [en] (WinNT; I)"`
		`wlHttp.Header["accept-charset"] ="iso-8859-1,*,utf-8"`
		`wlHttp.Header["proxy-connection"] ="Keep-Alive"`
		`wlHttp.Header["accept-language"] ="en"`

HTTP Object	Description	Example
Generate referer header	Generates the referer header only. This header tells the server which URL submitted the request. For example, if you click a link from page xxx, the browser will send that url as the referer. This option is selected by default. This option is automatically selected and cannot be changed when Generate All Headers is selected.	`wlHttp.Header["Referer"] = "http://www.easycar.com/"`
Generate custom headers	Generates any headers that are not explicitly defined in the RFC, such as the SOAP Action header. This option is selected by default.	

Comment status	Writes a comment about the status of your transactions (that is, any GET statement), including information about the contents of the pages.	`wlHttp.Get("http://www.RadView.com/")`
		`//200 OK`

Comment request headers	Writes a comment for each HTTP request.	`// Request Headers:`
		`// user-agent=Mozilla/4.0 (compatible; MSIE 5.01; Windows NT)`
		`// accept-encoding=gzip, deflate`
		`// proxy-connection=Keep-Alive`

Comment response headers	Writes a comment for each reply to HTTP request.	`// Response Headers:`
		`// content-type=text/html`
		`// server=Microsoft-IIS/4.0`
		`// date=Thu, 06 Jan 2000 16:12:44 GMT`
		`// via=1.1 localhost (Jigsaw/1.0a5)// 200 OK`

Encode binary data	Used to specify if the binary data should be encoded. By default this flag is not selected.	If a mobile operator wants to simulate the sending of binary data from the browser (phone) to the server. Part of the binary data is a value (for example, phone number) that needs parameterization. When the EncodeBinaryData flag is selected, the binary form data "x0Ax0BAMIRx00" appears as "%0A%0BAMIR%00" in the script.

HTTP Object	Description	Example
Generate VIEWSTATE data	Enables filtering the VIEWSTATE data while recording. When this is not selected, VIEWSTATE data will be commented out in the script.	
Save all redirection headers	Records the headers for all URL redirections.	
Generate Client side cookies	When unchecked, the web page sets cookies from the JavaScript and you must implement the cookies manually in the script. If selected, the cookies from the headers are compared to cookies that the server sends. If there is a difference, the correct SetCookie command is added to the script. This is performed during recording. The cookie value is obtained from the recorded traffic. WebLOAD automatically inserts a comment before the `SetCookie` command in the script to let the user know that the cookie was added automatically.	

Parse JSON	Used to specify how to present JSON values in the script appearing in the JavaScript View pane – whether as a long string, or parsed.	An example of a long script display:
		`wlHttp.FormData["data"]="{\"id\":\"001\", \"type\":\"donut\", \"name\":\"Cake\" }"`
		
		An example of parsed JSON display:
		`var json = {`
		`    "id": "0001",`
		`    "type": "donut",`
		`    "name": "Cake"`
		`}`

HTTP Object	Description	Example

Generate Host List	Enables running the same script on different hosts, with minimum editing, by replacing host names with variables.
	
	If selected, each host name in the recording is replaced by a variable.
	
	To specify the actual value of each host variable, define it in the *InitAgenda()* function
		Instead of the script including, for example:
		`wlHttp.Get("https://www.amazon.com/")`
		It will include the following:
		`wlHttp.Get("https://" + domain_www_amazon_com + "/")`
		You can then, using the same script, run one test in which you edit the `InitAgenda()` function so that:
		`domain_www_amazon_com="www.amazon.com"`
		and another test in which:
		`domain_www_amazon_com="www.qalab.com"`

Setting the WebLOAD Recorder to Record Post Data Types

Use the Post Data tab in the Recording and Script Generation Options dialog box to instruct the WebLOAD Recorder how to treat different data types when it is recording. Data can be written in the script as part of the command, as a data block, or in a data file. A data block is stored within the script itself, and is useful when you prefer to see the data directly. A data file stores the data in a local text file, and is useful when you are working with large amounts of data which would be too cumbersome to store within the script code itself, or binary data. When working with data files, only the name of the text file is stored in the script itself. Data can also be recorded as FormData, in which the data is formatted in a tidy name-value format and is url-encoded when sent to the server.

While recording a script, WebLOAD automatically identifies if there are no name-value pairs, checks if there is a valid content type (for example, `text/plain`), and records it accordingly (for example, as Data).

Note: The content type application/x-www-form-urlencoded (with or without a charset), should always be recorded as FORMDATA, unless you explicitly specify to record it as DATA or DATA FILE.

For complete details on Post Data recording, see the *WebLOAD Scripting Guide*.

To set the WebLOAD Recorder to record data types:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

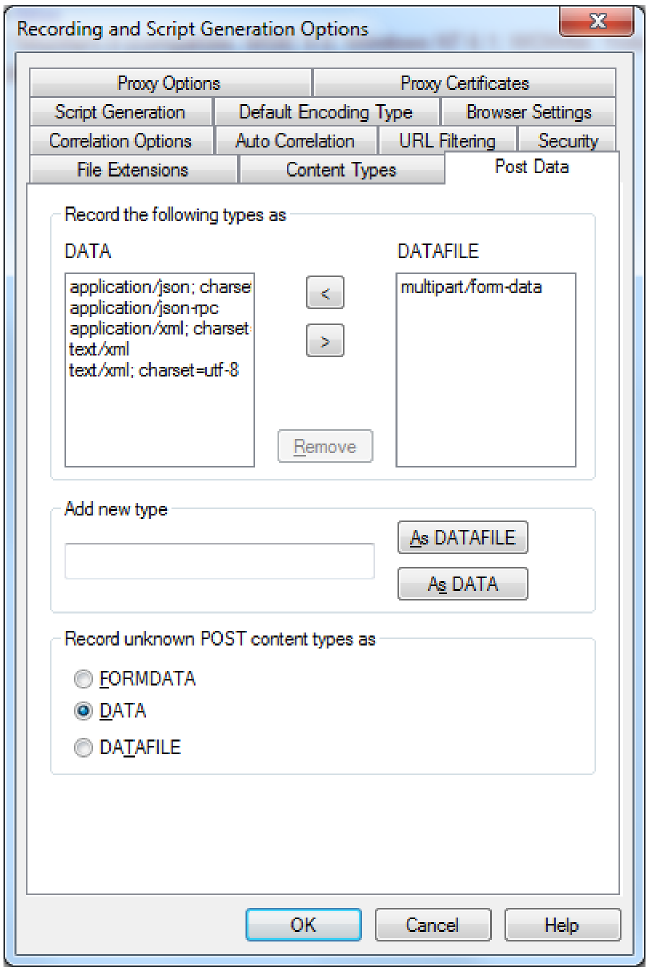
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Post Data** tab.

   The Post Data tab moves to the front of the dialog box.



Record the following types as

DATA
application/json; charse
application/json-rpc
application/xml; charset
text/xml
text/xml; charset=utf-8

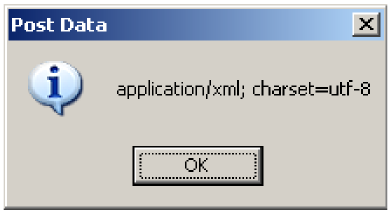
DATAFILE
multipart/form-data

Add new type

Record unknown POST content types as
FORMDATA [✓ DATA] DATAFILE

Figure 116: Post Data Tab

3. Fill in the fields, as described in Table 22, below.
4. Optionally, you can double-click an item from the DATA block or DATAFILE block lists to display the item's full text.



application/xml; charset=utf-8

Figure 117: Post Data Full Text Message Box

5. Click **OK** to return to the Post Data tab.
6. Click **OK** to save the record options settings.

The following table defines all the fields and options in the Post Data tab.

Table 22: Post Data Tab Fields and Options

Field	Description
DATA block	Lists the types of data that the WebLOAD Recorder records as DATA blocks in the script's JavaScript. A DATA block is recorded without HTTP encoding and is not structured. During playback, the WebLOAD Recorder makes this data into form data and sends it without any further modification. A DATA block is for posting data that is not meant to be HTTP encoded, for example Web service calls.
DATAFILE block	Lists the types of data that the WebLOAD Recorder records as DATAFILE blocks (files with a name and a path). A DATAFILE block can store text and binary data. During playback, the WebLOAD Recorder copies and then sends this file with multipart form data, using a MIME protocol.
Remove	Click this button to delete a selected DATA block or DATAFILE block from both lists.
Add new type	Type the name of a new type you want to be added to either of the lists.
As DATAFILE	Adds the new type you entered in the Add new type field to the DATAFILE block list.
As DATA	Adds the new type you entered in the Add new type field to the DATA block list.
Record Unknown Post Types as	Select to instruct the WebLOAD Recorder to record any data type not defined on this tab as: • FORMDATA • DATA • DATAFILE

By default, the following content types are recorded as DATA blocks:

- `application/json; charset=utf-8`
- `application/json-rpc`
- `application/xml; charset=utf-8`
- `text/xml`
- `text/xml; charset=utf-8`

Note: The recorder searches for exact content types from this list. Therefore, `text/xml` and `text/xml; charset=utf-8` are different content types even though the former is a subset of the latter.

WebLOAD Recorder deals specially with the following content types:

- multipart/form-data – This content type is used for uploading files. The actual content type sent by the client is `multipart/form-data;boundary=long-string`. The recorder searches for the `multipart/form-data` content type and then records the request as a From Data, although it appears in the DATAFILE block list.
- multipart/text – This content type is similar to `multipart/form-data`, except that the `multipart/text` content type is not used for uploading files. The `multipart/text` content type is therefore handled as a DATA block, but since it contains a variable in the name (the value of `boundary` in `multipart/text;boundary=` …), the treatment of this content type is hard-coded. For example, any content type that starts with `multipart/text` is recorded as a DATA block content type.
- soap messages – This content type is always recorded as DATA blocks.

Configuring the Default Encoding Type

Use the Default Encoding Type tab in the Recording and Script Generation Options dialog box to set up the default encoding type.

To configure the default encoding type:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

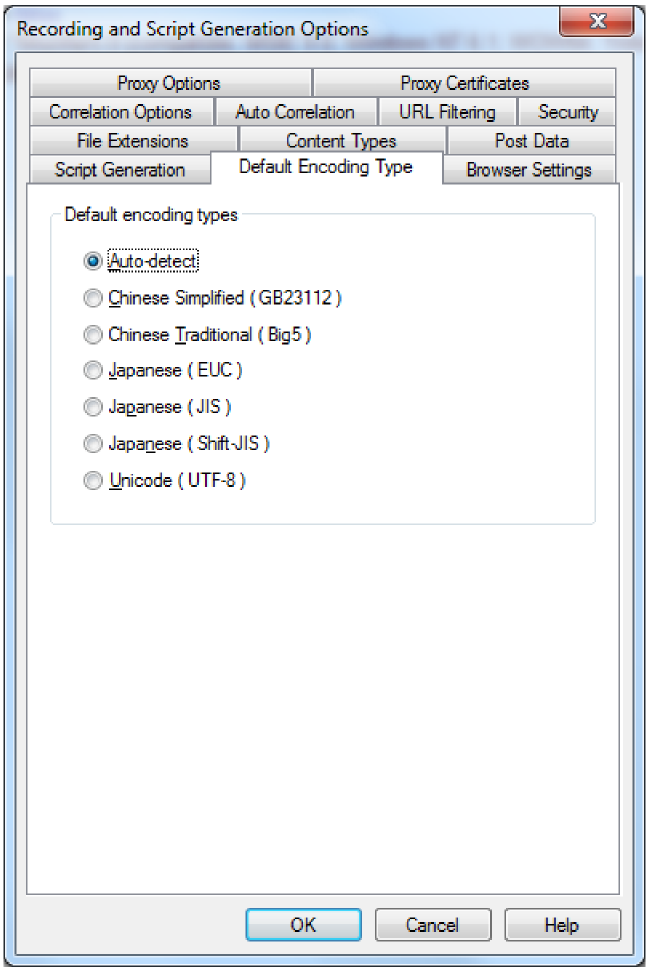
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Default Encoding Type** tab.

   The Default Encoding Type tab moves to the front of the dialog box.



Default encoding types

- [✓ Auto-detect]
- Chinese Simplified (GB23112)
- Chinese Traditional (Big5)
- Japanese (EUC)
- Japanese (JIS)
- Japanese (Shift-JIS)
- Unicode (UTF-8)

Figure 118: Default Encoding Type Tab

3. Select an option as the default encoding type.
4. Click **OK**.

Configuring the Default Browser

Use the Browser Settings tab in the Recording and Script Generation Options dialog box to set up the default browser.

To configure the default browser:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

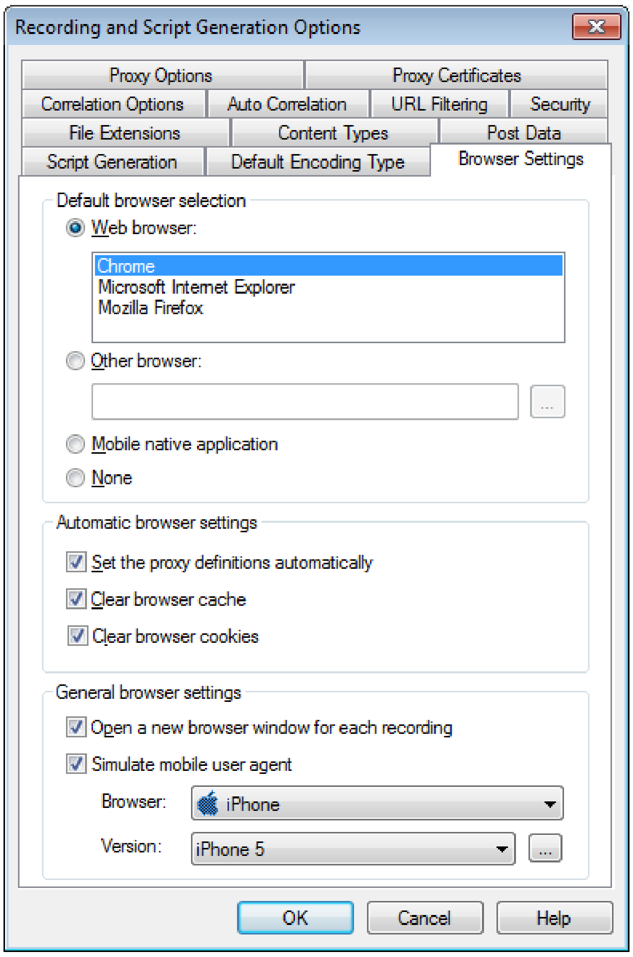
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Browser Settings** tab.

The Browser Settings tab moves to the front of the dialog box.


Recording and Script Generation Options

Tabs: Proxy Options, Proxy Certificates, Correlation Options, Auto Correlation, URL Filtering, Security, File Extensions, Content Types, Post Data, Script Generation, Default Encoding Type, Browser Settings

Default browser selection
- (●) Web browser: Chrome (selected); Microsoft Internet Explorer; Mozilla Firefox
- ( ) Other browser:
- ( ) Mobile native application
- ( ) None

Automatic browser settings
- [✓] Set the proxy definitions automatically
- [✓] Clear browser cache
- [✓] Clear browser cookies

General browser settings
- [✓] Open a new browser window for each recording
- [✓] Simulate mobile user agent
- Browser: iPhone
- Version: iPhone 5

OK | Cancel | Help

Figure 119: Browser Settings Tab

3. Fill in the fields, as described in Table 23.
4. Click **OK**.

   A message appears stating that in order for WebLOAD Recorder to change your proxy definition automatically, you must close all instances of the browser before recording.

   After you close all instances of the browser, the WebLOAD Recorder screen appears.

The following table defines all the fields and options in the Browser Settings tab.

Table 23: Browser Settings Tab Fields and Options

Field	Description
Default browser selection	
Web browser	Select this option to define Google Chrome, Mozilla Firefox or Microsoft Internet Explorer as your default browser. If you selected Mozilla Firefox as your browser, and Mozilla Firefox was installed on the machine *after* WebLOAD Recorder was installed, a message appears recommending that you install the Firefox extension responsible for setting the proxy definitions automatically. If you accept, the extension is installed. If you do not accept, the **Set the proxy definitions automatically** checkbox is automatically cleared, and you should configure the proxy value manually (see *Configuring the Proxy Value for Your Browser* on page 14). The next time you check the **Set the proxy definitions automatically** checkbox, WebLOAD Recorder will show the installation message again.
Other browser	Select this option and browse to define a browser other than Google Chrome, Mozilla Firefox or Microsoft Internet Explorer as your default browser.
Mobile native application	Select this option to define a mobile native application as your default browser. This option is intended for recording from a mobile device. To do so, you must setup the device and the system as described in Recording Mobile Applications (on page 375).
None	Select this option to define that there is no default browser.
Automatic browser settings	
Set the proxy definitions automatically	If you selected either Mozilla Firefox or Microsoft Internet Explorer, you can also set WebLOAD Recorder to configure their proxy settings automatically (default). If you want to configure the proxy value manually, see *Configuring the Proxy Value for Your Browser* (on page 14).
Clear browser cache	Select this option to clear the browser cache before recording. This option is selected, by default.
Clear browser cookies	Select this option to clear the browser's cookie history before recording. This option is selected, by default.
General browser settings	
Open a new browser window for each recording	Select this option to open a new browser window each time you start recording. The first time you start recording, a message is displayed with information about this option. You can disable this message by checking the **Don't show this message again** checkbox.

Field	Description
Simulate mobile user agent	Select this option to simulate a mobile web application. If you select this option, define the specific user agent (browser type and browser version) you wish to simulate. You can click the Change button ... to edit the user agent definition. See *Editing Browser Version Definitions* (on page 164).

Configuring the Correlation Options

Use the Correlation Options tab in the Recording and Script Generation Options dialog box to set up the correlation options.

To configure the correlation options:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

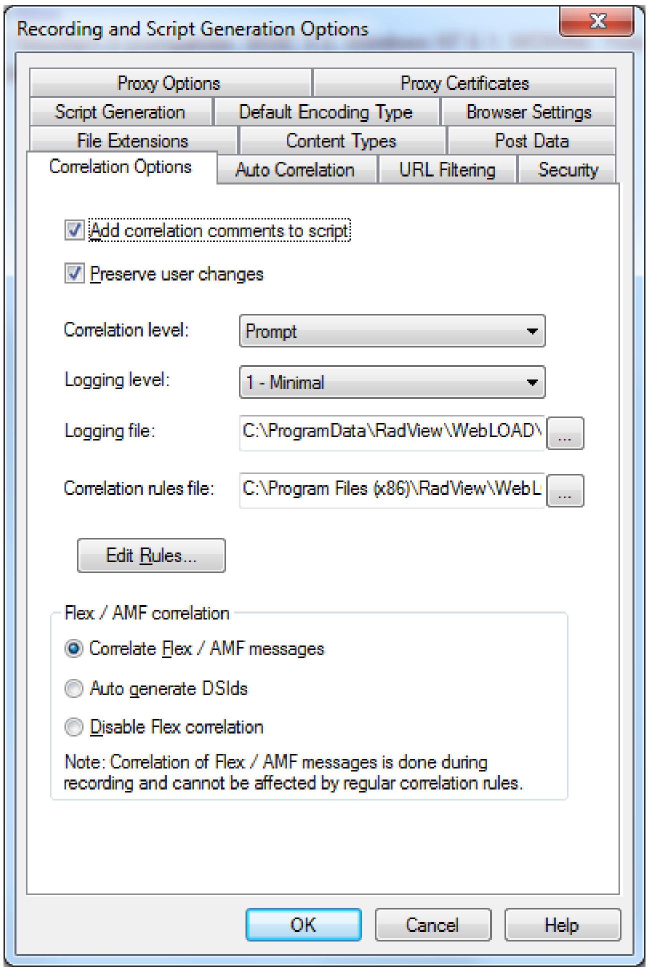
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Correlation Options** tab.

   The Correlation Options tab moves to the front of the dialog box.


Recording and Script Generation Options

[✓] Add correlation comments to script

[✓] Preserve user changes

Correlation level: Prompt

Logging level: 1 - Minimal

Logging file: C:\ProgramData\RadView\WebLOAD\

Correlation rules file: C:\Program Files (x86)\RadView\WebL

Flex / AMF correlation

● Correlate Flex / AMF messages

○ Auto generate DSIds

○ Disable Flex correlation

Note: Correlation of Flex / AMF messages is done during recording and cannot be affected by regular correlation rules.

Figure 120: Correlation Options Tab

3. Fill in the fields, as described in Table 24.
4. Click **OK**.

Table 24: Correlation Options Tab Fields and Options

Field	Description

Add correlation comments to script	Select this option to instruct WebLOAD to add comments to your script in the places where correlation was performed and create a log of all the changes that were made to your script's JavaScript.
	
	When selected, the following comment is added to your script before a command that extracts the dynamic value from a response or that uses a parameter instead of a dynamic value in a request:
	
	`//WLCORR – Extracting the dynamic value from the response according to Correlation Rule <ID>s`
	
	-Or-
	
	`//WLCORR – Using the Correlation Parameter instead of the dynamic value according to Correlation Rule <ID>s`
	
	where `<ID>` is the correlation rule ID.

Field	Description
Preserve user changes	Specify whether to preserve or discard user changes before running correlation. • When this option is unselected, all manual (user) changes to the script are discarded before running correlation. This is equivalent to performing Script Regeneration prior to running correlation. • When this option is selected (default), user changes are preserved when correlation is run. If the changes introduced by correlation conflict with the changes made by the user, the user is requested to resolve the conflict, as described in *Resolving Conflicts between Manual Changes and Correlation Changes* on page 96.
Correlation level	Specify the correlation level to determine the type of correlation to run automatically after recording the script: Possible values: • Do not run – Do not run correlation after recording the script. • Use existing rules – Perform manual correlation after recording the script, using the existing rules. • Discover rules – Perform automatic correlation after recording the script, to discover and suggest new rules. • Prompt – After recording the script, a dialog box is displayed enabling you to select the type of correlation you wish to perform (Do not run, Use existing rules, or Discover rules).
Logging level	Specify the correlation logging level to determine the amount and content of the comments that the correlation engine adds to your script's JavaScript. Possible values: • 0 – None. No log messages are added to the JavaScript. • 1 – Minimal. Fatal, Error, and Warning messages are added to the JavaScript. Fatal messages indicate that an unrecoverable error occurred, Error messages indicate that a recoverable error occurred, and Warning messages indicate that there is a possible error. • 2 – Medium. In addition to the messages added with the minimal logging level, Info messages are added to the JavaScript. Info messages provide important information, such as, when a rule finds a value or when a correlation hint is found. • 3 – Full. In addition to the messages added with the medium logging level, Debug messages are added to the JavaScript. Debugging messages provide detailed information about the script.
Logging file	Specify the location of the correlation log file. The default file is `correlation.log` and the default location is: `C:\Program Files\Radview\WebLOAD\Log`

Field	Description
Correlation rules file	Specify the location of the correlation rules XML file. The default file is `correlationRules.xml` and the default location is: `C:\Program Files\Radview\WebLOAD\Extensions\Correlation`
Edit Rules	Open the Correlation Rules Editor. For more information about the Correlation Rules Editor, see *Configuring the Correlation Rules* (on page 98).
Flex / AMF correlation	
Correlate Flex / AMF messages	Create automatic correlation rules to correlate RPC Flex messages and Messaging Flex messages. Selecting this option enables correlation of the DSId value. This correlation is part of the AMF script generation, which means the correlation is performed during recording.
Auto generate DSIds	If selected, a new DSId is generated during session initialization. The generated DSId is used for all AMF requests until the next session initialization. If not selected, the DSId is retrieved from the session initialization request (nil request) and is used for all AMF requests until the next session initialization.
Disable Flex correlation	Disable Flex correlation.

Configuring the Auto-Correlation Options

Use the Auto Correlation Options tab in the Recording and Script Generation Options dialog box to set up the Auto Discovery correlation options.

To configure the Auto Discovery correlation options:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

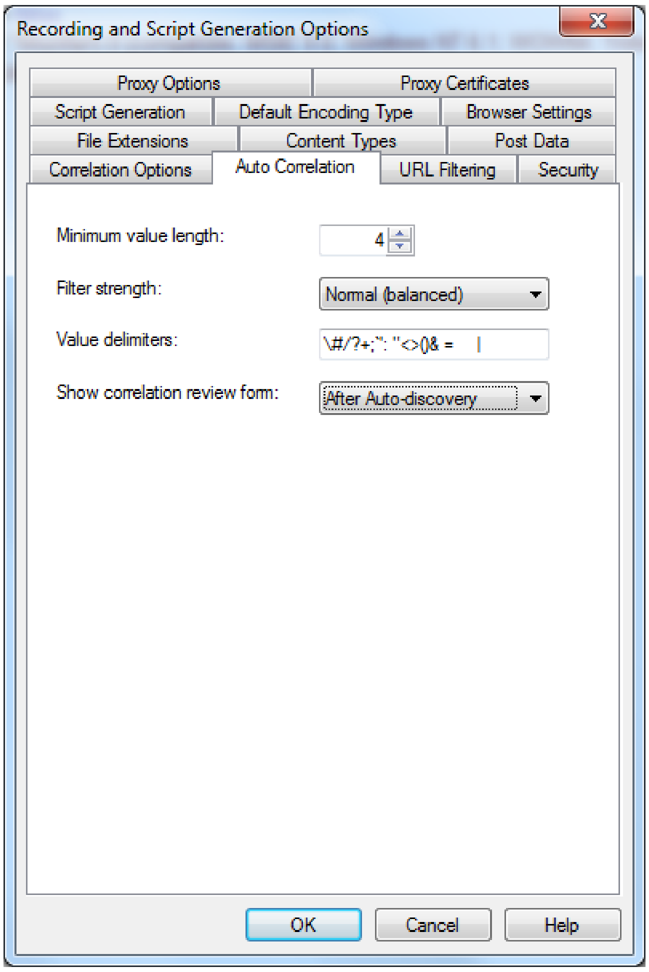
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Auto Correlation Options** tab.

   The Auto Correlation Options tab moves to the front of the dialog box.



Minimum value length: 4

Filter strength: Normal (balanced)

Value delimiters: `\#/?+;": "<>()& =    |`

Show correlation review form: After Auto-discovery

Figure 121: Auto Correlation Options Tab

3. Fill in the fields, as described in Table 25.
4. Click **OK**.

Table 25: Auto Correlation Options Tab Fields and Options

Field	Description
Minimum value length	Specify the minimum length of the value to be considered for correlation. Shorter values, even if matched by a rule, are ignored.

Field	Description
Filter strength	Specify the rules to display in the Correlation Review Form, according to the rule's score. Each rule is given a score during auto-discover correlation according to an algorithm that calculates the chances of the rule being used. Specify the filter strength as follows: • Strict (few records) – Display only the rules that are very likely to be used in the script. This leads to faster script execution, but also has a high risk of missing a necessary rule. • Normal (balanced) – Displays rules that are likely to be used in the script. This leads to average script execution, includes most (if not all) of the necessary rules and displays some rules that are not used. • Weak (many records) – Display rules that have a chance of being used in the script. This leads to slower script execution and displays many rules that are not used.
Value delimiters	Specify the characters to be considered delimiters when searching for a dynamic value during correlation. The correlation engine searches for the dynamic value in the script, where the value is surrounded by a specific delimiter. For example, in: `SessionID=1234&Day` '&' is a delimiter, which defines '`1234`' and 'Day' as two separate strings.
Show correlation review form	Specify when to show the Correlation Review form after performing correlation. Possible values: • Never. • Always. • After Auto-discovery.

Configuring the URL Filtering Options

Use the URL Filtering tab in the Recording and Script Generation Options dialog box to configure which types of URLs the WebLOAD Recorder records.

To configure the URL filtering options:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

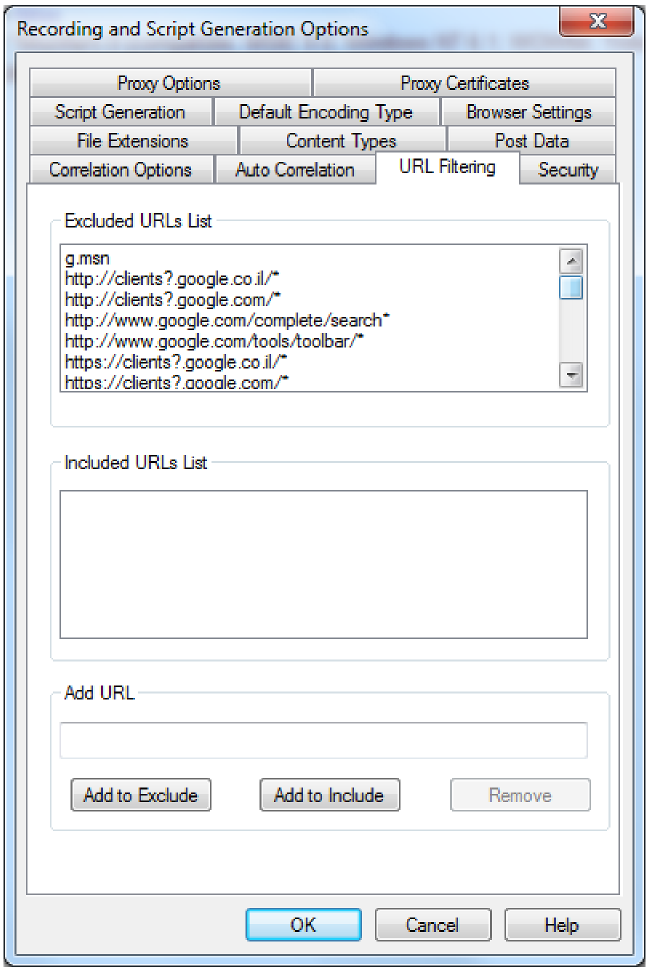
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **URL Filtering** tab.

   The URL Filtering tab moves to the front of the dialog box.



g.msn
http://clients?.google.co.il/*
http://clients?.google.com/*
http://www.google.com/complete/search*
http://www.google.com/tools/toolbar/*
https://clients?.google.co.il/*
https://clients?.google.com/*

Figure 122: URL Filtering Tab

3. Fill in the fields, as described in Table 26.
4. Click **OK**.

The following table describes the fields in the URL Filtering tab.

Table 26: URL Filtering Tab Fields and Options

Field	Description
Excluded URLs List	Lists the URLs that WebLOAD Recorder does not record. WebLOAD Recorder ignores all actions involving any URL in this list when it is encountered during a Web session.

Field	Description
Included URLs List	Lists the URLs that WebLOAD Recorder records. WebLOAD Recorder records all actions involving any URL in this list when it is encountered during a Web session.
Edit URLs List	Type a URL in this field to add the URL to either the Included URLs List or Excluded URLs List.
Add to Exclude	Click to add the URL in the Edit URLs List field to the Excluded URLs List.
Add to Include	Click to add the URL in the Edit URLs List field to the Included URLs List.
Remove	Click to delete a selected URL from either the Included URLs List or the Excluded URLs List.

Configuring the File Extensions

Use the File Extension tab in the Recording and Script Generation Options dialog box to configure which types of files the WebLOAD Recorder records.

Both the File Extensions and the Content Types tabs (see *Configuring the Content Types to Record* on page 198), enable you to specify the types of data that are accepted and recorded by WebLOAD Recorder, or not accepted and ignored. On the File Extensions tab, you specify which objects should be recorded or ignored, according to their file extension, such as "`.gif`", "`.wav`", or "`.txt`".

In a case where the file extension and content types contradict each other, precedence is given to the record filter as opposed to the ignore filter. For example, if the File Extensions and Content Types tabs are configured with the following settings:

- Filter the following file extensions as – Recorded Extensions: `gif`
- Filter the following content types as – Ignored Types: `image/gif`

A resource with the `gif` file extension that contains `image/gif` content is recorded in WebLOAD Recorder even though the `image/gif` content type is set to be ignored.

To configure the file extensions:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

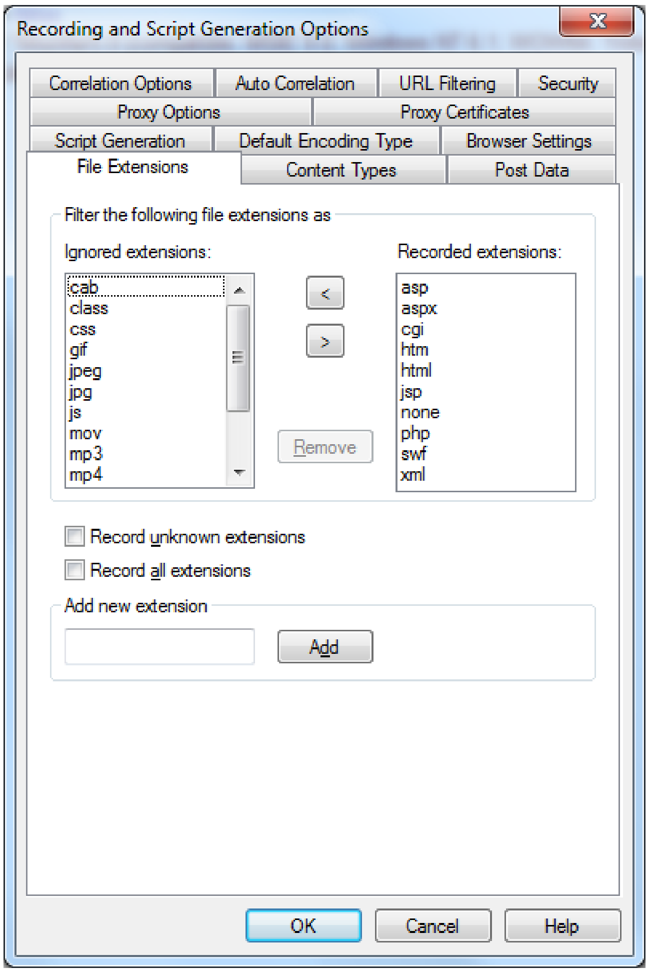
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **File Extensions** tab.

The File Extensions tab moves to the front of the dialog box.



Filter the following file extensions as

Ignored extensions: cab, class, css, gif, jpeg, jpg, js, mov, mp3, mp4

Recorded extensions: asp, aspx, cgi, htm, html, jsp, none, php, swf, xml

☐ Record unknown extensions

☐ Record all extensions

Add new extension

Figure 123: File Extensions Tab

3. Fill in the fields, as described Table 27.
4. Click **OK**.

The following table describes the fields in the File Extensions tab.

Table 27: File Extensions Tab Fields

Field	Description
Ignored extensions	Lists the file extensions that WebLOAD Recorder does not record. WebLOAD Recorder ignores all actions involving any file extension in this list when it is encountered during a Web session.
Recorded extensions	Lists the file extensions that WebLOAD Recorder records. WebLOAD Recorder records all actions involving any file extension in this list when it is encountered during a Web session.
Remove	Click this button to delete a selected file extension from both lists.

Field	Description
Record unknown extensions	Select this option to record all actions involving any unknown file extensions encountered during a Web session. File extensions not defined and listed in the Ignored Extensions window are treated as if they were included in the Recorded Extensions window.
Record all extensions	Select this option to disregard the settings in the Ignored / Recorded Extensions lists. WebLOAD Recorder then records all actions involving all file extensions encountered during a Web session, including unknown file extensions.
Add new extension	Type a new file extension.
Add	Click this button to add the new file extension to the Ignored Extensions list.

Configuring the Content Types to Record

Use the Content Types tab in the Recording and Script Generation Options dialog box to set up which types of Web content the WebLOAD Recorder records.

Both the Content Types and the File Extensions tabs (see *Configuring the File Extensions* on page 196), enable you to specify the types of data that are accepted and recorded by WebLOAD Recorder, or not accepted and ignored. On the Content Types tab you define which objects should be recorded by type, such as "`image/gif`", "`image/jpeg`", or "`text/html`".

In a case where the content types and file extension contradict each other, precedence is given to the record filter as opposed to the ignore filter. For example, if the Content Types and File Extensions tabs are configured with the following settings:

- Filter the following content types as – Recorded Types: `image/gif`
- Filter the following file extensions as – Ignored Extensions: `gif`

A resource with the `gif` file extension that contains `image/gif` content is recorded in WebLOAD Recorder even though the `gif` file extension is set to be ignored.

To configure the content types to record:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

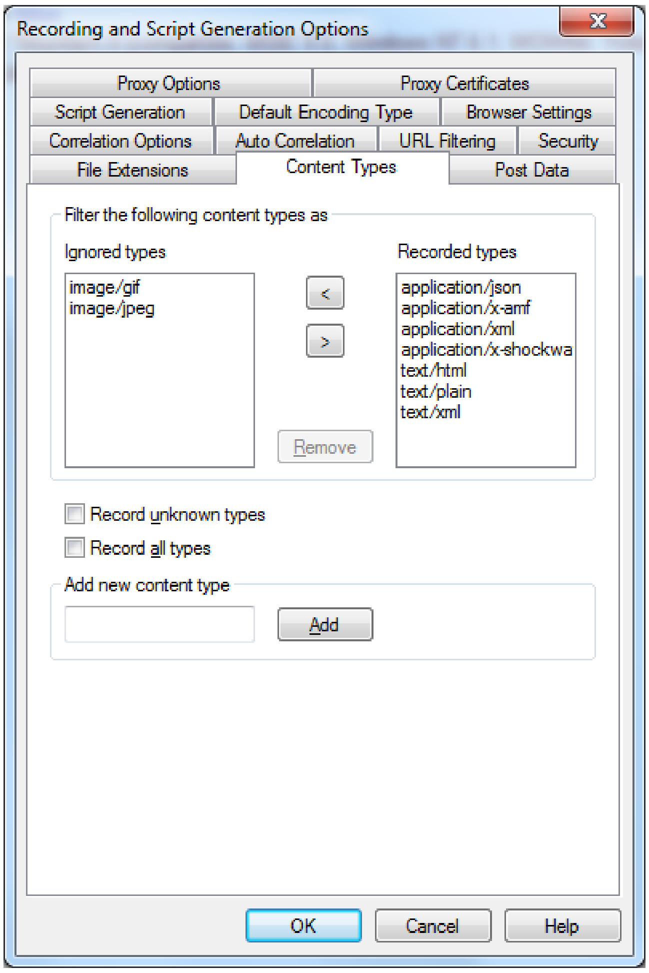
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Content Types** tab.

The Content Types tab moves to the front of the dialog box.



Recording and Script Generation Options

Tabs: Proxy Options, Proxy Certificates, Script Generation, Default Encoding Type, Browser Settings, Correlation Options, Auto Correlation, URL Filtering, Security, File Extensions, Content Types, Post Data

Filter the following content types as

Ignored types: image/gif, image/jpeg

Recorded types: application/json, application/x-amf, application/xml, application/x-shockwa, text/html, text/plain, text/xml

Record unknown types: unchecked

Record all types: unchecked

Add new content type

Figure 124: Content Types Tab

3. Fill in the fields, as described Table 28.
4. Click **OK**.

The following table describes the fields in the Content Types tab.

Table 28: Content Types Tab Fields

Field	Description
Ignored types	Lists the content types that WebLOAD Recorder does not record. WebLOAD Recorder ignores all actions involving any content type in this list when it is encountered during a Web session.
Recorded types	Lists the content types that WebLOAD Recorder records. WebLOAD Recorder records all actions involving any content type in this list when it is encountered during a Web session.
Remove	Click this button to delete a selected content type from both lists.

Field	Description
Record unknown types	Select this option to record all actions involving any unknown content types encountered during a Web session. Content types not defined and listed in the Ignored Types area are treated as if they were included in the Recorded Types area.
Record all types	Select this option to disregard the settings in the Ignored / Recorded Types lists. WebLOAD Recorder then records all actions involving all content types encountered during a Web session, including unknown content types.
Add new content type	Type a new content type.
Add	Click this button to add the new content type to the Ignored Types list.

Setting the Proxy Options

Use the Proxy Options tab in the Recording and Script Generation Options dialog box to designate the proxy server at your organization as the *application proxy* during recording sessions or to change the proxy port number for WebLOAD Recorder.

When you record scripts with the WebLOAD Recorder, your browser must be configured to use proxy port 9884 (which is the default proxy port). In other words, you must record scripts through proxy port 9884.

WebLOAD Recorder enables you to configure a double proxy configuration, which instructs the recorder to use two application proxies, one for regular HTTP traffic and another for secure (SSL) traffic. To configure the double proxy, see *Configuring a Double Proxy* (on page 203).

To set the proxy options:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

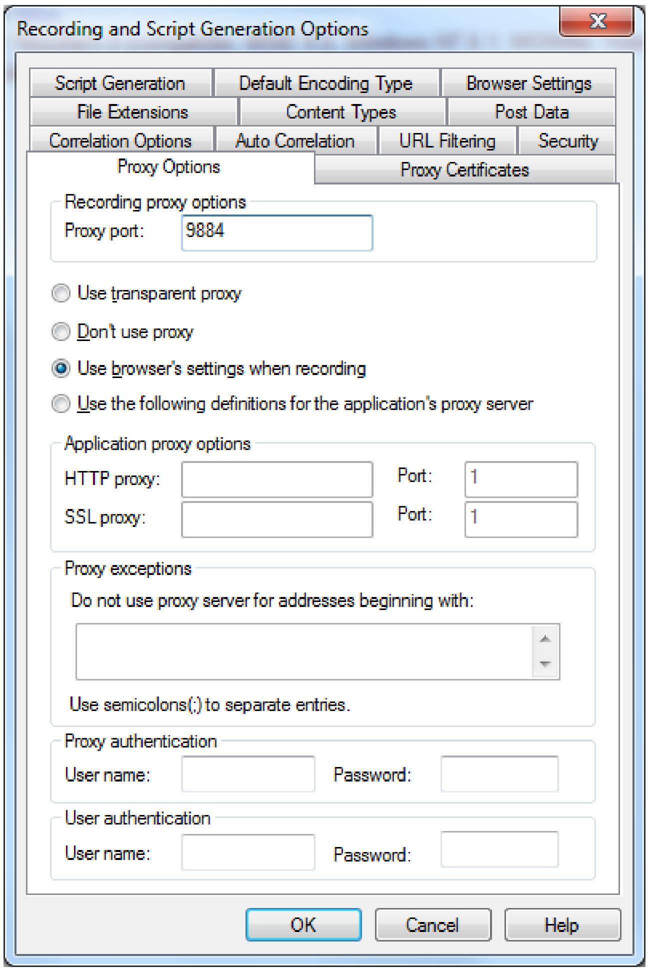
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Proxy Options** tab.

   The Proxy Options tab moves to the front of the dialog box.


Recording and Script Generation Options

Recording proxy options

Proxy port: 9884

( ) Use transparent proxy

( ) Don't use proxy

(●) Use browser's settings when recording

( ) Use the following definitions for the application's proxy server

Application proxy options

HTTP proxy: Port: 1

SSL proxy: Port: 1

Proxy exceptions

Do not use proxy server for addresses beginning with:

Use semicolons(;) to separate entries.

Proxy authentication

User name: Password:

User authentication

User name: Password:

Figure 125: Proxy Options Tab

3. Fill in the fields, as described in Table 29.
4. Click **OK**.

The following table describes the fields and options on the Proxy Options tab.

Table 29: Proxy Options Tab Fields and Options

HTTP Object	**Description**
Recording proxy options	
Proxy port	The port number for the WebLOAD Recorder proxy-recorder. The default value is 9884. When you record scripts, your browser must use the default value.
Use transparent proxy	Select this option to enable WebLOAD Recorder to record from any Web client that does not support proxy configurations. When selected, the **Proxy port** field is disabled. For more information, see *Recording Desktop Web Applications* on page 59.

HTTP Object	Description
Application proxy options	
Use the following definitions for the application's proxy server	Select this option if you use a proxy server to access the Internet. When selected, the **HTTP proxy/Port, SSL proxy/Port** and the **Proxy authentication** area fields are enabled and updated with the current settings from your Internet browser. (This is only relevant for Internet Explorer and Mozilla Firefox. If you are using a different Internet browser, update these fields manually). For additional information on determining if your browser is configured with a proxy, see *Troubleshooting* (on page 62).
HTTP proxy/Port	The address and port number of your organization's proxy, if one exists (for example, to access the Internet beyond a company firewall). Modifying these fields automatically updates your default browser's proxy settings and restores the original settings when the recording process is complete.
SSL proxy/Port	The address and port number of your organization's Secure proxy, if one exists (for example, to access the Internet beyond a company firewall). Use these fields in conjunction with the HTTP Proxy/Port fields to define a double proxy. Modifying these fields automatically updates your default browser's proxy settings and restores the original settings when the recording process is complete.
Use browser's settings when recording	Select this option to enable WebLOAD Recorder to use your default browser's proxy settings when recording a script. When selected, WebLOAD Recorder copies your default browser's proxy settings into the **HTTP Proxy/Port** and **SSL Proxy/Port** fields. (This is only relevant for Internet Explorer and Mozilla Firefox. If you are using a different Internet browser, this is irrelevant).
Proxy authentication	
User name	The user name used for proxy authentication purposes.
Password	The password used for proxy authentication purposes.

HTTP Object	Description
Proxy exceptions	
Do not use proxy server for addresses beginning with	Enter the address of complex addresses you wish to bypass. A proxy bypass entry can begin with a protocol type such as `http://` or `https://`. If a protocol type is used, the exception entry applies only to requests for that protocol. Note that the protocol value is not case sensitive. Multiple entries should be separated by a semicolon (;). Next, enter an Internet address, an IP address, or domain name. If no protocol is specified, any request using the address is bypassed. If a protocol is specified, requests with the address are bypassed only if they are of the indicated protocol type. Both address entries and protocol types are not case sensitive. This field allows a wildcard character (*) to be used in place of zero or more characters.
User Authentication	
User Name	The user name used for user authentication purposes.
Password	The password used for user authentication purposes.

Configuring a Double Proxy

A double proxy configuration is a way to instruct the recorder to use two application proxies: one for non-secure HTTP traffic and one for SSL traffic. When you define only an HTTP proxy as the application proxy in the Proxy Options tab in the Recording and Script Generation Options dialog box, the recorder uses the same definition for both traffic types.

In order to instruct the recorder to use a separate proxy for secured HTTP traffic, define the SSL Proxy and Port values.

WebLOAD Recorder also enables you to set authentication information for accessing the proxy. In the SSL proxy configuration, the User Name and Password values (in the Proxy Authentication frame) are used for both HTTP and SSL proxies. In order to set different authentication information for the SSL proxy, add the following lines to `wlproxyinclude.js` (which can be found in the WebLOAD include directory):

```
ProxyObject.RSecondarySSLProxyUserName = “radview”

ProxyObject.RSecondarySSLProxyPassword=“rad1”
```

Finally, using this configuration will generate JavaScript code to indicate to the playback engine that it needs to use two proxies:

```
wlHttp.UseSameProxyForSSL = false
wlHttp.HttpProxy =

wlHttp.HttpsProxy =
```

The engine will fit the relevant proxy to the request.

Setting the Proxy Certificates

Use the Proxy Certificates tab in the Recording and Script Generation Options dialog box to configure the Server Side and Client Side certificates.

To set the proxy certificates:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

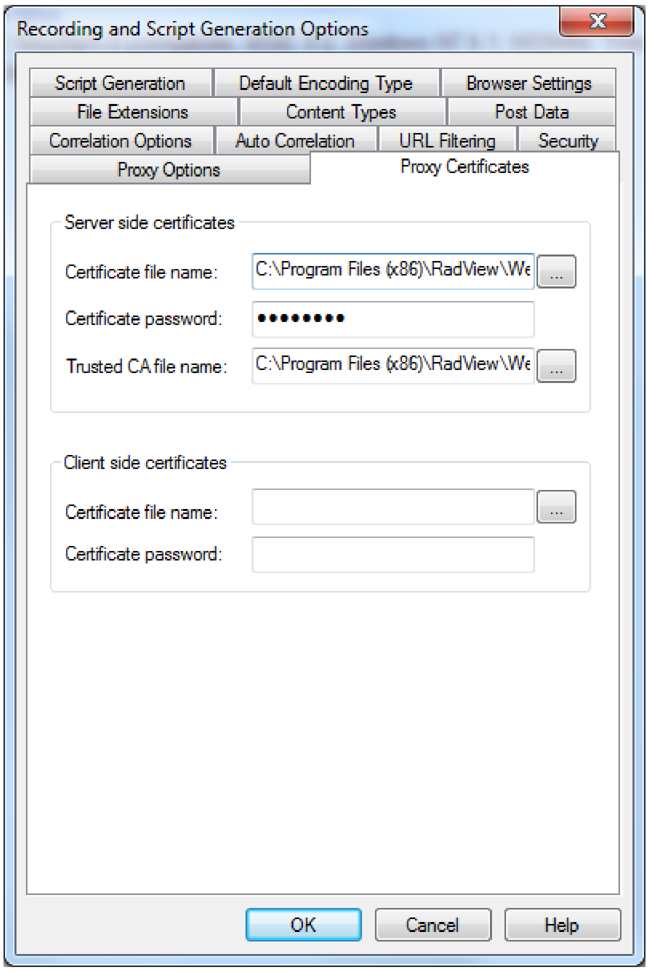
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Proxy Certificates** tab.

   The Proxy Certificates tab moves to the front of the dialog box.


Recording and Script Generation Options

Server side certificates

Certificate file name: C:\Program Files (x86)\RadView\We

Certificate password: ••••••••

Trusted CA file name: C:\Program Files (x86)\RadView\We

Client side certificates

Certificate file name:

Certificate password:

Figure 126: Proxy Certificates Tab

3. Fill in the fields, as described Table 30.
4. Click **OK**.

The following table describes the fields and options on the Proxy Certificates tab.

Table 30: Proxy Certificates Tab Options

Field	**Description**
Server side certificates	
Certificate file name	Browse to the server certificate file that will be used to emulate a server certificate for the user client application. Default: The certificate supplied with the WebLOAD installation.
Certificate password	Type the password for the supplied certificate file. Default: password of the supplied by RadView certificate.

Field	Description
Trusted CA file name	Browse to a Trusted CA file that is a certificate file with the list of trusted certificate authorities. **Note:** We recommend that you use the file supplied with the WebLOAD installation.
Client side certificates	
Certificate file name	Browse to the client certificate file that will be used by the proxy to connect to Internet sites.
Certificate password	Type the password for the supplied certificate file.

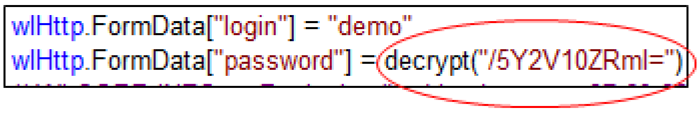
Setting Security Options

Use the Security tab in the Recording and Script Generation Options dialog box to mask passwords in the script.

There are two kinds of passwords you can mask:

- Protocol passwords – WebLOAD protocol password fields. These are the various possible password fields of the wlHttp object. They include the five following fields:
  - ProxyNTPassWord
  - ProxyPassWord
  - HttpsProxyPassWord
  - PassWord
  - NTPassWord
- Form passwords – Password fields in form data. These can vary, depending on the form. You can list the names of the passwords fields whose contents you wish to encrypt.

In the JavaScript code, the encrypted password is replaced with a ‘decrypt’ statement for the encrypted value, as shown in the following example:



```
wlHttp.FormData["login"] = "demo"
wlHttp.FormData["password"] = decrypt("/5Y2V10ZRml=")
```

Note that the purpose of the masking operation is to make sure passwords are not **visible** in plain text.

To set the password encryption options:

1. Click **Recording and Script Generation Options** in the **Tools** tab of the ribbon.

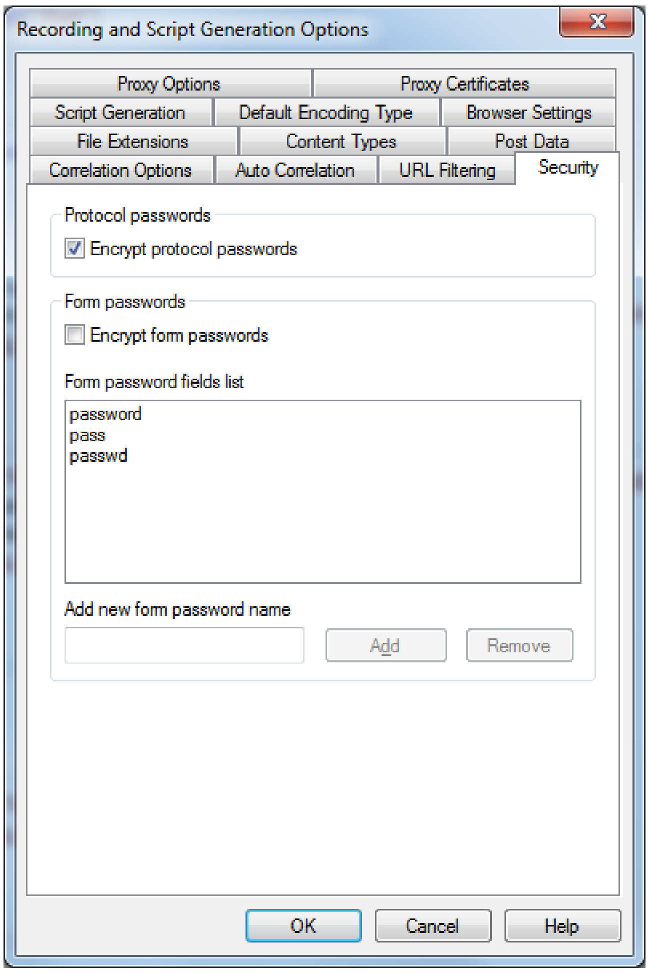
   Or-

   Select **Recording and Script Generation Options** from the WebLOAD Recorder **Home** tab of the ribbon.

   The Recording and Script Generation Options dialog box appears (see Figure 114).

2. Select the **Security** tab.

   The Security tab moves to the front of the dialog box.



Protocol passwords

[✓] Encrypt protocol passwords

Form passwords

[ ] Encrypt form passwords

Form password fields list

password
pass
passwd

Add new form password name

Figure 127: Security Tab

3. Fill in the fields, as described in Table 31.
4. Click **OK**.

The following table describes the fields and options on the Security tab.

Table 31: Security Tab Options

Field	Description
Protocol Passwords	
Encrypt Protocol Passwords	Select this option to instruct WebLOAD to encrypt all WebLOAD protocol passwords.
Form Passwords	
Encrypt Form Passwords	Select this option to instruct WebLOAD to encrypt all form passwords specified in the Form Password Fields List.
Form Password Fields List	Lists the form passwords that WebLOAD will encrypt.
Add new form password name	Type the name of a form password field to add it to the Form Password Fields List.
Add	Click to add a form password field to the Form Password Fields List.
Remove	Click to delete a selected password field from the Form Password Fields List.

Configuring the Settings

WebLOAD Recorder enables you to specify settings for WebLOAD Recorder.

Opening the Settings

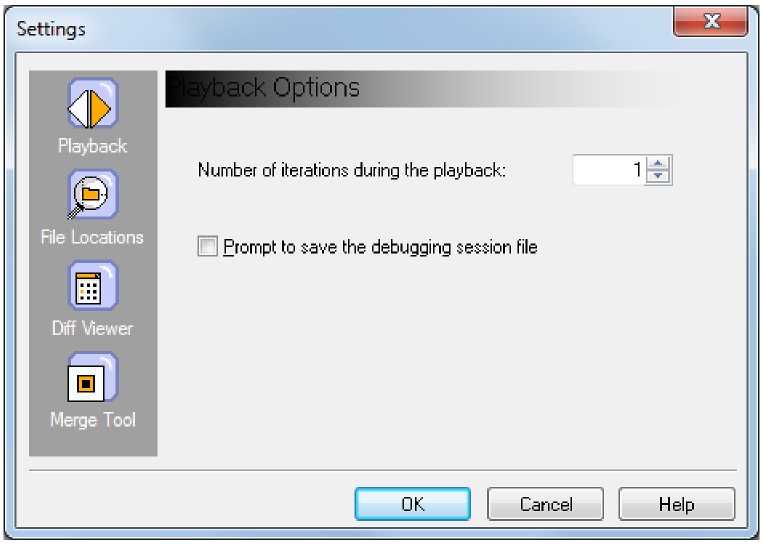
To open the Settings dialog box:

- Click **Settings** in the **Tools** tab of the ribbon.

  Or-

  Select **Settings** from the WebLOAD Recorder **Home** tab of the ribbon.

  The Settings dialog box opens.


Settings

Playback Options

Number of iterations during the playback: 1

☐ Prompt to save the debugging session file

Figure 128: Settings Dialog Box

The following table describes the options in the Settings dialog box.

Table 32: Settings Dialog Box Options

Setting	Description
Playback	Set the number of iterations to run when running scripts with WebLOAD Recorder (Default:1) and determine when to be prompted to save the session file.
File Locations	Define the default file locations during a test session.
Diff Viewer	Define which program is used for comparing recordings to playbacks. The default is WinMerge.
Merge Tool	Define which program is used for resolving code conflicts by editing the JavaScript code. The default is WinMerge.

Setting Playback Options

Set the number of iterations to be run during a test session and whether to prompt to save the session file before returning from debug mode to edit mode.

To set Playback iterations:

1. In the Settings dialog box (Figure 128), click **Playback**.

   The Playback Options screen is displayed (see Figure 128).

2. Specify the number of iterations to run during script playback. The default value is 1.

3. Select **Prompt to save the debugging session file** if you wish to be prompted to save the session file before switching to edit mode. When this is not selected, you are prompted to save the session file only when closing a script or exiting WebLOAD Recorder.
4. Click **OK**.

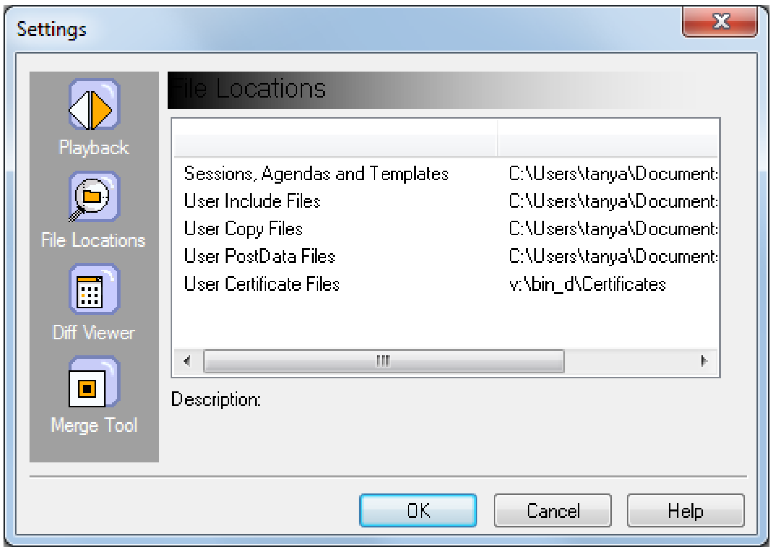
Setting File Locations

Define the default file locations during a test session.

To set the file locations:

1. In the Settings dialog box (Figure 128), click **File Locations**.

   The File Locations screen is displayed.



Sessions, Agendas and Templates	C:\Users\tanya\Document:
User Include Files	C:\Users\tanya\Document:
User Copy Files	C:\Users\tanya\Document:
User PostData Files	C:\Users\tanya\Document:
User Certificate Files	v:\bin_d\Certificates

Figure 129: Settings Dialog Box with File Validation Test

The Description area at the bottom of the dialog provides a short explanation of each file location item.

The following file locations can be defined:

- Sessions, scripts, and Templates: Default storage location for WebLOAD Recorder session, project, and script files.
- User Include Files: Default path for user `Include` files.
- User Copy Files: Default path for user `Copy` files.
- User PostData Files: Default path for user `PostData` files.
- User Certificate Files: Default path for user `Certificate` files.

2. Double-click the file location option that you wish to reset, and select a new file location.
3. Click **OK**.

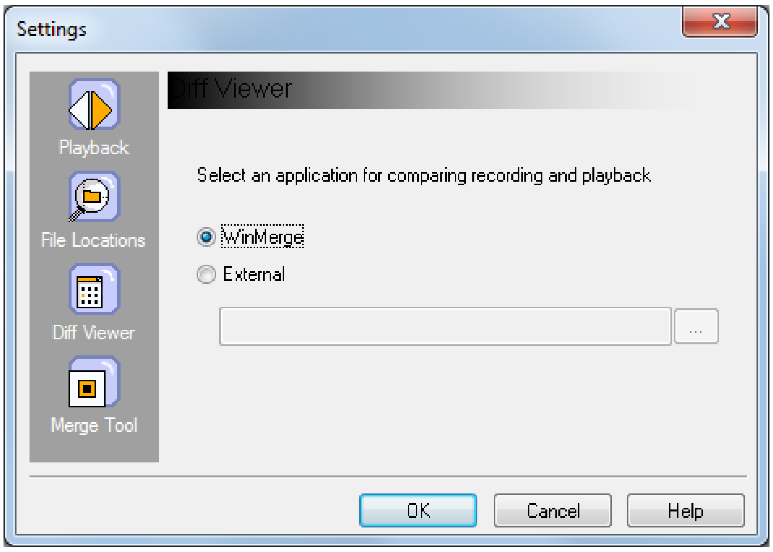
Defining the Difference Viewer Application

Define which application is used for comparing and displaying the differences which may exist between a recording and its playback.

To define the difference viewer application:

1. In the Settings dialog box (Figure 128), click **Diff Viewer**.

   The Diff Viewer screen is displayed.



Select an application for comparing recording and playback

[✓ WinMerge] External

Figure 130: Settings Dialog Box with Diff Viewer Options

   By default, WinMerge is selected.

2. Optionally, select **External** and enter the relevant information into the corresponding field to specify a different application. Enter the following information:
   a. The path to the application's executable file. (Mandatory.)
   b. % rname – Represents the name for the dialog box which displays the recording file. (Optional.)
   c. % pname – Represents the name for the dialog box which displays the playback file. (Optional.)
   d. % record – Represents the path of the recording file. (Optional.)
   e. % playback – Represents the path of the playback file. (Optional.)

Examples:

- ExamDiff Pro:
  C:\Program Files\ExamDiff Pro\ExamDiff.exe % record % playback --left_display_name:% rname --right_display_name:% pname
- KDiff:
  C:\Program Files\KDiff\kdiff3.exe % record % playback --L1 % rname --L2 % pname
- Araxis
  C:\Program Files\Araxis\compare.exe /max /wait /title1:% rname /title2:% pname % record % playback

3. Click **OK**.

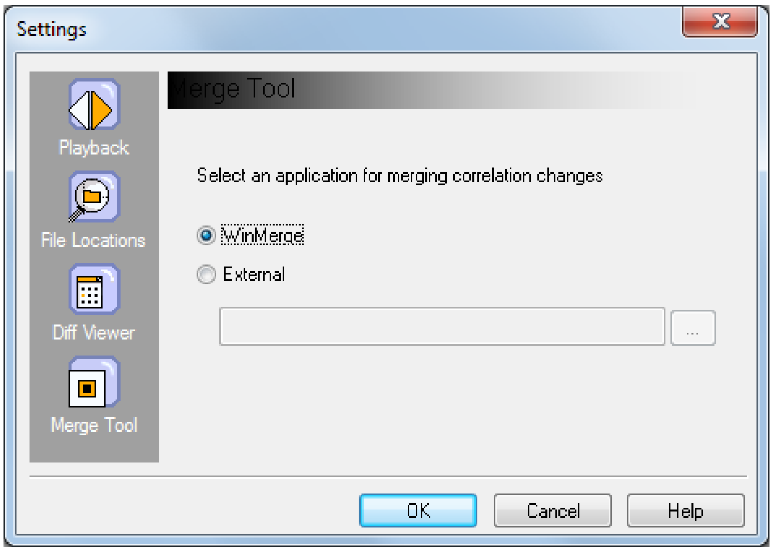
Defining the Merge Tool Application

Define which application is used for resolving conflicts between user changes and correlation changes made to the JavaScript code.

To define the merge tool application:

1. In the Settings dialog box (Figure 128), click **Merge Tool**.

   The Merge Tool screen is displayed.



Merge Tool

Select an application for merging correlation changes

(●) WinMerge
( ) External

Figure 131: Settings Dialog Box with Merge Tool Options

   By default, WinMerge is selected. WinMerge enables 2-way merging.

2. Optionally, select **External** and enter the relevant information into the corresponding field to specify a different application such as Araxis or TortoiseSVN which enable 3-way merging. Enter the following information:

a. %basefile – Represents the path of the base file, before user and correlation changes.

b. %corrfile – Represents the path of the file with the correlation changes.

c. %userfile – Represents the path of the file with the user changes.

d. %outfile – Represents the outcome of the merge file.

Examples:

- Perforce Merge:

  C:\Path-To\P4Merge.exe %basefile %corrfile %userfile %outfile

- KDiff3:

  C:\Path-To\kdiff3.exe %basefile %userfile %corrfile -o %outfile --L1 Base --L2 User --L3 Correlation

- Araxis:

  C:\Path-To\compare.exe /max /wait /3 /title1:Correlation /title2:Base /title3:User %corrfile %basefile %userfile %outfile /a2

- WinMerge (2.8 or later):

  C:\Path-To\WinMerge.exe %outfile

- DiffMerge:

  C:\Path-To\DiffMerge.exe -caption=%mname -result=%outfile –merge -nosplash -t1=%yname -t2=%bname -t3=%tname %userfile %basefile %corrfile

3. Click **OK**.

Customizing the Quick Access Toolbar

You can use the **Customize Quick Access Toolbar** option in the Quick Access toolbar to customize the Quick Access toolbar.

Configuring the Parameterization Manager

The Parameterization Manager enables you to edit a script containing static values and transform it into a script that will run multiple variations of the static values.

When recording a script, WebLOAD captures the data that is being sent, including login details, user selections, and entered text. When running the script under load, simulating many users, it is desirable to use variations in the data, so as to simulate the application more realistically. To do so, you can replace the static values with parameters.

Parameter values can come from a file, or be automatically generated numbers, strings and dates.

The Parameterization Manager enables you to specify how the script should select values from the data file. For example:

- Order considerations – Whether to randomly select values from the data file, or use them in the order they appear.
- Uniqueness considerations – Whether the same value can be used at the same time by different virtual clients.

You can also specify the update policy, which defines when a new value will be read or calculated. For example, whether to update the value on each round, or once at the beginning of the test.

Opening the Parameterization Manager

To open the Parameterization Manager dialog box:

- Click **Parameterization Manager** in the **Home** tab of the ribbon.

  The Parameterization Manager dialog box opens.

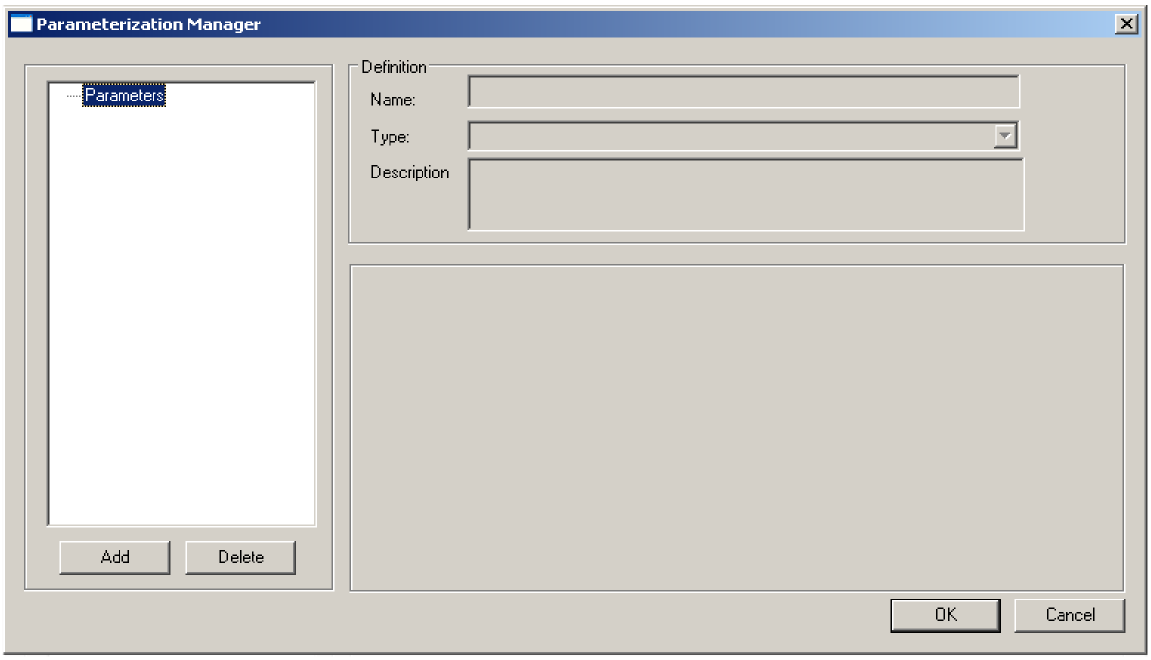


Figure 132: Parameterization Manager Dialog Box

Setting Parameters in the Parameterization Manager

1. In the Parameterization Manager Dialog Box (*Figure 132*), click **Add**.
2. In the **Name** field, enter a name for the parameter.
3. In the **Type** field, select the parameter type:
   - **Date/Time** – Defines a date/time parameter. For more information see *Defining a Date/Time Parameter* (on page 215).
   - **File** – Defines a data file parameter. For more information see *Defining a Data File* (on page 218). To create a new data file, see *Creating a Data File* (on page 222).
   - **Number** – Defines a number parameter. For more information see *Defining a Number Parameter* (on page 222).
   - **Random String** – Defines a random string parameter. For more information see *Defining a Random String Parameter* (on page 226).

The parameters definitions are stored with the script. You can change the parameters' definition at any time by using the Parameterization Manager again.

Defining a Date/Time Parameter

To define a date/time parameter:

1. In the Parameterization Manager Dialog Box (*Figure 132*), click **Add**. The Parameterization Manager dialog box opens.
2. In the **Type** field, select **Date/Time**. The fields appropriate for defining a date/time parameter appear in the dialog box.

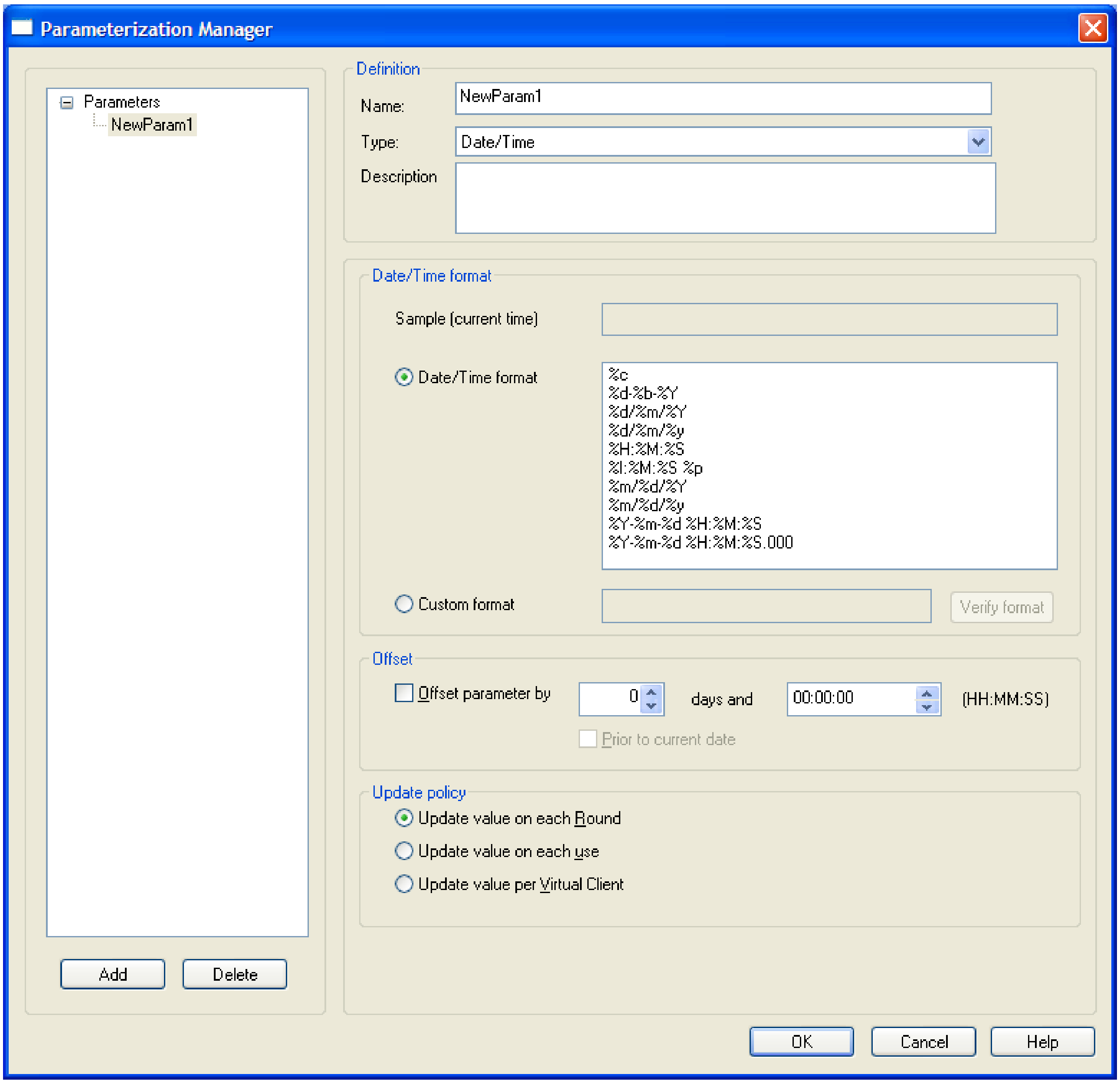


Figure 133: Parameterization Manager – Date/Time Dialog Box

3. In the **Description** field, optionally enter a description for the date/time parameter.
4. Fill in the fields as described in Table 33.
5. Click **OK**.

The following table describes the fields and buttons in the Parameterization Manager – Date/Time dialog box.

Table 33: Parameterization Manager – Date/Time Dialog Box Options

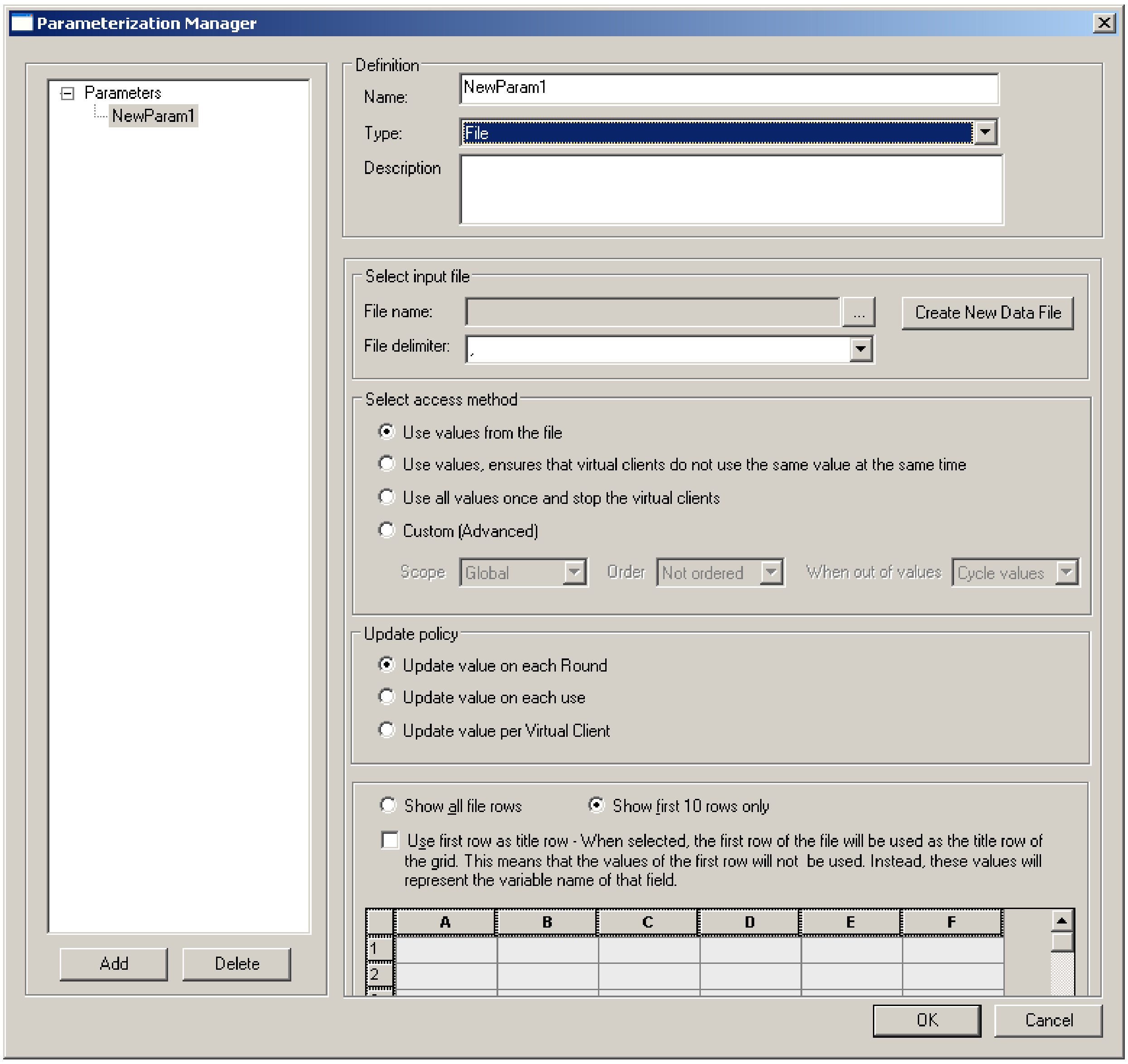
Setting	Description
Date/Time Format	
Sample (current time)	Shows the current time in the format you select in **Date/Time format**.
Date/Time format	Various predefined date/time formats. Select the desired format.
Custom format	Enables you to define a custom date/time format using the supported field types. The valid field options are: • %a – Abbreviated weekday name (such as, Fri). • %A – Full weekday name (such as, Friday). • %b – Abbreviated month name (such as, Oct). • %B – Full month name (such as, October). • %c – Standard date and time string (Sun Oct 17 04:41:13 2010). • %d – Day of the month (1-31). • %H – Hour, in 24-hour format (00-23). • %I – Hour, in 12-hour format (1-12). • %j – Day of the year (1-366). • %m – Month in numerical format (1-12). • %M – Minute (0-59). • %p –AM/PM. • %S – Second (0-59). • %U – Week of the year, (0-53), where week 1 has the first Sunday. • %w – Weekday in numerical format (0-6), where Sunday is 0. • %W – Week of the year, (0-53), where week 1 has the first Monday. • %x – Date representation, as preferred in your locale. • %X – Time representation, as preferred in your locale. • %y – Abbreviated year (0-99). • %Y – Full year (such as 2011). • %Z – Time zone name. • %% – Percent sign. In addition, you can enter any kind of separator between fields, including spaces, dashes, underscores, slashes, and periods.

Setting	Description
Verify Format	After entering a custom format, click this button to verify whether the format is valid: • If it is valid, a sample of the format's output is displayed in the **Sample (current time)** field. • If it is invalid, a popup window appears indicating that you must enter a valid value.
Offset	Specifies that the date/time parameter will not consider the current date and time but another date and time, in the future or in the past.
Offset parameter by	Determines by how many days and how much time to offset the current date.
Prior to current date	Specifies a negative offset (prior to the current day and time).
Update Policy	Defines when to update the parameter.
Update value on each Round	The virtual clients update the parameter once per round. Thus, if the same parameter appears again in the same round, it will get the same value.
Update value on each use	The virtual clients update the parameter's value each time it is used.
Update value per Virtual Client	The virtual clients update the parameter's value once at the beginning of the test (when running the InitClient function). All usage of the parameter by that virtual client will always use the same value.

Defining a Data File

To select a data file:

1. In the Parameterization Manager Dialog Box (*Figure 132*), click **Add**.
2. In the **Type** field, select **File**. The fields appropriate for selecting a data file and configuring its settings appear in the dialog box.



Parameterization Manager

Parameters
NewParam1

Definition
Name: NewParam1
Type: File
Description

Select input file
File name:
File delimiter: ,
Create New Data File

Select access method
[✓ Use values from the file]
Use values, ensures that virtual clients do not use the same value at the same time
Use all values once and stop the virtual clients
Custom (Advanced)
Scope: Global Order: Not ordered When out of values: Cycle values

Update policy
[✓ Update value on each Round]
Update value on each use
Update value per Virtual Client

Show all file rows [✓ Show first 10 rows only]

Use first row as title row - When selected, the first row of the file will be used as the title row of the grid. This means that the values of the first row will not be used. Instead, these values will represent the variable name of that field.

	A	B	C	D	E	F
1						
2						

Add Delete OK Cancel

Figure 134: Parameterization Manager – File Dialog Box

3. In the **Description** field, optionally enter a description of the file.
4. Fill in the fields as described in Table 34.
5. Click **OK**.

The following table describes the fields and buttons in the Parameterization Manager – File dialog box.

Table 34: Parameterization Manager – File Dialog Box Options

Setting	Description
Select Input File	
File Name	Enables selecting the data file. Click ... to select a data file.

Setting	Description
File Delimiter	The character separating the fields in each row of the data file.
Create New Data File	Enables creating a new data file. For information, see *Creating a Data File* (on page 222).
Select access method	Defines the method for reading the next value/row from the file. The predefined methods are the most common and useful methods.
Use values from the file	Use rows from the file without any specific restrictions. This is the recommended method to use when applicable. This method corresponds to the following Custom settings: Scope – *local*, Order – *random*, When Out of Values – *cycle*. For the explanations of Scope, Order and When Out of Values, see the explanations of the **Custom (Advanced)** option.
Use values, ensure that virtual clients do not use the same value at the same time	Use unique rows from the file so that a row cannot be used by two virtual clients at the same time. This is useful for example if the value is the login name, and the system under test does not allow the same user to be logged in twice. This method corresponds to the following Custom settings: Scope – *global unique*, Order – *random*, When Out of Values – *cycle*. For the explanations of Scope, Order and When Out of Values, see the explanations of the **Custom (Advanced)** option.
Use all values once and stop the virtual clients	Use each row once. When all rows have been used, the virtual clients will be stopped. This method corresponds to the following Custom settings: Scope – *global unique*, Order – *random*, When Out of Values – *stop virtual client*. For the explanations of the Scope, Order and When Out of Values parameters, see the explanations of the **Custom (Advanced)** option.
Custom (Advanced)	Enables selecting any combination of Scope, Order and When Out of Value settings.
Scope	Defines the scope (sharing policy) of values. • **Local** – Each virtual client reads rows from its own copy of the pool. • **Global unique** – All virtual clients read a unique row from a global pool, which is shared by all virtual clients on all load generators. A row cannot be used by two virtual clients at the same time. • **Global** – All virtual clients in the session read rows from the shared (global) pool. However, the rows are not necessarily unique – two virtual clients may happen to use the same row at the same time. Note that if you select Global, there is not much point in enforcing order on the values because all virtual clients run at the same time, so it is not possible to read the values efficiently in a certain order. Therefore, specify Random or Not-ordered in the **Order** field.

Setting	Description
Order	Defines the method for reading the next row from the file: • **Random** – Every virtual client gets a random row from the file. All available rows have the same probability of being selected at any given point. • **Not Ordered** – Every virtual client gets a random row from among the rows that have been used less times. Over time, all rows are used approximately the same number of times. • **Ordered** – Every virtual client gets the next row from the file (sequential order). If necessary, the file is read through many times. Select this option only if sequential order is crucial for the application. When running more than one virtual client concurrently, the order of execution is anyway not defined, therefore this option is discouraged. Note that specifying *Ordered* in conjunction with a *Global* or *Global Unique* Scope and *Cycle* When Out of Values, has unavoidable performance costs.
When out of values	Defines whether the rows can be used any number of times, or only once. • **Cycle values** – Each row can be used any number of times. • **Stop virtual client** – After each row was used once, stop any virtual client that requests another row. An error message is written to the monitor log window. • **Keep last value** – After each row was used once, keep re-using the last value.
Update Policy	Defines when a parameter is updated, meaning when a new row is read. • **Update value on each Round** – A virtual client reads a new row from the file per round. Thus, if the same parameter appears again in the same round, it will get the same value. • **Update value on each use** – A virtual client reads a parameter's row each time it is used. • **Update value per Virtual Client** – A virtual clients reads a new row from the file when initialized (when running the InitClient function). All usage of a parameter by that virtual client will always use the same value.
Show all file rows / Show first 10 rows only	Determines which rows the grid displays.
Use first row as title row	Uses the first row of the file as the title row. If you select this option, the values of the first row are not used as data but as parameter names. For further explanations, refer to *Inserting User-Defined Parameters in a Script* (on page 228).

Creating a Data File

You can create a new data file.

To create a data file:

1. From the Parameterization Manager – File dialog box (Figure 134), click **Create New Data File**.

   The Create Data File dialog box appears.

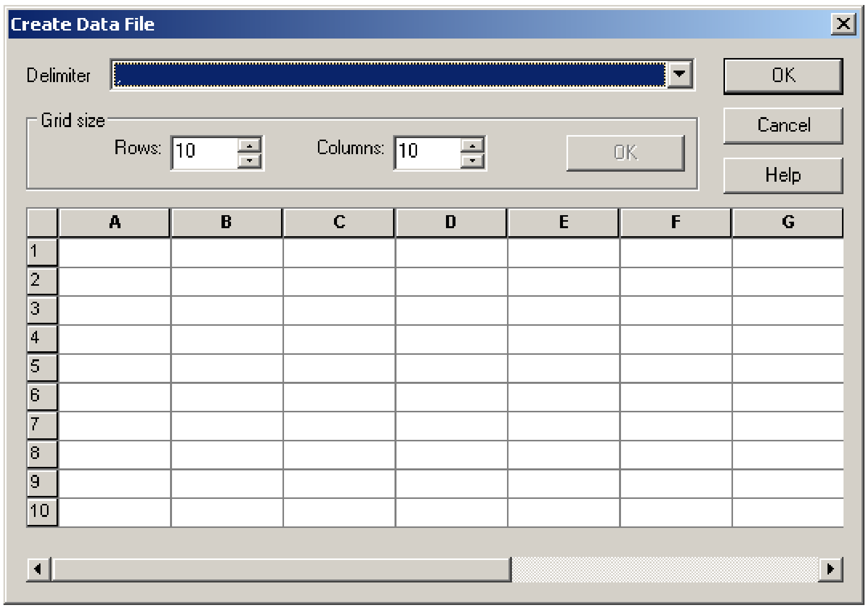


Figure 135: Create Data File Dialog Box

2. Select a file delimiter from the drop-down list.
3. Type the number of rows in the Rows field. The default is 10 rows.
4. Type the number of columns in the Columns field. The default is 10 columns.
5. If you did not use the default values, click **OK**.
6. In the table, type a value in each cell.
7. Click **OK**.

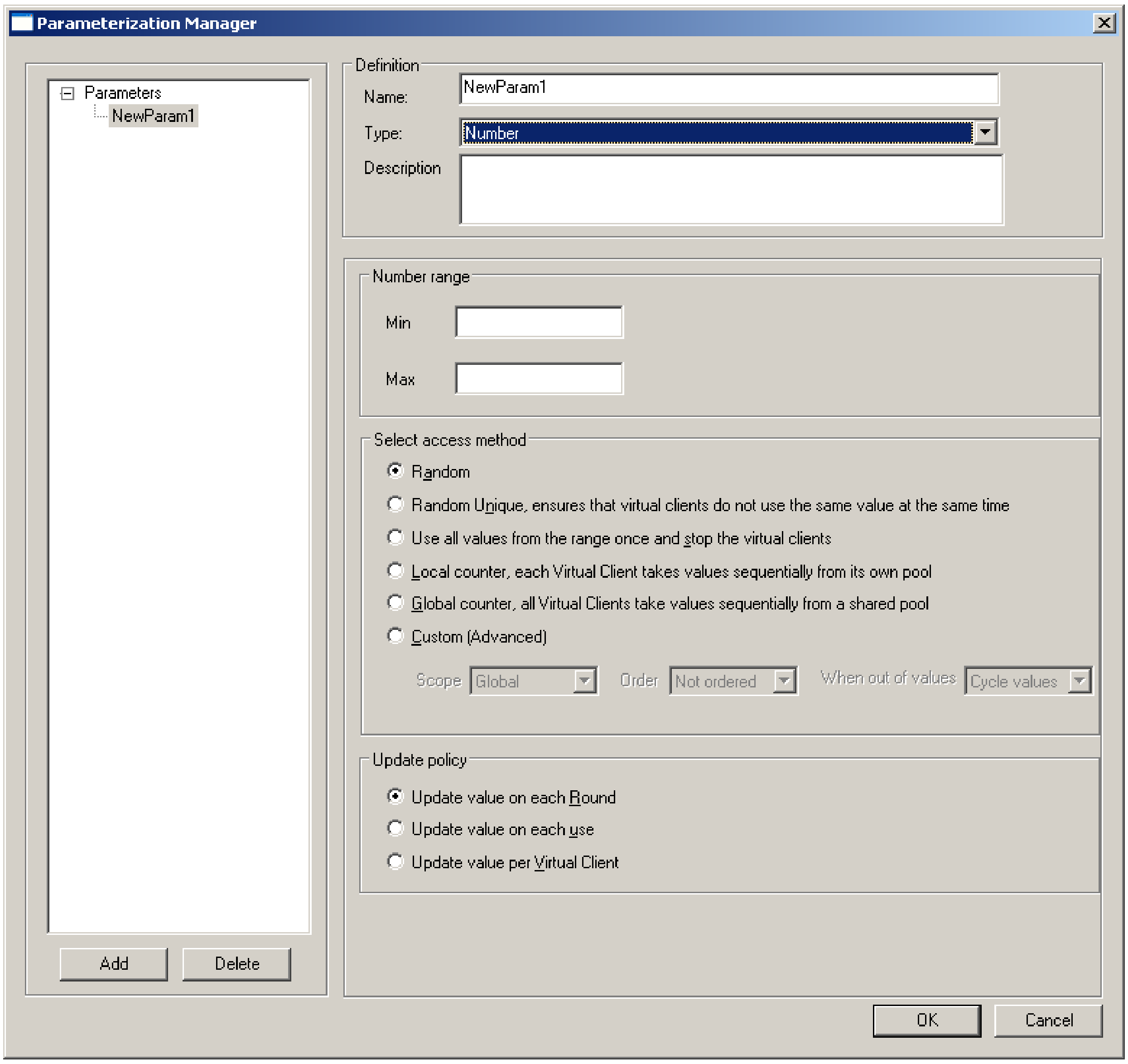
   A Save As dialog box appears. Save the new data file.

Defining a Number Parameter

To define a number parameter:

1. In the Parameterization Manager Dialog Box (*Figure 132*), click **Add**. The Parameterization Manager dialog box opens.

2. In the **Type** field, select **Number**. The fields appropriate for defining a number parameter appear in the dialog box.


Parameterization Manager

Parameters
NewParam1

Definition
Name: NewParam1
Type: Number
Description

Number range
Min
Max

Select access method
(●) Random
( ) Random Unique, ensures that virtual clients do not use the same value at the same time
( ) Use all values from the range once and stop the virtual clients
( ) Local counter, each Virtual Client takes values sequentially from its own pool
( ) Global counter, all Virtual Clients take values sequentially from a shared pool
( ) Custom (Advanced)

Scope: Global
Order: Not ordered
When out of values: Cycle values

Update policy
(●) Update value on each Round
( ) Update value on each use
( ) Update value per Virtual Client

Figure 136: Parameterization Manager – Number Dialog Box

3. In the **Description** field, optionally enter a description of the number parameter.
4. Fill in the fields as described in Table 35.
5. Click **OK**.

The following table describes the fields and buttons in the Parameterization Manager – Number dialog box.

Table 35: Parameterization Manager – Number Dialog Box Options

Setting	Description
Number range	
Min	The minimum value for the number range.
Max	The maximum value for the number range.
Select access method	Defines the method for determining the next number value. The predefined methods are the most common and useful methods.
Random	Use random numbers freely. This method corresponds to the following Custom settings: Scope – *local*, Order – *random*, When Out of Values – *cycle*. For the explanations of Scope, Order and When Out of Values, see the explanations of the **Custom (Advanced)** option.
Random Unique, ensures that virtual clients do not use the same value at the same time	Use unique numbers. A number cannot be used by two virtual clients at the same time. This method corresponds to the following Custom settings: Scope – *global unique*, Order – *random*, When Out of Values – *cycle*. For the explanations of Scope, Order and When Out of Values, see the explanations of the **Custom (Advanced)** option.
Use all values from the range once and stop the virtual clients	Use each number once. When all numbers in the range have been used, the virtual clients will be stopped. This method corresponds to the following Custom settings: Scope – *global unique*, Order – *random*, When Out of Values – *stop virtual client*. For the explanations of Scope, Order and When Out of Values, see the explanations of the **Custom (Advanced)** option.
Local counter, each Virtual Client takes values sequentially from its own pool.	Each virtual client will pass through the numbers in the range. This method corresponds to the following Custom settings: Scope – *local*, Order – *ordered*, When Out of Values – *cycle*. For the explanations of Scope, Order and When Out of Values, see the explanations of the **Custom (Advanced)** option.
Global counter, all Virtual Clients take values sequentially from a shared pool	Use increasing integer values. Each value can be used only once. When the whole range is used, the virtual clients are stopped. This method corresponds to the following Custom settings: Scope – *global unique*, Order – *ordered*, When Out of Values – *stop virtual client*. For the explanations of Scope, Order and When Out of Values, see the explanations of the **Custom (Advanced)** option.

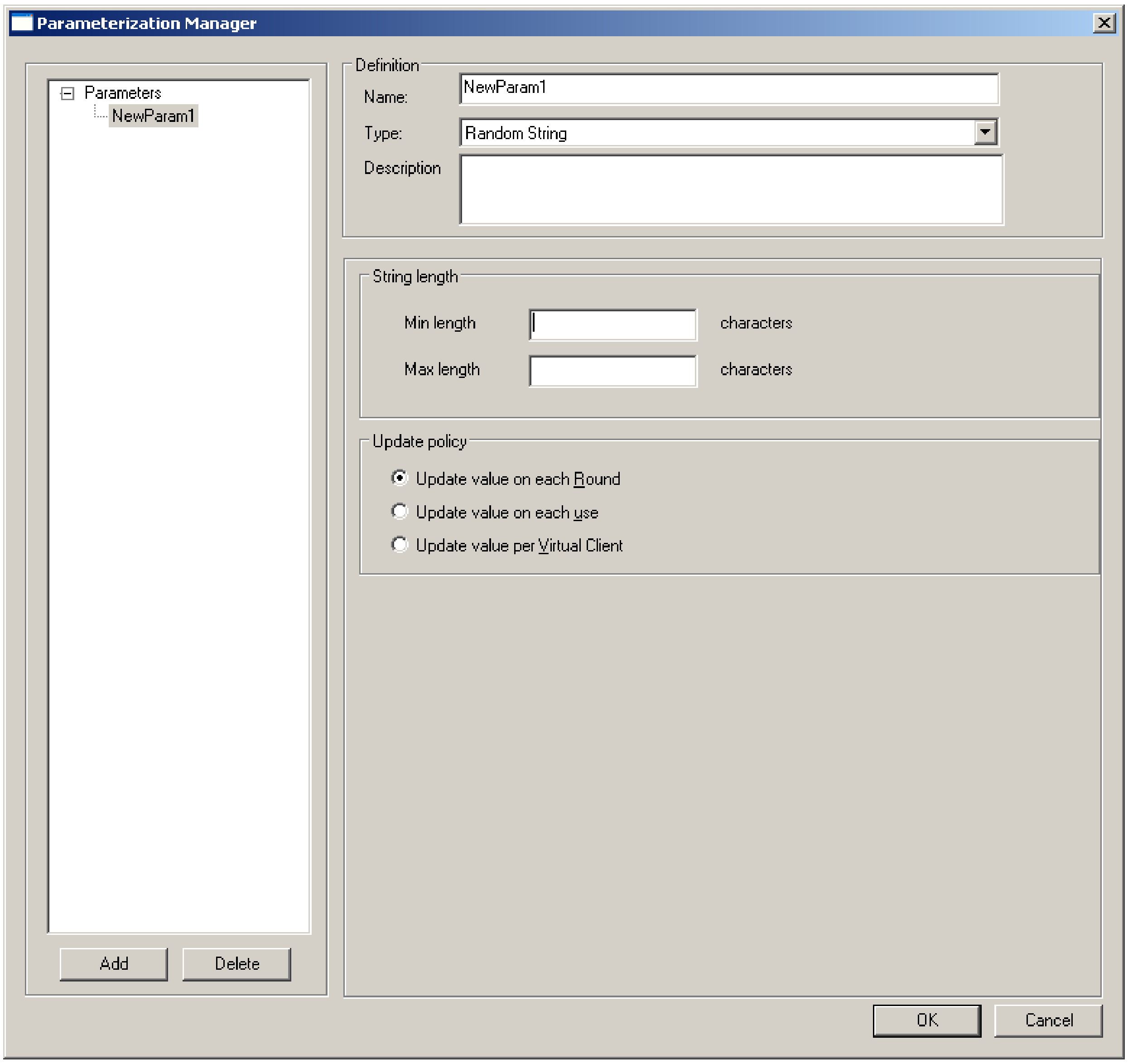
Setting	Description
Custom (Advanced)	Enables selecting any combination of Scope, Order and When Out of Value settings.
Scope	Defines the scope (sharing policy) of values. • **Local** – Each virtual client reads values from its own copy of the pool. • **Global unique** – All virtual clients read a unique value from a global pool, which is shared by all clients on all load generators. A value cannot be used by two virtual clients at the same time. • **Global** – All virtual clients in the session read values from the shared (global) pool. However, the values are not necessarily unique – two virtual clients may happen to use the same value at the same time. Note that if you select Global, there is not much point in enforcing order on the values because all virtual client run at the same time, so it is not possible to read the values efficiently in a certain order. Therefore, specify Random or Not-ordered in the **Order** field.
Order	Defines the method for determining the next number value: • **Random** – Every virtual client gets a random number value. All available values have the same probability of being selected at any given point. • **Not Ordered** – Every virtual client gets a random number value from among the values that have been used less times. Over time, all rows are used approximately the same number of times. • **Ordered** – Every virtual client gets the next number value. If necessary, the sequence of numbers is gone through many times. Select this option only if sequential order is crucial for the application. In general, this option is not recommended Note that specifying *Ordered* in conjunction with a *Global* or *Global Unique* scope and *Cycle* When Out of Values, has unavoidable performance costs.
When out of values	Defines whether the values can be used any number of times, or only once. • **Cycle values** – Each value can be used any number of times. • **Stop virtual client** – After each value was used once, stop any virtual client that requests another value. An error message is written to the monitor log window. • **Keep last value** – After each value was used once, keep re-using the last value.

Setting	Description
Update Policy	Defines when a parameter is updated, meaning when a new value is read. • **Update value on each Round** – A virtual client reads a new value per round. Thus, if the same parameter appears again in the same round, it will get the same value. • **Update value on each use** – A virtual client reads a parameter's value each time it is used. • **Update value per Virtual Client** – A virtual clients reads a new value when initialized (when running the InitClient function). All usage of a parameter by that virtual client will always use the same value.

Defining a Random String Parameter

To define a random string parameter:

1. In the Parameterization Manager Dialog Box (*Figure 132*), click **Add**. The Parameterization Manager dialog box opens.
2. In the **Type** field, select **Random String**. The fields appropriate for defining a random string parameter appear in the dialog box.



Name: NewParam1

Type: Random String

Update policy:
- [✓ Update value on each Round]
- Update value on each use
- Update value per Virtual Client

Figure 137: Parameterization Manager – Random String Dialog Box

3. In the **Description** field, optionally enter a description of the random string parameter.
4. Fill in the fields as described in Table 36.
5. Click **OK**.

The following table describes the fields and buttons in the Parameterization Manager – Random String dialog box.

Table 36: Parameterization Manager – Random String Dialog Box Options

Setting	Description
String length	
Min	The minimum length of the string, in number of characters.

Setting	Description
Max	The maximum length of the string, in number of characters.
Update Policy	Defines when to update the parameter, meaning when the virtual clients get a new value for the parameter. • **Update value on each Round** – A virtual clients reads a new value per round. Thus, if the same parameter appears again in the same round, it will get the same value. • **Update value on each use** – A virtual clients reads a parameter's value each time it is used. • **Update value per Virtual Client** – A virtual clients reads a new value when initialized (when running the InitClient function). All usage of the parameter by that virtual client will always use the same value.

Note: Using a random string parameter in a script does not provide unique values. If you need unique values, or special formatting of the string, create a data file with unique values and use File parameterization (see *Defining a Data File* on page 218).

Inserting User-Defined Parameters in a Script

WebLOAD Recorder enables you to edit parameters having static values and replace the static values with a call to a set of specified values. During runtime, the script runs the parameter using values from the set.

The first step is to use the Parameterization Manager to define the set of values (see *Configuring the Parameterization Manager* on page 213*)*. The set of values is a type of parameter (Number parameter, String parameter, Date/Time parameter or Data File parameter). The second step, described in this section, is to replace a static value in the script with a call to the defined parameter.

To insert a user-defined parameter in a script:

1. In the main window, click **Open** in the **File** tab of the ribbon, and open the script you want to edit.
2. In the JavaScript View pane, select the static value you want to replace.
   For example, in the line :

   ```
   wlHttp.FormData["name"] = "john"
   ```

   select `"John"`.
3. Right-click and select **Insert Variable**.

   The Insert Variable menu appears (Figure 54).

4. Select the parameter you defined in the Parameterization Manager.

   The selected parameter replaces the static value in the script.
   In our example, if you selected `Users_firstname.getValue()` from the Insert Variable menu, the line now shows:

   ```
   wlHttp.FormData["name"] = Users_firstname.getValue();
   ```

Note that if you are using a parameter from a data file, the parameter name reflects whether the data file includes a title row.

- If the data file includes a title row, the parameter name is of type:
  `{Parameter name}_{column title}.getValue().`
- If the data file does not include a title row, the column number is used, and the parameter name is of type:
  `{Parameter name}_Col{column number}.getValue()`

Note: To replace multiple occurrences of a static value, you can use the **Edit ➤ Replace** tool.

Example of Using User-Defined Parameters in a Script

If the original recorded script includes:

```
wlHttp.FormData["first_name"] = "John"
wlHttp.FormData["last_name"] = "Smith"
wlHttp.FormData["age"] = "47"
```

and you wish to replace the static values (`John`, `Smith`, `47`) with parameters, you can define a random number parameter 'Age', and a file parameter that calls the 'Users' data file having columns 'firstName' and 'lastName'.

Using the Insert Variable menu, modify the script as follows:

```
wlHttp.FormData["first_name"] = Users_firstName.getValue();
wlHttp.FormData["last_name"] = Users_lastName.getValue();
wlHttp.FormData["age"] = Age.getValue();
```

Appendix A

The WebLOAD Recorder Toolbox Set

This section describes the WebLOAD Recorder toolbox set.

The WebLOAD Recorder Toolbox Items

The following are the WebLOAD Recorder Toolbox items:

Table 37: Toolbox Items

Toolbox Items	
General	
• Sleep	• Message
• JavaScriptObject	• Comment
• Try	• Catch
Load	
• Begin Transaction	• End Transaction
• Set Timer	• Send Timer
• Synchronization Point	• Send Measurement
• URL Screening	• Value Extraction
• Define Concurrent	• Execute Concurrent
Internet Protocols	
• FTP-connect	• FTP-upload
• FTP-download	• FTP-disconnect
• SMTP-send message	• POP-retrieve
• POP-Delete	• IMAP-Connect
• IMAP-Retrieve	• IMAP-Delete
• IMAP-CreateMailbox	• IMAP-ListMailboxes
• IMAP-DeleteMailbox	• IMAP-RenameMailbox
• IMAP-SubscribeMailbox	• IMAP-UnsubscribeMailbox

Toolbox Items	
• IMAP-ListSubscribedMailbox	• IMAP-Search
• NNTP-Connect	• NNTP-GetArticle
• NNTP-GetArticleCount	• NNTP-PostArticle
• TCP-Connect	• TCP-Send
• TCP-Receive	• TCP-Erase
• TELNET-Connect	• TELNET-Receive
• TELNET-Send	• TELNET-Erase
• UDP-Bind	• UDP-Broadcast
• UDP-Receive	• UDP-Send
• UDP-Erase	• LDAP-Bind
• LDAP-Search	• LDAP-UnBind
IoT Protocols	
• MQTT-Connect	• MQTT-Subscribe
• MQTT-Send	• MQTT-GetMessages
• AMQP-Connect	• AMQP-Send
• AMQP-GetMessages	
Database	
• OpenDB	• Oracle OpenDB
• MySQL OpenDB	• Execute Command
• Fetch Data	• DB GetLine
• Oracle DB GetLine	• MySQL DB GetLine
• DB Load	• Oracle DB Load
• MySQL DB Load	
Verifications	
• WS-SingleNode	• WS-MultipleNodes
• Flex:Verify-Ext	• Flex:Extract-Ext
Multimedia	
• Streaming-Create	• Streaming-Play
• Streaming-Play with range	• Streaming-Wait for Media and Stop
• Streaming-Wait for Media and Pause	• Streaming-Close
WebSocket	

Toolbox Items	
• WebSocket Connect	• WebSocket Send
• WebSocket Close	
Web Services	
• HTTP	• WSDL
JMS	
• JMS-Connect-JNDI	• JMS-Connect-HornetQ
• JMS-Send	• JMS-Receive
Real Clients	
• Selenium Driver	• Selenium Report Statistics
• JUnit Runner	• PerfectoMobile

The WebLOAD Recorder General Toolbox

The following table describes the purpose of each of the WebLOAD Recorder General Toolbox items:

Table 38: General Toolbox Items

Script Item	Purpose
Sleep	Emulates the time it takes users to get from one page to the next - includes download time and the time it takes to read the page.
Message	Places an informational or error message in the script. This message will appear in the Log window when you play back the script. Message objects can also be used to print out the values of variables.
JavaScript Object	Enables you to insert JavaScript code directly into the script. You can code directly in JavaScript. To add a JavaScript Object to the script, drag the **JavaScript Object** icon into the Script Tree, and then open the object to insert JavaScript code.
Comment	Places comments in your script. The comment will appear in the JavaScript View pane when viewing the entire script.
Try…Catch	Places Try and Catch statements in your script. You can use the Try...Catch statements for structured exception handling.

Sleep

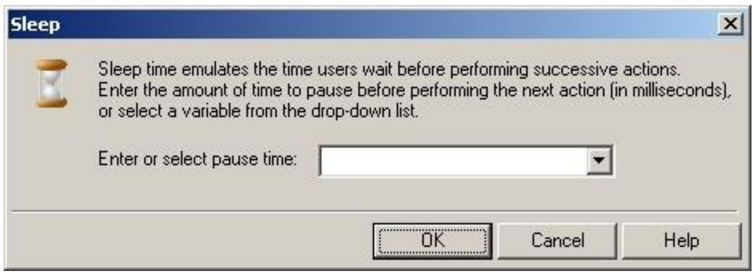
Users vary their activity when accessing a Web application, sometimes pausing between transactions and occasionally only accessing the server intermittently. The time a user waits between performing consecutive actions is known as sleep time.

When you record a script, WebLOAD Recorder automatically records the actual sleep time and inserts sleep icons into the script. You can edit the recorded sleep times manually, add more sleep statements, and control how WebLOAD is influenced by the sleep timers in the script.

To insert sleep timers:

1. Drag the **Sleep** icon from the General toolbox into the Script Tree at the desired location.

   The Sleep dialog box opens.



Sleep

Sleep time emulates the time users wait before performing successive actions. Enter the amount of time to pause before performing the next action (in milliseconds), or select a variable from the drop-down list.

Enter or select pause time:

Figure 138: Sleep Dialog Box

2. In the **Enter or select pause time** field, enter or select the duration of the sleep. The default value is 1000 milliseconds.

   The Sleep item appears in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

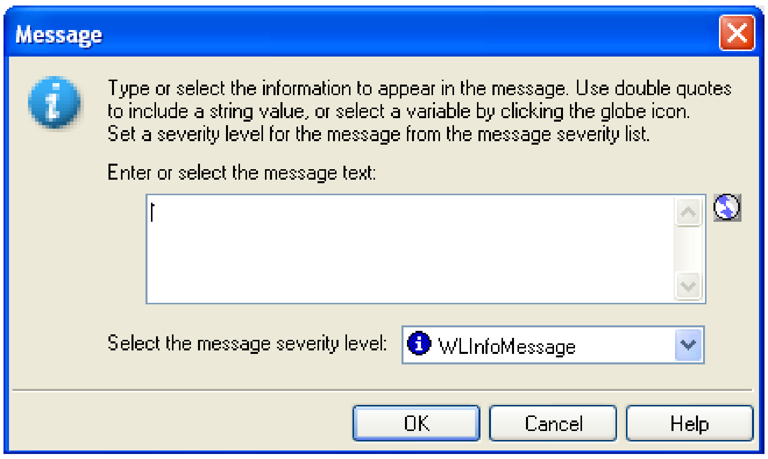
Message

While running a test session, WebLOAD Recorder and WebLOAD Recorder’s Log windows display information about session execution. You can include Message nodes in your script, defining points at which to send error and/or notification messages to the Log window.

To insert a message:

1. Drag the **Message** icon from the General toolbox into the Script Tree at the desired location.

   The Message dialog box opens.


Message

Type or select the information to appear in the message. Use double quotes to include a string value, or select a variable by clicking the globe icon. Set a severity level for the message from the message severity list.

Enter or select the message text:

Select the message severity level: WLInfoMessage

Figure 139: Message Dialog Box

2. Create a text message by typing the text you want to appear in the message into the input text box.

Note: When entering a string value to the message, the string must be enclosed in quotation marks; for example, "Sample Message".

3. To add a global variable to the message text, click the **globe** icon to the right of the input text box and select a global variable from the drop-down list.
4. Select a severity level for the message from the drop-down list.

   The following severity levels are available:

   - Information message (WLInfoMessage)
   - Minor error message (WLMinorError)
   - Error message (WLError)
   - Severe error message (WLSevereError)
   - Debug message (WLDebugMessage)
5. Click **OK**.

   The Message item appears in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

JavaScriptObject

JavaScript Objects enable you to insert JavaScript code directly into the script, giving you access to advanced functionality not available through the WebLOAD Recorder graphic interface. For example, working with XML or COM, or retrieving data from a database, are all tasks that require some additional programming code.

To insert a JavaScriptObject:

1. Drag the **JavaScriptObject** icon from the General toolbox into the Script Tree at the desired location.

   The JavaScriptObject item appears in the Script Tree and the WebLOAD Recorder protocol block is added to the script.

2. Open the object in JavaScript Editing mode to insert JavaScript code, as described in *Using the JavaScript Editor* (on page 75).

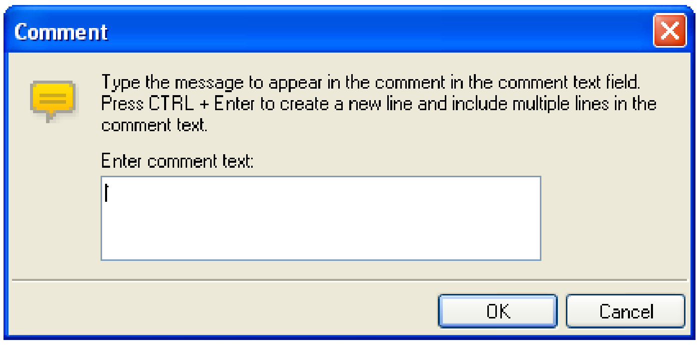
Comment

WebLOAD Recorder enables you to add comments to your script to describe an activity or provide information about a specific operation.

To insert a comment:

1. Drag the **Comment** icon from the General toolbox into the Script Tree at the desired location.

   The Comment dialog box opens.


Comment

Type the message to appear in the comment in the comment text field. Press CTRL + Enter to create a new line and include multiple lines in the comment text.

Enter comment text:

Figure 140: Comment Dialog Box

2. Enter the text you want to appear in the comment.
3. Click **OK**.

The Comment item appears in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Try / Catch Statements

You can use the Try…Catch statements for structured exception handling. This enables you to execute a particular block of statements if a specified exception occurs while your code is running.

To insert a Try...Catch statement:

1. Drag the **Try** icon from the General toolbox into the Script Tree directly before the first action you want to include in the Try...Catch block.
2. Drag the **Catch** icon from the General toolbox into the Script Tree directly after the last action you want to include in the Try...Catch block.

   The Try and Catch items appear in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The WebLOAD Recorder Load Toolbox

The following table describes the purpose of each of the WebLOAD Recorder Load Toolbox items:

Table 39: Load Toolbox Items

Script Item	Purpose
Begin Transactions End Transactions	Adds named transactions to the script to measure the performance of logical actions in your script, such as a Login process. By inserting named transactions into your script, you can take a series of simple actions, define them as a single transaction, and set success or failure criteria for the complete transaction.
Set Timer Send Time	Timers let you time any operation or group of operations in a script and send the time statistics to the WebLOAD Console.
SynchronizationPoint	Create peak server loads that stress your system to the limit by deliberately forcing multiple Virtual Clients to perform key tasks and execute a given command at precisely the same moment in real time.

Script Item	Purpose
SendMeasurement	Create a new measurement name and assign a value to the measurement available in WebLOAD reports.
URL Screening	Specify the URLs the WebLOAD engine should ignore (not fetch).
ValueExtraction	Extract a value from a string using a prefix and suffix.
DefineConcurrent	Define a starting point after which the WebLOAD engine collects all Post and Get HTTP requests, but does not execute them, until an `Execute Concurrent` function is run.
ExecuteConcurrent	Defines a starting point after which the WebLOAD engine stops collecting and begins executing all the Post and Get HTTP requests that were defined since the last `Define Concurrent` function, concurrently (using multithreading).

Begin and End Transaction

In addition to the automatic transactions provided by WebLOAD, you can use the WebLOAD Recorder GUI to easily add named transactions to the script to measure the performance of logical actions in your script, such as a Login process. By inserting named transactions into your script, you can take a series of simple actions, define them as a transaction, and set success or failure criteria for the transaction. Each transaction can be a simple action, such as a query, or a complex action that may include several steps.

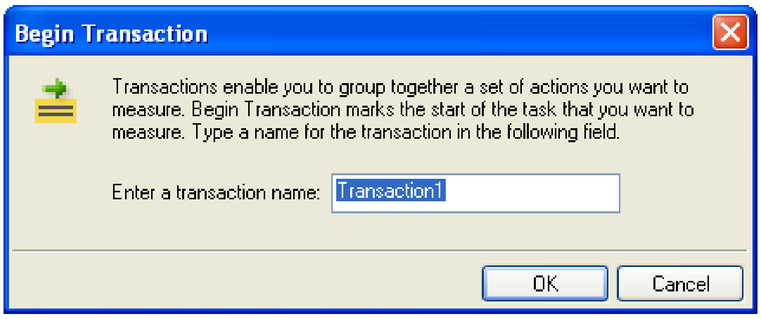
To measure transactions, you must mark the beginning and end of the transaction in your script. During runtime, WebLOAD measures the time it takes to complete the transaction and reports the results in the WebLOAD Integrated reports, Statistics reports, and Data Drilling report.

Note: You can add an unlimited number of transactions into your script, each with a different name.

To mark the beginning of a transaction:

1. Drag the **Begin Transaction** icon from the Load toolbox into the Script Tree, directly above the first action you want to include in the transaction.

   The Begin Transaction dialog box opens.



Begin Transaction

Transactions enable you to group together a set of actions you want to measure. Begin Transaction marks the start of the task that you want to measure. Type a name for the transaction in the following field.

Enter a transaction name: Transaction1

Figure 141: Begin Transaction Dialog Box

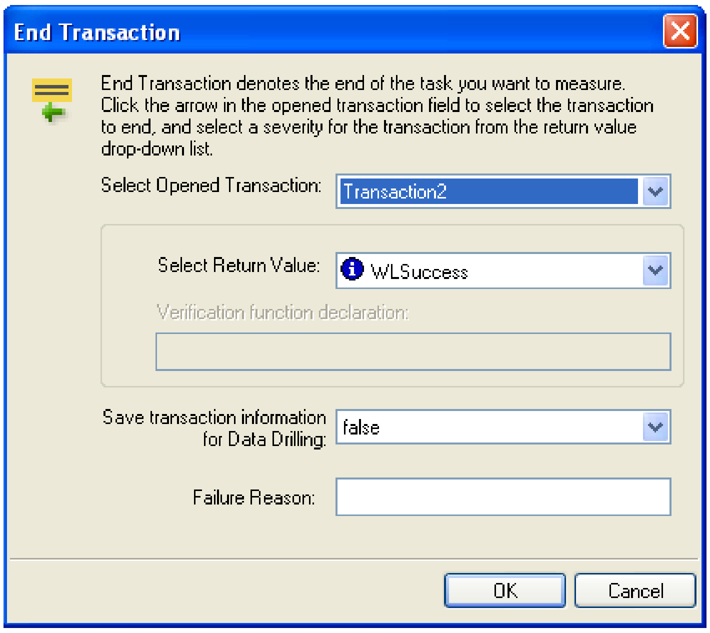
2. Enter a logical name for the transaction; for example, Login.
3. Click **OK**.

   The Begin Transaction item appears in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

To mark the end of a transaction:

1. Drag the **End Transaction** icon from the Load toolbox into the Script Tree, directly after the last action you want included in the script.

   The End Transaction dialog box opens.



End Transaction

End Transaction denotes the end of the task you want to measure. Click the arrow in the opened transaction field to select the transaction to end, and select a severity for the transaction from the return value drop-down list.

Select Opened Transaction: Transaction2

Select Return Value: WLSuccess

Verification function declaration:

Save transaction information for Data Drilling: false

Failure Reason:

Figure 142: End Transaction Dialog Box

2. Select the transaction to end from the Select Opened Transaction drop-down list.
3. Select a return value for the transaction from Select Return Value drop-down list.

You can select from the return values provided, or select Custom Function to create your own verification function to call when the transaction is complete.

For information on creating custom functions, see the *WebLOAD Scripting Guide*.

4. To set WebLOAD to save the results of all transaction instances, successes and failures, for later analysis with Data Drilling, select **true** in the Save transaction information for Data Drilling field. Select **false** (default) to save only results of failed transaction instances that triggered some sort of error flag.
5. Optionally, enter a text string to specify a possible reason for a transaction failure within your transaction verification function in the Failure Reason field. This reason will also appear in the Statistics Report.
6. Click **OK**.

   The End Transaction item appears in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Set and Send Timer

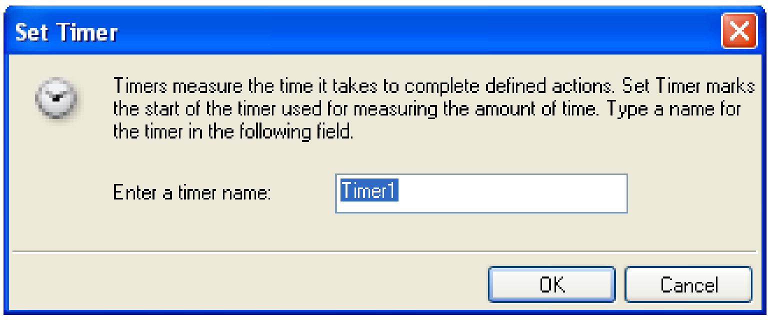
Timers let you time any operation or group of operations in a script and send the time statistics to the WebLOAD Console. For example, you can add a timer to measure the amount of time needed to complete a series of user activities on a single Web page. You can add timers to a script directly through the WebLOAD Recorder.

Note: When you set a timer, it is automatically zeroed.

To mark the beginning of a timer:

1. Drag the **Set Timer** icon from the Load toolbox into the Script Tree directly before the first action you want to include in the timed task.

   The Set Timer dialog box opens.



Set Timer

Timers measure the time it takes to complete defined actions. Set Timer marks the start of the timer used for measuring the amount of time. Type a name for the timer in the following field.

Enter a timer name: Timer1

Figure 143: Set Timer Dialog Box

2. Type a name for the timer in the Enter a timer name field.

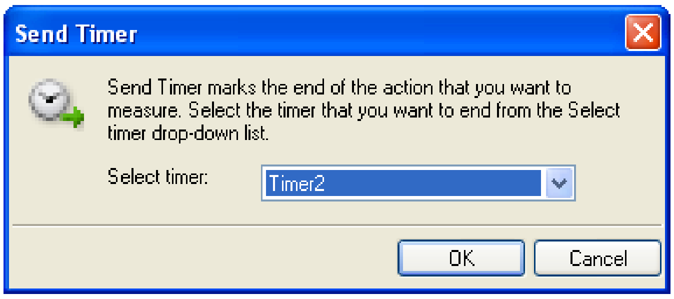
3. Click **OK**.

   The Set Timer item appears in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

To mark the end of the timer:

1. Drag the **Send Timer** icon from the Load toolbox into the Script Tree directly after the last action you want included in the timed task.

   The Send Timer dialog box opens.


Send Timer

Send Timer marks the end of the action that you want to measure. Select the timer that you want to end from the Select timer drop-down list.

Select timer: Timer2

Figure 144: Send Timer Dialog Box

2. From the Select Timer drop-down list, to select the timer to end.
3. Click **OK**.

   The Send Timer item appears in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Synchronization Point

During a test session, WebLOAD simulates the random nature of the real world, where even with hundreds or thousands of Web site hits, users do not all necessarily execute the same commands at precisely the same instant. However, for testing purposes, you may wish to create peak server loads that stress your system to the limit by deliberately forcing multiple Virtual Clients to perform key tasks and execute a given command at precisely the same moment in real time.

WebLOAD provides Synchronization Points to coordinate the actions of multiple Virtual Clients. A Synchronization Point is a meeting place where Virtual Clients wait before continuing with a script. When one Virtual Client arrives at a Synchronization Point, WebLOAD holds the Client at that point until all the other Virtual Clients arrive. When all the Virtual Clients have arrived, they are all released at once to perform the next action in the script simultaneously.

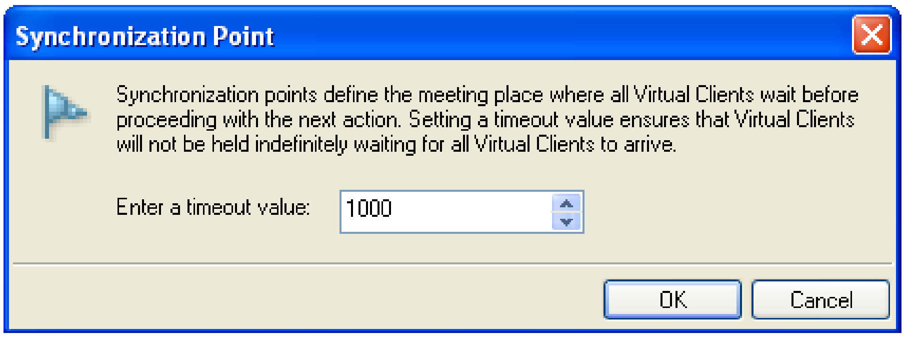
For example, suppose that you want to simulate 500 users, all trying to access a form on the same Web page simultaneously. To maximize the impact of this test situation, all 500 Virtual Clients must access the form at exactly the same time. Add a Synchronization Point before the form entry node to ensure that all the Virtual Clients log in simultaneously.

WebLOAD Recorder enables you to define the meeting place where all Virtual Clients wait. You can also optionally set the timeout value, the number of milliseconds that WebLOAD will wait for all of the Virtual Clients to arrive at the Synchronization Point. The timeout is a safety mechanism that prevents an infinite wait if any of the Virtual Clients does not arrive at the Synchronization Point for any reason. Once the timeout period expires, WebLOAD releases the rest of the Virtual Clients. Setting a timeout value is important to ensure that the test session will not 'hang' indefinitely in case of error.

To insert a Synchronization Point:

1. Drag the **Synchronization Point** icon from the Load toolbox into the Script Tree directly before the action you want all Virtual Clients to perform simultaneously.

   The Synchronization Point dialog box opens.


Synchronization Point

Synchronization points define the meeting place where all Virtual Clients wait before proceeding with the next action. Setting a timeout value ensures that Virtual Clients will not be held indefinitely waiting for all Virtual Clients to arrive.

Enter a timeout value: 1000

Figure 145: Synchronization Point Dialog Box

2. In the Timeout Value field, enter or select a timeout value for the Synchronization Point. The default value is 1000 milliseconds.

   The Synchronization Point item appears in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

During a test session, the `SynchronizationPoint()` function returns a value to WebLOAD. This value indicates whether the function was successful or not. All failures are logged and displayed in the WebLOAD and Console Log windows, similar to any other WebLOAD test failure.

Synchronization Point function calls may return one of the following return values:

- `WLSuccess`—synchronization succeeded. All Virtual Clients arrived at the Synchronization Point and were released together.
- `WLLoadChanged`—synchronization failed. A change in the load size was detected while Virtual Clients were being held at the Synchronization Point. All Virtual clients were released.
- `WLTimeout`—synchronization failed. The timeout expired before all Virtual Clients arrived at the Synchronization Point. All Virtual Clients were released.
- `WLError`—synchronization failed. Invalid timeout value. All Virtual Clients were released.

For a complete explanation and example of the `SynchronizationPoint` function syntax, see WebLOAD Actions, Objects, and Functions, in the *WebLOAD JavaScript Reference Guide*.

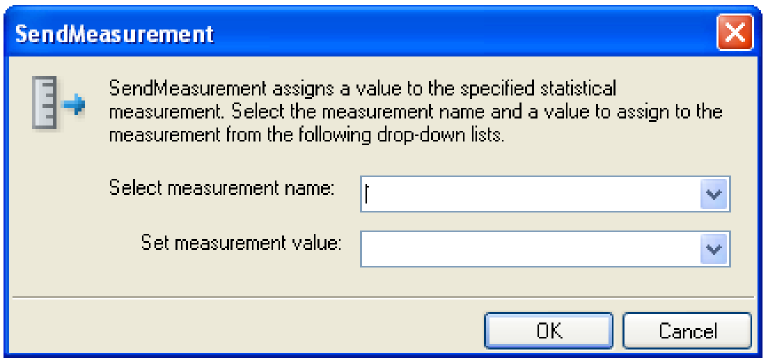
Send Measurement

WebLOAD Recorder enables you to insert Send Measurement actions into your script to create a new measurement name and assign a value to the measurement. During runtime the measurement is displayed in the WebLOAD statistics report.

To create and set the value for a measurement:

1. Drag the **Send Measurement** icon from the Load toolbox into the Script Tree at the desired location.

   The Send Measurement dialog box opens.



SendMeasurement

SendMeasurement assigns a value to the specified statistical measurement. Select the measurement name and a value to assign to the measurement from the following drop-down lists.

Select measurement name:

Set measurement value:

Figure 146: Send Measurement Dialog Box

2. Type or select a name for the measurement in the Select measurement name field.
3. Type or select a value for the measurement in the Set measurement value field.
4. Click **OK**.

The Send Measurement item appears in the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

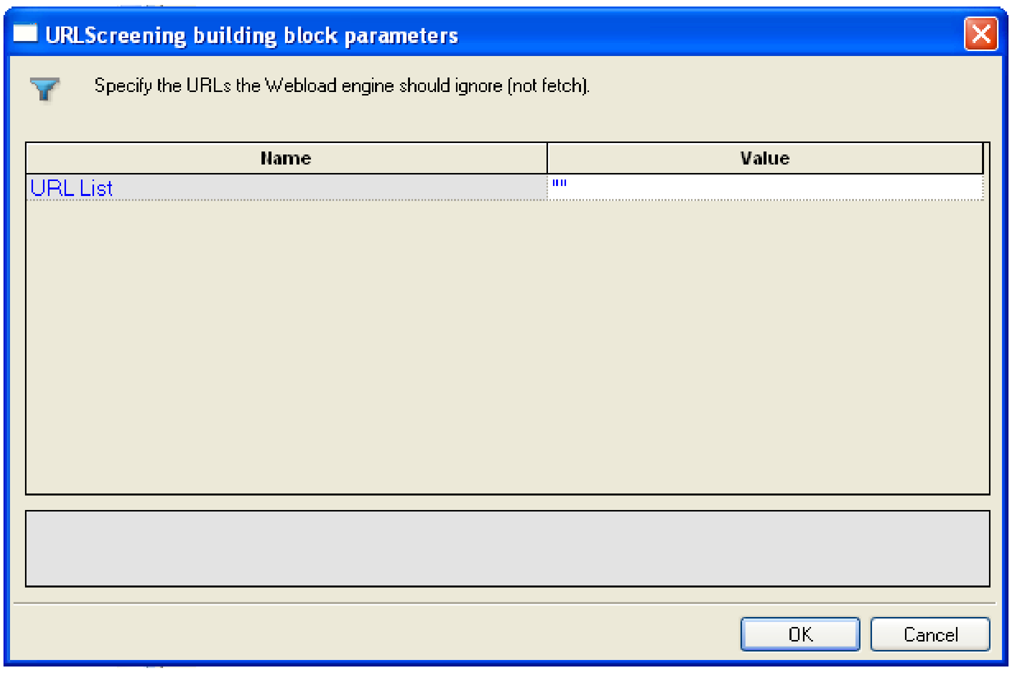
URL Screening

WebLOAD Recorder enables you to add URL screening to a script to define the URLs that the WebLOAD protocol engine should ignore during runtime. The ability to ignore links on the page being tested is a useful feature. For example, many Web sites include links to external sites. If these sites are not relevant to the testing requirements, they should be ignored. Other links may be to advertisement sites that charge a fee every time the link is accessed. Hitting these links during a typical load test that may run hundreds or thousands of iterations would be a tremendous waste, so these links should also be ignored.

To add URL screening to a script:

1. Drag the **URL Screening** icon, from the Load toolbox, into the Script Tree at the desired location.

   The URL Screening Building Block parameters dialog box opens.


URLScreening building block parameters

Specify the URLs the Webload engine should ignore (not fetch).

Name	Value
URL List	""

Figure 147: URL Screening Building Block Parameters Dialog Box

2. Enter the URLs to ignore, separated by commas, in the Value field.
3. Click **OK**.

The URL Screening Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()` function, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Note: Fields that were not assigned a value in the dialog box are left as empty fields in the script code.

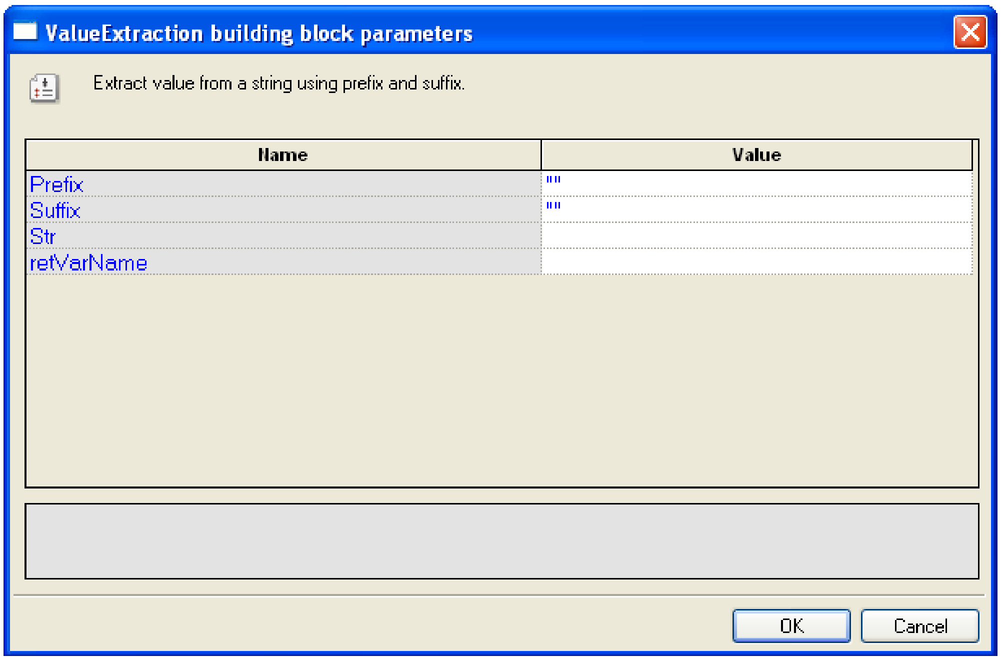
Value Extraction

WebLOAD Recorder enables you to add value extraction to a script to define the parameters for the extractValue JavaScript function.

To add value extraction to a script:

1. Drag the **Value Extraction** icon, from the Load toolbox, into the Script Tree at the desired location.

   The Value Extraction Building Block parameters dialog box opens.



Extract value from a string using prefix and suffix.

Name	Value
Prefix	""
Suffix	""
Str	
retVarName	

Figure 148: Value Extraction Building Block Parameters Dialog Box

2. In the Prefix field, enter a prefix.
3. In the Suffix field, enter a suffix.
4. In the Str field, enter the string that will be searched.
5. In the retVarName, enter the variable name that will be generated to the script.
6. Click **OK**.

The Value Extraction Building Block is added to the Script Tree. The JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Note: Fields that were not assigned a value in the dialog box are left as empty fields in the script code.

Define Concurrent

WebLOAD Recorder enables you to collect Post and Get HTTP requests and simultaneously execute them by two or more threads, as defined in the MultiThread Virtual Clients number. This is configured in the Browser Parameters tab in WebLOAD Console's Script Options dialog box.

Note: WebLOAD Recorder does not perform the Post and Get HTTP requests concurrently.

To simultaneously execute Post and Get HTTP requests, you must define where in the script to begin collecting the requests and where to stop collecting and begin executing them. The HTTP requests are collected until the engine encounters an `Execute Concurrent` function in the script. For more information about the Execute Concurrent Building Block, see *Execute Concurrent* (on page 246).

These Post and Get HTTP requests are saved in a file which you can access at any time. For more information, refer to the *WebLOAD JavaScript Reference*.

To start collecting HTTP requests in a script:

- Drag the **Define Concurrent** icon from the Load toolbox into the Script Tree at the desired location.

  The Define Concurrent Building Block is added to the Script Tree. The JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Execute Concurrent

WebLOAD Recorder enables you to simultaneously execute all the Post and Get HTTP requests that were defined since the last `Define Concurrent` function by two or more threads, as defined in the MultiThread Virtual Clients number. This is configured in the Browser Parameters tab in WebLOAD Console's Script Options dialog box.

Note: This function can only be inserted in your script *after* a Define Concurrent function. For more information about the Define Concurrent function, see *Define Concurrent* (on page 246).

When the engine encounters the `Execute Concurrent` function, it stops collecting the HTTP requests in the script and starts their execution.

To start concurrently executing HTTP requests in a script:

- Drag the **Execute Concurrent** icon from the Load toolbox into the Script Tree at the desired location.

The Execute Concurrent Building Block is added to the Script Tree. The JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The WebLOAD Recorder Internet Protocols Toolbox

Use the WebLOAD Recorder Internet Protocols Building Blocks to simply and easily add Internet Protocols functionality to your test session script without having to write numerous lines of code.

To add Internet Protocols Building Blocks to a test script directly through the WebLOAD Recorder GUI:

- Drag the selected Internet Protocols icon from the Internet Protocols toolbox and drop it into the Script Tree at the appropriate point.

  WebLOAD Recorder automatically adds the appropriate JavaScript code to your test session script.

WebLOAD Recorder provides full support for secure sites that utilize the SSL security protocol. The same FTP, POP, and SMTP functionality that is available for standard-security sites is also provided for sites that utilize the SSL security protocol. WebLOAD Recorder SSL protocol support is virtually transparent for the web site tester. Simply choose the appropriate Building Block, such as FTP-Connect, for example. Activate the SSL Protocol feature by setting the Boolean SSLFlag property to true. Complete the rest of the Building Block properties as described for standard Building Block use.

Note: The Internet Protocols Building Blocks displayed in the Internet Protocols toolbox correspond to only a small part of the WebLOAD Recorder Internet Protocols function set. These Building Blocks are provided for the most commonly used Internet Protocols activities. For a description of the complete set of Internet Protocols functions supported by WebLOAD Recorder, see the *WebLOAD Internet Protocols Reference* in the *WebLOAD JavaScript Reference Guide*.

The following Internet Protocols dialog boxes are described here:

Table 40: *Internet Protocols Dialog Boxes*

Protocol	Building Blocks	
FTP	FTP-connect *FTP-Connect* (on page 250)	FTP-upload *FTP-Upload* (on page 252)
	FTP-download *FTP-Download* (on page 254)	FTP-disconnect *FTP-Disconnect* (on page 256)
SMTP	SMTP-send message *SMTP-Send Message* (on page 256)	
POP	POP-retrieve *POP-Retrieve* (on page 258)	POP-Delete *POP-Delete* (on page 260)
IMAP	IMAP-Connect *IMAP-Connect* (on page 263)	IMAP-Retrieve *IMAP-Retrieve* (on page 265)
	IMAP-Delete *IMAP-Delete* (on page 266)	IMAP-CreateMailbox *IMAP-CreateMailbox* (on page 268)
	IMAP-ListMailboxes *IMAP-ListMailboxes* (on page 269)	IMAP-DeleteMailbox *IMAP-DeleteMailbox* (on page 269)
	IMAP-RenameMailbox *IMAP-RenameMailbox* (on page 271)	IMAP-SubscribeMailbox *IMAP-SubscribeMailbox* (on page 272)
	IMAP-UnsubscribeMailbox *IMAP-UnsubscribeMailbox* (on page 273)	IMAP-List Subscribed Mailboxes *IMAP-ListSubscribedMailboxes* (on page 275)

Protocol	Building Blocks	
	IMAP-Search *IMAP-Search* (on page 275)	
NNTP	NNTP-Connect *NNTP-Connect* (on page 278)	NNTP-GetArticle *NNTP-GetArticle* (on page 280)
	NNTP-GetArticleCount *NNTP-GetArticleCount* (on page 282)	NNTP-PostArticle *NNTP-PostArticle* (on page 283))
TCP	TCP-Connect *TCP-Connect* (on page 285))	TCP-Send *TCP-Send* (on page 287)
	TCP-Receive *TCP-Receive* (on page 288)	TCP-Erase *TCP-Erase* (on page TCP-Erase)
TELNET	TELNET-Connect *TELNET-Connect* (on page 289)	TELNET-Receive *TELNET-Receive* (on page 291))
	TELNET-Send *TELNET-Send* (on page 293)	TELNET-Erase *TELNET-Erase* (on page 294)
UDP	UDP-Bind *UDP-Bind* (on page 295)	UDP - Broadcast *UDP-Broadcast* (on page 297)
	UDP-Receive *UDP-Receive* (on page 298)	UDP-Send *UDP-Send* (on page 299)
	UDP-Erase *UDP-Erase* (on page 300)	
LDAP	LDAP-Bind	LDAP-Search
	LDAP-UnBind	

Each Internet Protocols icon opens a different dialog box. Enter the required values in the Value field. Explanations are provided at the bottom of the dialog box for each parameter as it is selected in the dialog box.

Note: Values that must be enclosed within quotation marks are indicated in the Value column by sets of quotation marks. Type the field value within the quotation marks that automatically appear in the input-text box that pops-up when the value field is selected. Fields that were not assigned a value in the dialog box are left as empty fields in the script code.

Once you have finished defining the new Internet Protocols activity, the new action is reflected in the Script Tree. An Internet Protocols icon is added to the Script Tree for each Internet Protocols activity defined. WebLOAD Recorder automatically adds the corresponding JavaScript code to your test session script.

To see the complete sequence of JavaScript code for all the Internet Protocols Building Blocks that have been added to the Script Tree, click the Agenda root node in the Script Tree.

Note: The JavaScript code for each of the Internet Protocols Building Blocks can be found in the Internet Protocols library files, part of the `Include` directory under the WebLOAD installation directory. Each protocol has its own library file. For example, the SMTP functions refer to the `wlSMTP.js` file.

FTP

Dragging an FTP icon into your Script Tree opens an FTP Building Block parameters dialog box.

FTP toolbox items include:

- **FTP-Connect**: Open an FTP connection.
- **FTP-Upload**: Designate a file to be uploaded to a remote host.
- **FTP-Download**: Designate a file to be downloaded from a remote host.
- **FTP-Disconnect**: Disconnect from a remote host.

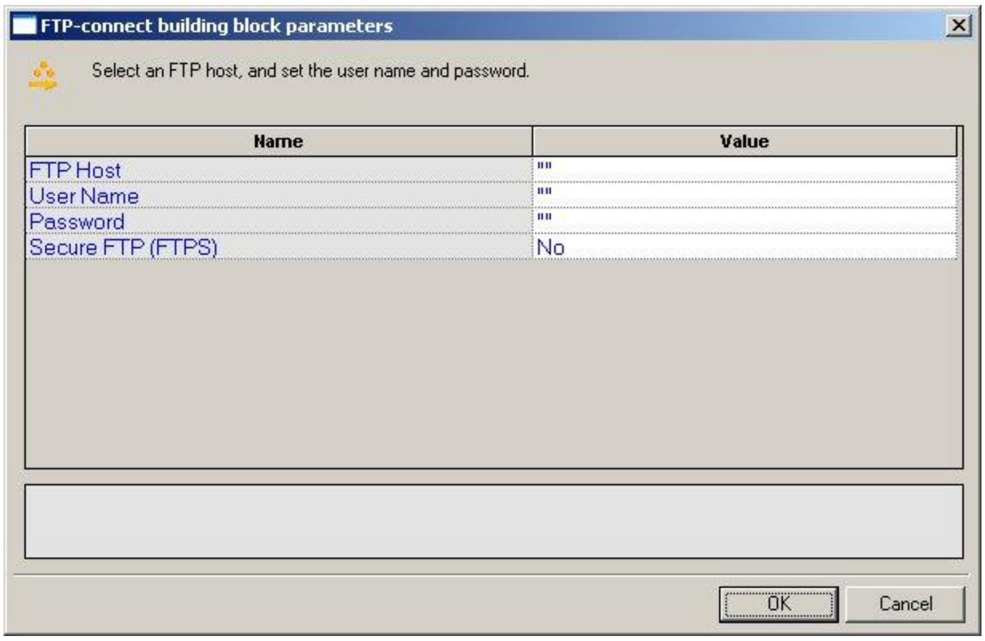
FTP-Connect

Use the FTP-Connect Building Block to open an FTP connection.

To enter a value:

1. Drag the **FTP-Connect** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The FTP-Connect Building Block parameters dialog box opens.


FTP-connect building block parameters
Select an FTP host, and set the user name and password.
OK
Cancel

Name	Value
FTP Host	""
User Name	""
Password	""
Secure FTP (FTPS)	No

Figure 149: FTP-Connect Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the UserName field is used to define the user ID to be used when logging in to the specified FTP host. WebLOAD Recorder automatically sends the user-specified name and password to the FTP host when connecting.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 41.
4. Click **OK**.

   The FTP-Connect Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()` and `InitClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the `InitAgenda()` function notes that the connection will be utilizing SSL security, and therefore includes the WebLOAD Recorder FTP/SSL library file. The `InitClient()` function includes a command to define a separate FTP/SSL object for each client. Within the main body of the script, an FTP connection is opened using the connection name, user name, and password specified by the user.

The fields in the FTP-Connect Building Block parameters dialog box are described in the following table:

Table 41: FTP Connect Building Block Parameters Dialog Box Fields

Field Name	Description
FTP Host	Specify the name of the FTP host connection. Type the FTP Host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The FTP host is identified either through a DNS number or a full name string. A host name string must be enclosed within quotation marks.
User Name	Specify a user ID for the FTP connection. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The user name must be enclosed within quotation marks.
Password	Specify a password for authentication during the FTP connection. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The password must be enclosed within quotation marks.
Secure FTP (FTPS)	Select the appropriate Boolean value to indicate whether the site being accessed utilizes the SSL security protocol.

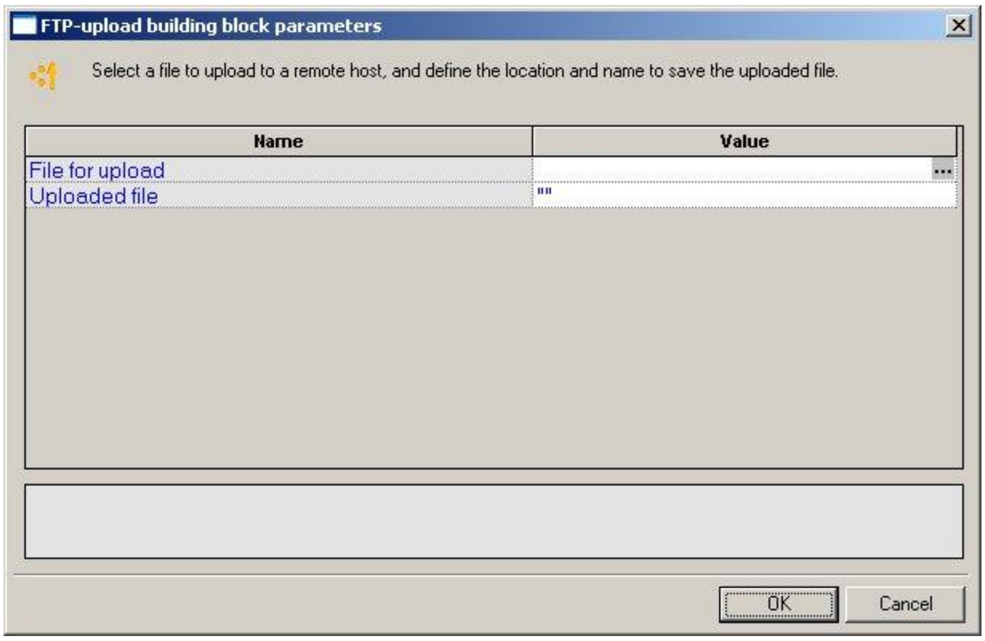
FTP-Upload

Use the FTP-Upload Building Block to designate a file to be uploaded to a remote host.

To enter a value:

1. Drag the **FTP-Upload** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The FTP-Upload Building Block parameters dialog box opens.


FTP-upload building block parameters

Select a file to upload to a remote host, and define the location and name to save the uploaded file.

Name	Value
File for upload	
Uploaded file	""

Figure 150: FTP-Upload Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Uploaded File field is used to define the name and location for the file to be saved on the specified FTP host.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 42.

Note: If the script will be running for multiple clients or over multiple rounds, use global variables to specify a unique file name for each client and/or round, to avoid file access conflicts and to make it easier to work with and analyze the files after the test is completed. For example:

```
"k:\Ftp\files\inputFiles\text_upload_"+ ThreadNum + RoundNum +
".txt"
```

4. Click **OK**.

The FTP-Upload **B**uilding Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Note: The WebLOAD Recorder global variables `ThreadNum` and `RoundNum are` used to differentiate between the files uploaded by different clients during different test iterations.

The fields in the FTP-Upload Building Block parameters dialog box are described in the following table:

Table 42: FTP-Upload Building Block Parameters Dialog Box Fields

Field Name	Description
File for upload	Specify the name of the file to be uploaded to the specified FTP host. Select the appropriate file from the Browser window that appears when you click the ... button to the right of the Value input area for this field.
Uploaded file	Specify a name and location to save the uploaded file. Type the uploaded file name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The file name must be enclosed within quotation marks.

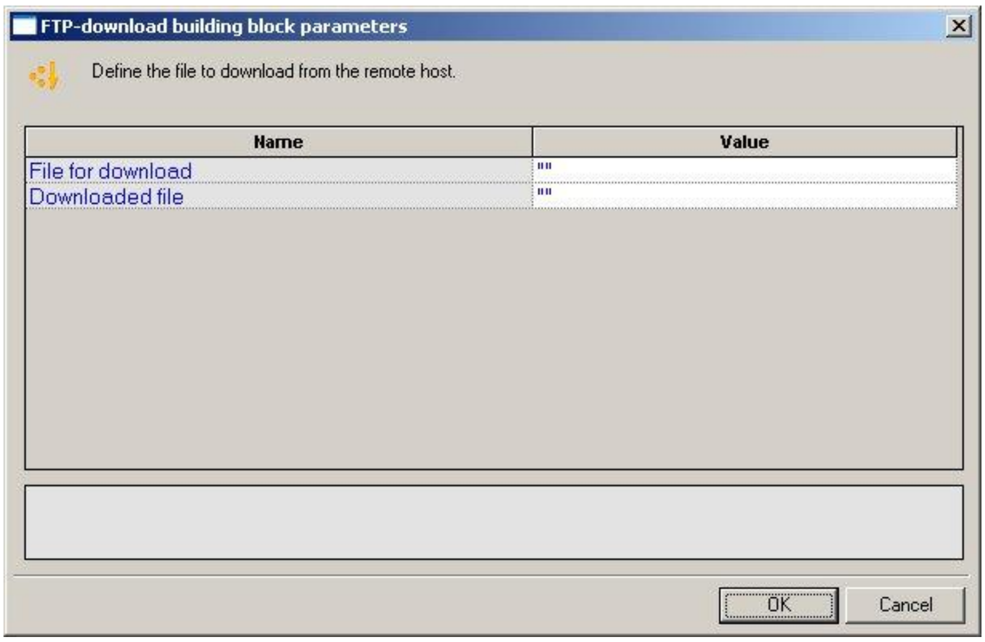
FTP-Download

Use the FTP-Download Building Block to designate a file to be downloaded from a remote host.

To enter a value:

1. Drag the **FTP-Download** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The FTP-Download Building Block parameters dialog box opens.


FTP-download building block parameters
Define the file to download from the remote host.
OK
Cancel

Name	Value
File for download	""
Downloaded file	""

Figure 151: FTP-Download Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the File for Download field is used to define the name of the file to be downloaded from the specified FTP host.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 43.

Note: If the script will be running for multiple clients or over multiple rounds, use global variables to specify a unique file name for each client and/or round, to avoid file access conflicts and to make it easier to work with and analyze the files after the test is completed. For example:

```
"k:\Ftp\files\inputFiles\text_upload_" + ThreadNum + RoundNum +
".txt"
```

4. Click **OK**.

   The FTP-Download Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the name of the file to be downloaded is passed as a parameter to the `ftp.Download()` function. The file name to which the downloaded file should be saved is assigned as a value to the `ftp.Outfile` variable.

The fields in the FTP-Download Building Block parameters dialog box are described in the following table:

Table 43: FTP-Download Building Block Parameters Dialog Box Fields

Field Name	Description
File for download	Specify the name of the file to be downloaded from the specified FTP host. Type the name of the file to be downloaded into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The file name must be enclosed within quotation marks.
Downloaded file	Specify a name and location to save the downloaded file. Type the name and location in which to save the downloaded file into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The file name must be enclosed within quotation marks.

FTP-Disconnect

Use the **FTP-Disconnect** Building Block to disconnect from a remote host.

To enter a value:

- Drag the **FTP-Disconnect** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

  The FTP-Disconnect Building Block is added to the Script Tree. The JavaScript code, including the `TerminateClient()` function, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

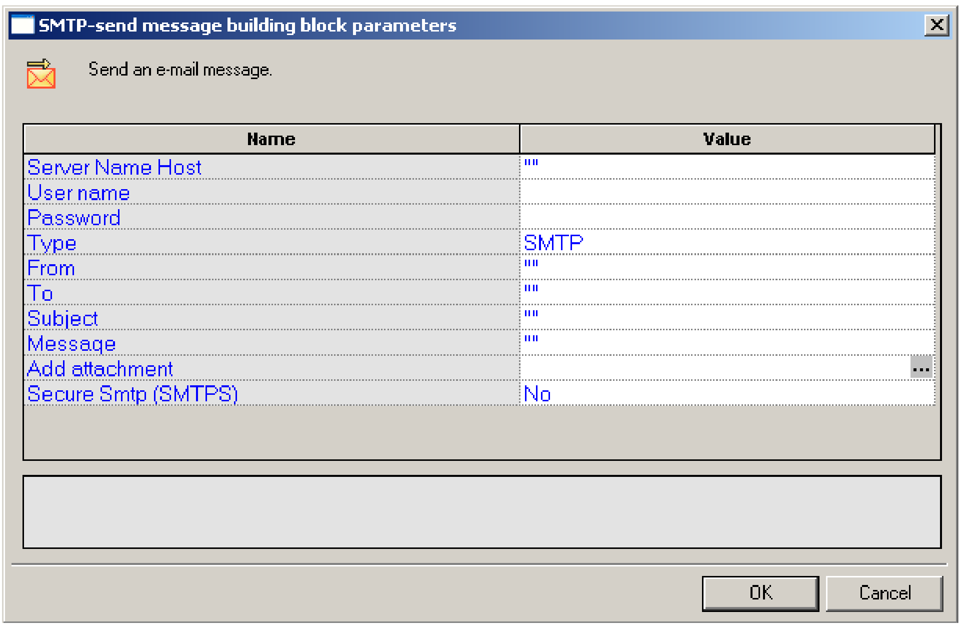
SMTP-Send Message

Use the SMTP-Send Message Building Block to define an email to be sent.

To enter a value:

1. Drag the **SMTP-Send Message** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The SMTP-Send Message Building Block parameters dialog box opens.



Name	Value
Server Name Host	""
User name	
Password	
Type	SMTP
From	""
To	""
Subject	""
Message	""
Add attachment	...
Secure Smtp (SMTPS)	No

Figure 152: SMTP-Send Message Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Server Name Host designates the name of the host to which the email should be sent.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 44.
4. Click **OK**.

   The SMTP-Send Message Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the specified SMTP connection is opened, an email message constructed from the user input is sent out, and the SMTP connection is closed.

The fields in the SMTP-Send Message Building Block parameters dialog box are described in the following table:

Table 44: SMTP Send Message Building Block Parameters Dialog Box Fields

Field Name	Description
Server Name Host	Specify the name of the host to which the email should be sent. Type the host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The file name must be enclosed within quotation marks. **Note:** The host can be designated either with a full text name or DNS number.
User name	Specify a user name with which to login to the mail server. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The user name must be enclosed within quotation marks.
Password	Specify a password with which to login to the mail server. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The password must be enclosed within quotation marks.
Type	Select which type to use: • SMTP • ESMTP (SMTP extensions – supports graphics and other attachments.)
From	Specify the name of the person sending the email. Type the sender's name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The name must be enclosed within quotation marks.

Field Name	Description
To	Specify the name of the person to whom the email should be sent. Type the receiver's name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The name must be enclosed within quotation marks.
Subject	Enter a short text line that appears as the subject line for the email being sent. Type the subject line into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The subject text must be enclosed within quotation marks.
Message	Enter the message text of the email being sent. Type the message text into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The message text must be enclosed within quotation marks.
Add attachment	Specify the name of a file to be attached to this email. Select the appropriate file from the Browser window that appears when you click the ... button to the right of the Value input area for this field.
Secure SMTP (SMTPS)	Select the appropriate Boolean value to indicate whether the site being accessed utilizes the SSL security protocol.

POP

Dragging a POP icon into your Script Tree opens a POP Building Block parameters dialog box.

POP toolbox items include:

- **POP-Retrieve**: Retrieve all waiting messages.
- **POP-Delete**: Delete all messages from a POP mailbox.

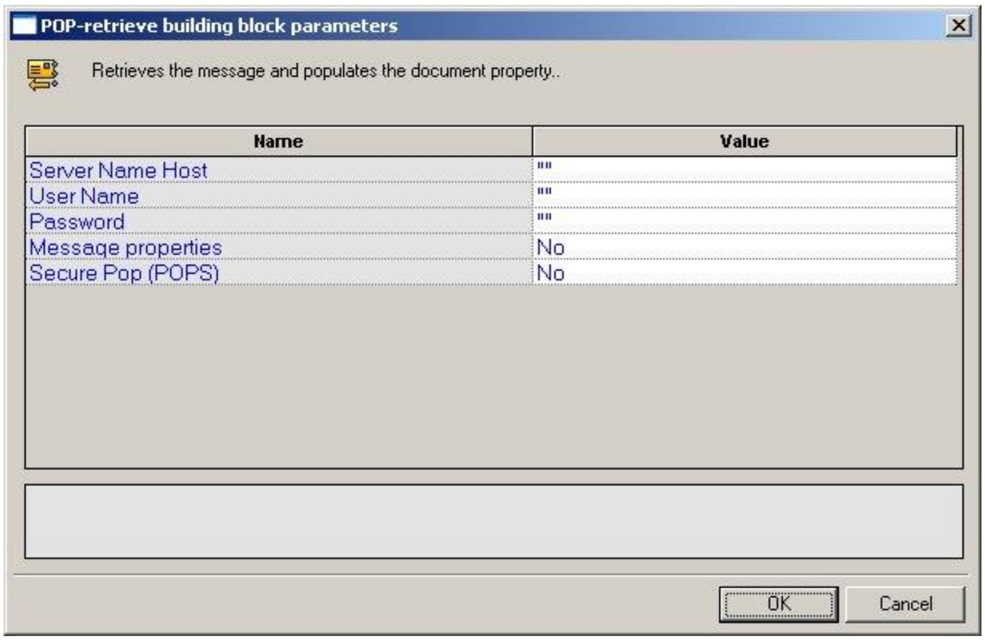
POP-Retrieve

Use the POP-Retrieve Building Block to retrieve all waiting messages, optionally together with a full set of header properties for each message.

To enter a value:

1. Drag the **POP-Retrieve** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The POP-Retrieve Building Block parameters dialog box opens.



POP-retrieve building block parameters

Retrieves the message and populates the document property..

Name	Value
Server Name Host	""
User Name	""
Password	""
Message properties	No
Secure Pop (POPS)	No

Figure 153: POP-Retrieve Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Message Properties field is a toggle that defines whether or not all the message properties should be retrieved.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 45.

4. Click **OK**.

   The POP-Retrieve Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the POP connection is opened using the connection name, user name, and password specified by the user. The waiting messages are retrieved and the message property values are saved to a local structure.

The fields in the POP-Retrieve Building Block parameters dialog box are described in the following table:

Table 45: POP-Retrieve Building Block Parameters Dialog Box Fields

Field Name	Description
Server Name Host	Specify the name of the POP host connection. Type the POP Host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The host name must be enclosed within quotation marks.
User Name	Specify a user ID for the POP connection. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The user name must be enclosed within quotation marks.
Password	Specify a password for authentication during the POP connection. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The password must be enclosed within quotation marks.
Message properties	A toggle that defines whether or not all the message properties should be retrieved. Toggle Message Properties on or off depending on whether you select Yes or No from the list displayed in the drop-down list box that appears when you click the small arrow to the right of the Value input area for this field.
Secure POP (POPS)	Select the appropriate Boolean value to indicate whether the site being accessed utilizes the SSL security protocol.

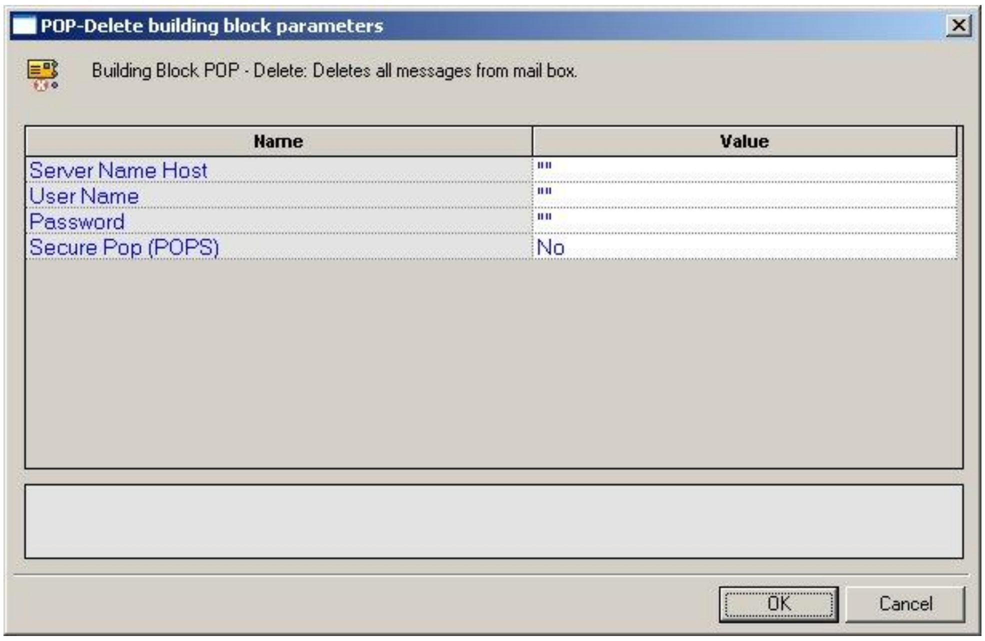
POP-Delete

Use the POP-Delete Building Block to delete all messages from a POP mailbox.

To enter a value:

1. Drag the **POP-Delete** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The POP-Delete Building Block parameters dialog box opens.



POP-Delete building block parameters

Building Block POP - Delete: Deletes all messages from mail box.

Name	Value
Server Name Host	""
User Name	""
Password	""
Secure Pop (POPS)	No

Figure 154: POP-Delete Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Server Name Host field is used to define the name of the mail server. WebLOAD Recorder automatically sends the user-specified name and password to the server when connecting.

3. Enter the appropriate field value into the Value column next to the field name, as described in Table 46.

4. Click **OK**.

   The POP-Delete Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, a POP connection is opened using the host name, user name, and password specified by the user. The code then loops through all messages on the server, deleting each message and printing a note to the user identifying the message that was just deleted. When all messages are deleted, the connection is closed.

The fields in the POP-Delete Building Block parameters dialog box are described in the following table:

Table 46: POP-Delete Building Block Parameters Dialog Box Fields

Field Name	Description
Server Name Host	Specify the name of the POP server connection. Type the POP host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The host name must be enclosed within quotation marks.
User Name	Specify a user ID for the POP connection. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The user name must be enclosed within quotation marks.
Password	Specify a password for authentication during the POP connection. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The password must be enclosed within quotation marks.
Secure POP (POPS)	Select the appropriate Boolean value to indicate whether the site being accessed utilizes the SSL security protocol.

IMAP

Dragging an IMAP icon into your Script Tree opens an IMAP Building Block parameters dialog box.

IMAP toolbox items include:

- **IMAP-Connect**: Start an IMAP session.
- **IMAP-Retrieve**: Retrieve all waiting messages.
- **IMAP-Delete**: Delete messages from an IMAP mailbox.
- **IMAP-CreateMailbox**: Create a new IMAP mailbox.
- **IMAP-ListMailboxes**: Generate a complete list of all IMAP mailboxes accessed through the current IMAP server.
- **IMAP-DeleteMailbox**: Delete an IMAP mailbox.
- **IMAP-RenameMailbox**: Rename an IMAP mailbox.
- **IMAP-SubscribeMailbox**: Subscribe to an IMAP mailbox.
- **IMAP-UnsubscribeMailbox**: Unsubscribe from an IMAP mailbox.
- **IMAP-ListSubscribeMailboxes**: Generate a complete list of all subscribed IMAP mailboxes accessed through the current IMAP server

- **IMAP-Search**: Search for a specific email item within an IMAP mailbox.

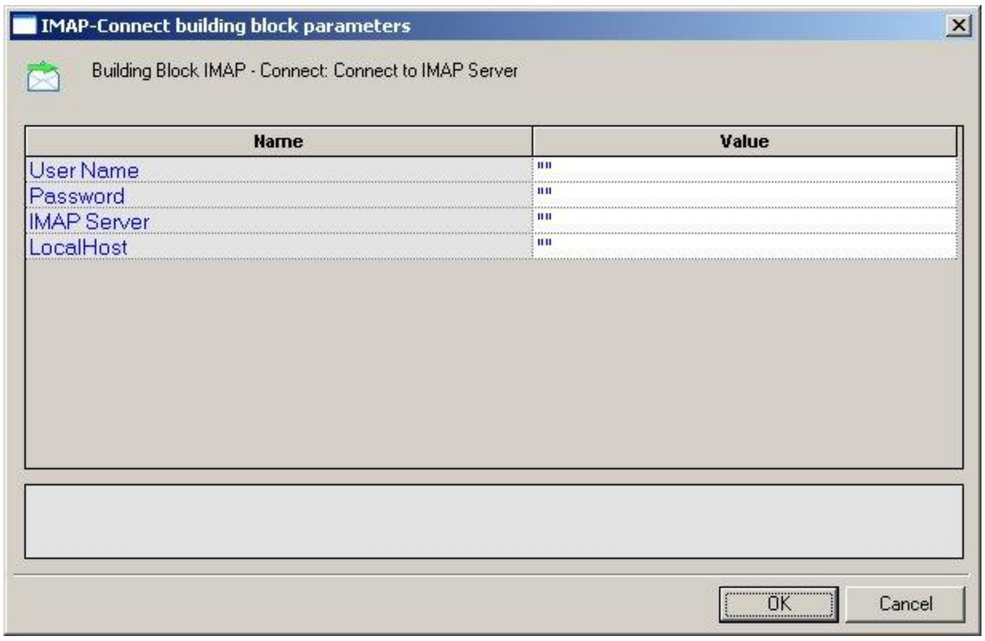
IMAP-Connect

Use the IMAP-Connect Building Block to start an IMAP session. When you connect, you are connecting to a specific mailbox within the host, as specified by your User ID.

To enter a value:

1. Drag the **IMAP-Connect** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The IMAP-Connect Building Block parameters dialog box opens.



IMAP-Connect building block parameters

Building Block IMAP - Connect: Connect to IMAP Server

Name	Value
User Name	""
Password	""
IMAP Server	""
LocalHost	""

Figure 155: IMAP-Connect Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the IMAP Server field is used to define the IMAP Server Name or IP to be used when logging in to the specified IMAP server. WebLOAD Recorder automatically sends the user-specified name and password to the IMAP server when connecting.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 47.

4. Click **OK**.

   The IMAP-Connect Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, an IMAP connection is opened using the connection name, local host name, user name, and password specified by the user.

The fields in the IMAP-Connect Building Block parameters dialog box are described in the following table:

Table 47: IMAP-Connect Building Block Parameters Dialog Box Fields

Field Name	Description
User Name	Specify an NT user ID for the IMAP connection. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The user name must be enclosed within quotation marks.
Password	Specify an NT password for authentication during the IMAP connection. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The password must be enclosed within quotation marks.
IMAP Server	Specify the IMAP server name or IP number. Type the IMAP server name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The IMAP host is identified either through an IP number or a full name string. A server name string must be enclosed within quotation marks.
LocalHost	Specify the name of the local host. Type the local host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The local host is identified either through a DNS number or a full name string. A host name string must be enclosed within quotation marks.

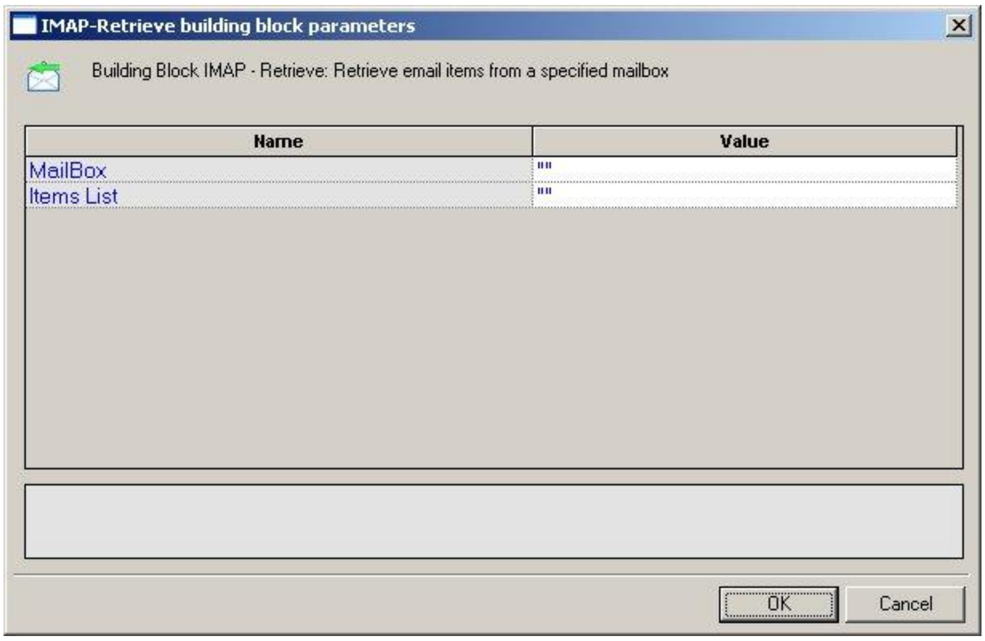
IMAP-Retrieve

Use the IMAP-Retrieve Building Block to retrieve all waiting messages, optionally together with a full set of header properties for each message.

To enter a value:

1. Drag the **IMAP-Retrieve** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The IMAP-Retrieve Building Block parameters dialog box opens.



IMAP-Retrieve building block parameters

Building Block IMAP - Retrieve: Retrieve email items from a specified mailbox

Name	Value
MailBox	""
Items List	""

Figure 156: IMAP-Retrieve Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Items List field contains a list of mailbox items to be retrieved.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 48.

4. Click **OK**.

   The IMAP-Retrieve Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the specified message is retrieved from the specified mailbox and the message property values are saved to a local structure. A comment embedded in the code describes the message attributes stored in the `imap` JavaScript object.

The fields in the IMAP-Retrieve Building Block parameters dialog box are described in the following table:

Table 48: IMAP-Retrieve Building Block Parameters Dialog Box Fields

Field Name	Description
MailBox	Specify the name of the mailbox from which messages should be retrieved. Type the mailbox name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The mailbox name must be enclosed within quotation marks.
Items List	Specify the messages to be retrieved. Type the message numbers into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The message numbers must be enclosed within quotation marks. You may specify a single message number, or you may specify a range, separated by a colon. For example, 1:10 returns messages one through ten. If you do not specify a message ID, the next message is returned.

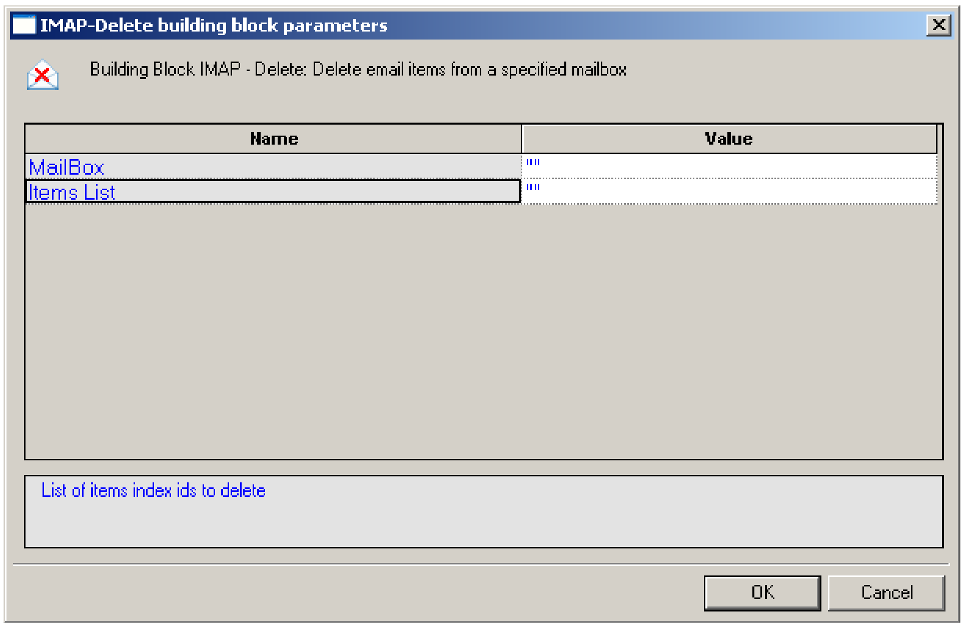
IMAP-Delete

Use the IMAP-Delete Building Block to delete messages from an IMAP mailbox.

To enter a value:

1. Drag the **IMAP-Delete** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The IMAP-Delete Building Block parameters dialog box opens.


IMAP-Delete building block parameters

Building Block IMAP - Delete: Delete email items from a specified mailbox

Name	Value
MailBox	""
Items List	""

List of items index ids to delete

Figure 157: IMAP-Delete Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Items List field contains a list of mailbox items to be deleted.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 49.
4. Click **OK**.

   The IMAP-Delete Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the messages specified by the user are deleted from the mail box specified by the user.

The fields in the IMAP-Delete Building Block parameters dialog box are described in the following table:

Table 49: IMAP-Delete Building Block Parameters Dialog Box Fields

Field Name	Description
MailBox	Specify the name of the mailbox from which messages should be deleted. Type the mailbox name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The mailbox name must be enclosed within quotation marks.

Field Name	Description
Items List	Specify the messages to be deleted. Type the message numbers into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The message numbers must be enclosed within quotation marks. You may specify a single message number, or you may specify a range, separated by a colon. For example, 1:10 deletes messages one through ten. If you do not specify a message ID, the current message is deleted.

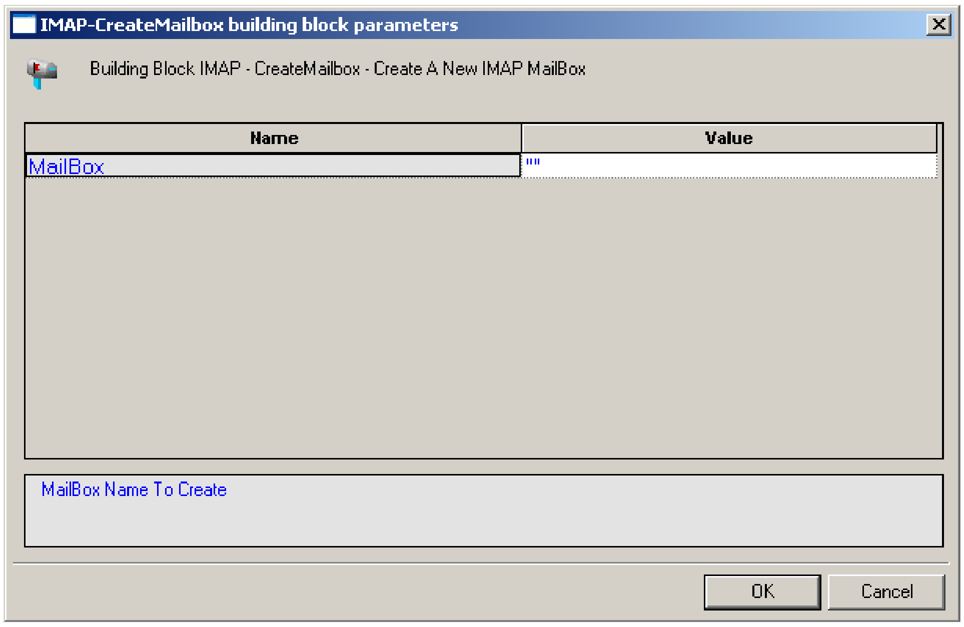
IMAP-CreateMailbox

Use the IMAP-CreateMailbox Building Block to create a new IMAP mailbox.

To enter a value:

1. Drag the **IMAP-CreateMailbox** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The IMAP-CreateMailbox Building Block parameters dialog box opens.



IMAP-CreateMailbox building block parameters

Building Block IMAP - CreateMailbox - Create A New IMAP MailBox

Name	Value
MailBox	""

MailBox Name To Create

Figure 158: IMAP-CreateMailbox Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the MailBox field contains the name of the mail box to be created.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 50.

4. Click **OK**.

   The IMAP-CreateMailbox Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, a new mailbox is created using the name specified by the user.

The field in the IMAP-CreateMailbox Building Block parameters dialog box is described in the following table:

Table 50: IMAP-CreateMailbox Building Block Parameters Dialog Box Field

Field Name	Description
MailBox	Specify the name of the mailbox to be created. Type the mailbox name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The mailbox name must be enclosed within quotation marks.

IMAP-ListMailboxes

Use the IMAP-ListMailboxes Building Block to generate a complete list of all IMAP mailboxes accessed through the current IMAP server.

To generate the list of IMAP mailboxes:

- Drag the **IMAP-ListMailboxes** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

  The IMAP-ListMailboxes Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

IMAP-DeleteMailbox

Use the IMAP-DeleteMailbox Building Block to delete an IMAP mailbox.

To delete an IMAP mailbox:

1. Drag the **IMAP-DeleteMailbox** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The IMAP-DeleteMailbox Building Block parameters dialog box opens.

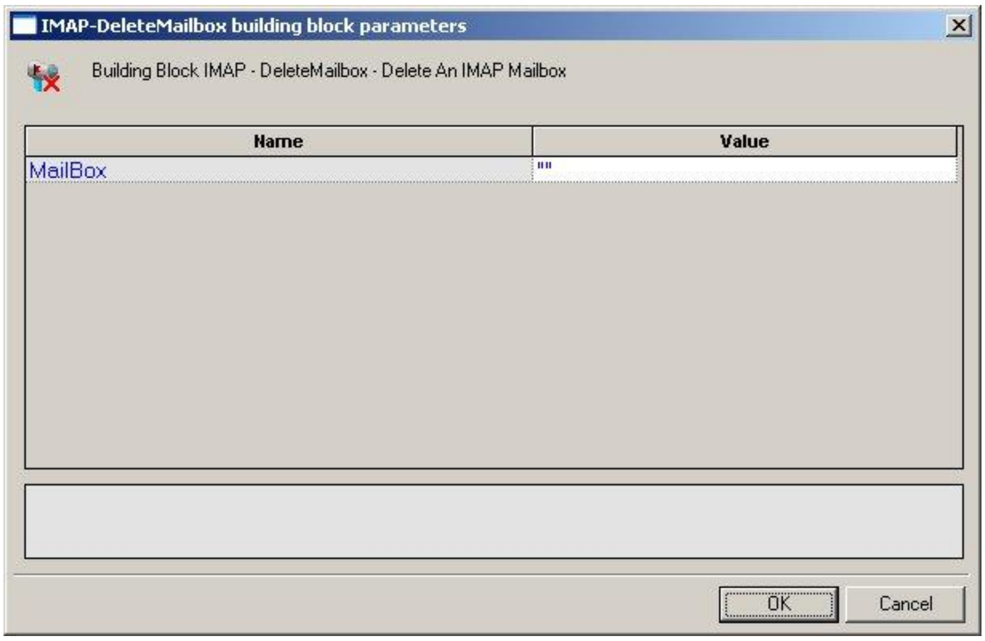

IMAP-DeleteMailbox building block parameters
Building Block IMAP - DeleteMailbox - Delete An IMAP Mailbox
Name
Value
MailBox
""
OK
Cancel

Figure 159: IMAP-DeleteMailbox Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the MailBox field contains the name of the mail box to be deleted.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 51.

4. Click **OK**.

   The IMAP-DeleteMailbox Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the mailbox specified by the user is deleted.

The field in the IMAP-DeleteMailbox Building Block parameters dialog box is described in the following table:

Table 51: IMAP-DeleteMailbox Building Block Parameters Dialog Box Field

Field Name	**Description**
MailBox	Specify the name of the mailbox to be deleted. Type the mailbox name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The mailbox name must be enclosed within quotation marks.

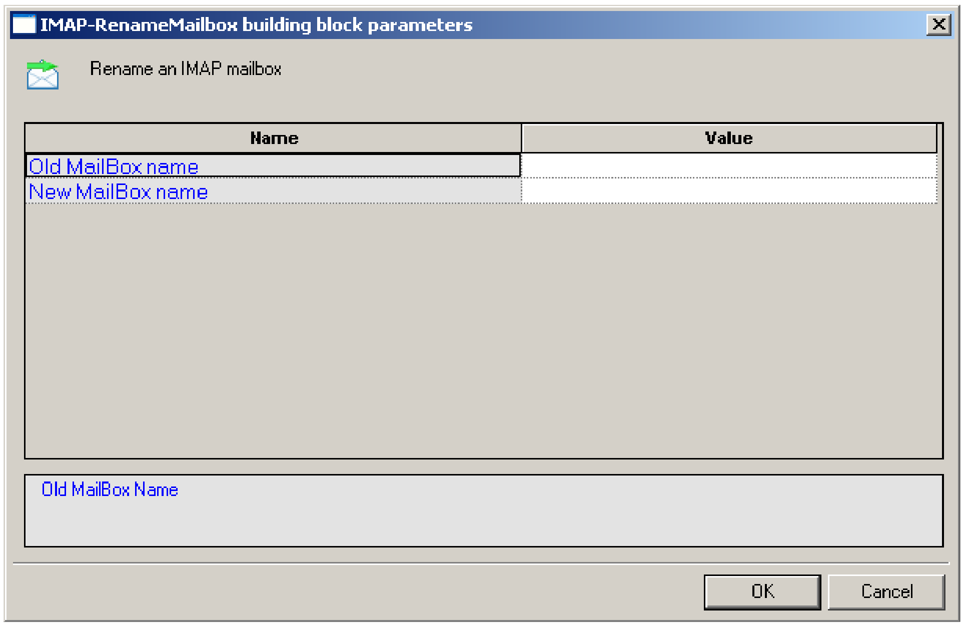
IMAP-RenameMailbox

Use the IMAP-RenameMailbox Building Block to rename an IMAP mailbox.

To rename an IMAP mailbox:

1. Drag the **IMAP-RenameMailbox** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The IMAP-RenameMailbox Building Block parameters dialog box opens.



Name	Value
Old MailBox name	
New MailBox name	

Old MailBox Name

Figure 160: IMAP-RenameMailbox Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area contains the name of the old mail box.

3. Enter the appropriate field value into the Value column next to the field name, as described in Table 52.
4. Click **OK**.

   The IMAP-RenameMailbox Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the mailbox is renamed using the name specified by the user.

The fields in the IMAP-RenameMailbox Building Block parameters dialog box are described in the following table:

Table 52: IMAP-RenameMailbox Building Block Parameters Dialog Box Field

Field Name	Description
Old MailBox name	Specify the name of the mailbox to be renamed. Type the old mailbox name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The mailbox name must be enclosed within quotation marks.
New Mailbox name	Specify the new name of the mailbox. Type the new mailbox name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The mailbox name must be enclosed within quotation marks.

IMAP-SubscribeMailbox

Use the IMAP-SubscribeMailbox Building Block to subscribe to an IMAP mailbox.

To subscribe to an IMAP mailbox:

1. Drag the **IMAP-SubscribeMailbox** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The IMAP-SubscribeMailbox Building Block parameters dialog box opens.

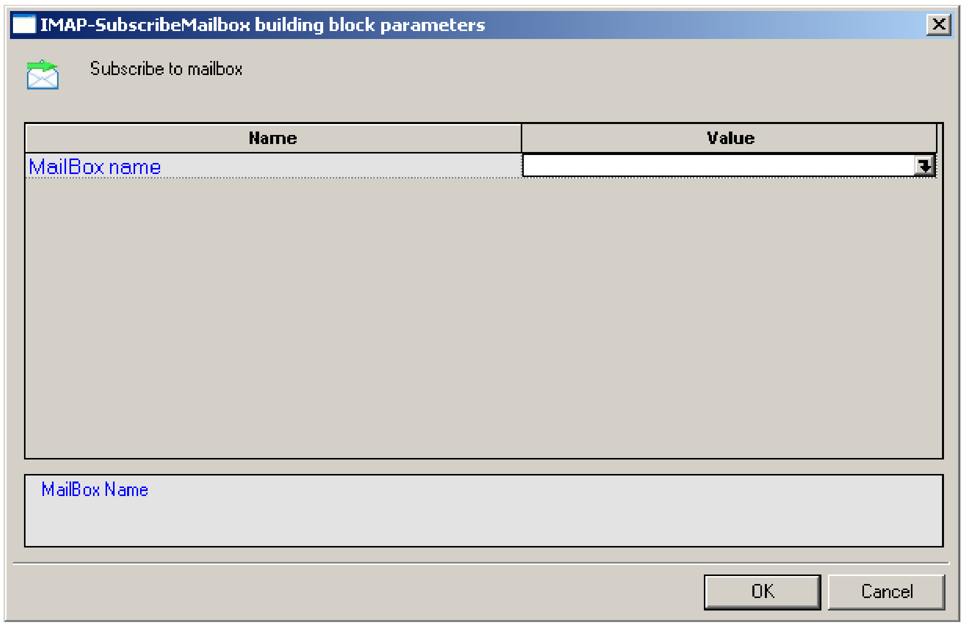


Figure 161: IMA-SubscribeMailbox Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area contains the name of the mail box.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 53.

4. Click **OK**.

   The IMAP-SubscribeMailbox Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the mailbox is renamed using the name specified by the user.

The field in the IMAP-SubscribeMailbox Building Block parameters dialog box is described in the following table:

Table 53: IMAP-SubscribeMailbox Building Block Parameters Dialog Box Field

Field Name	Description
MailBox name	Specify the name of the mailbox to which to subscribe. Type the mailbox name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The mailbox name must be enclosed within quotation marks.

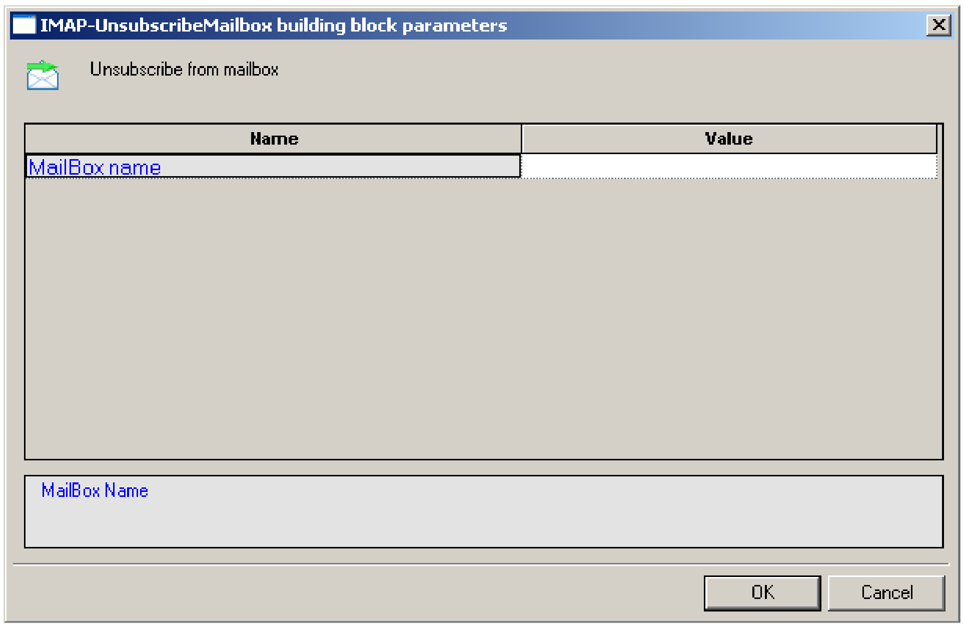
IMAP-UnsubscribeMailbox

Use the IMAP-UnsubscribeMailbox Building Block to unsubscribe from an IMAP mailbox.

To unsubscribe from an IMAP mailbox:

1. Drag the **IMAP-UnsubscribeMailbox** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The IMAP-UnsubscribeMailbox Building Block parameters dialog box opens.



MailBox Name

Figure 162: IMA-UnsubscribeMailbox Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area contains the name of the mail box.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 53.

4. Click **OK**.

   The IMAP-UnsubscribeMailbox Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the mailbox is renamed using the name specified by the user.

The field in the IMAP-UnsubscribeMailbox Building Block parameters dialog box is described in the following table:

Table 54: IMAP-UnsubscribeMailbox Building Block Parameters Dialog Box Field

Field Name	Description
MailBox name	Specify the name of the mailbox from which to unsubscribe. Type the mailbox name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The mailbox name must be enclosed within quotation marks.

IMAP-ListSubscribedMailboxes

Use the IMAP-ListSubscribedMailboxes Building Block to generate a complete list of all subscribed IMAP mailboxes.

To generate the list of subscribed IMAP mailboxes:

- Drag the **IMAP-ListSubscribedMailboxes** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

  The IMAP-ListSubscribedMailboxes Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

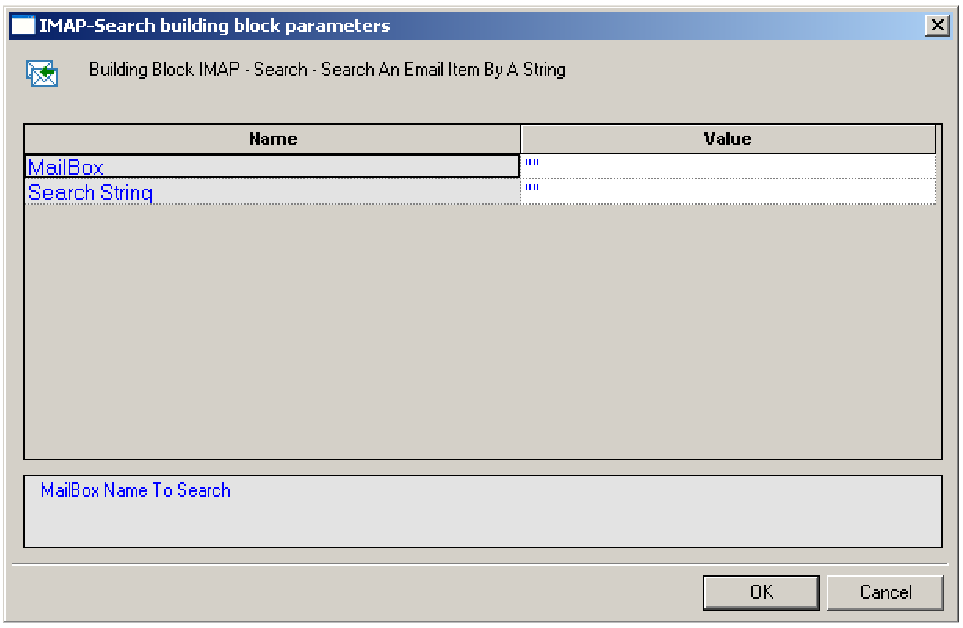
IMAP-Search

Use the IMAP-Search Building Block to search for a specific email item within an IMAP mailbox.

To search for a specific email item in an IMAP mailbox:

1. Drag the **IMAP-Search** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The IMAP-Search Building Block parameters dialog box opens.



IMAP-Search building block parameters

Building Block IMAP - Search - Search An Email Item By A String

Name	Value
MailBox	""
Search String	""

MailBox Name To Search

Figure 163: IMAP-Search Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

For example, in the preceding figure, the comment area explains that the MailBox field contains the name of the mail box to be searched.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 55.
4. Click **OK**.

   The IMAP-Search Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the mailbox specified by the user is searched for all mail items containing the string “timesheet”.

The fields in the IMAP-Search Building Block parameters dialog box are described in the following table:

Table 55: IMAP-Search Building Block Parameters Dialog Box Fields

Field Name	Description
MailBox	Specify the name of the mailbox to be searched. Type the mailbox name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The mailbox name must be enclosed within quotation marks.
Search String	Specify the search criteria for the current mailbox search. Valid search criteria include:
	ALL - All messages in the mailbox - this is the default initial key for AND-ing.
	ANSWERED - Messages with the \\Answered flag set.
	BCC - Messages that contain the specified string in the envelope structure’s BCC field.
	BEFORE - Messages whose internal date is earlier than the specified date.
	BODY - Messages that contain the specified string in the body of the message.
	CC - Messages that contain the specified string in the envelope structure’s CC field.
	DELETED - Messages with the \\Deleted flag set.
	DRAFT - Messages with the \\Draft flag set.
	FLAGGED - Messages with the \\Flagged flag set.
	FROM - Messages that contain the specified string in the envelope structure’s FROM field.

Field Name	Description
	HEADER - Messages that have a header with the specified field-name and that contains the specified string in the field-body.
	KEYWORD - Messages with the specified keyword set.
	LARGER - Messages with a size larger than the specified number of octets.
	NEW Messages that have the \\Recent flag set but not the \\Seen flag. This is functionally equivalent to "(RECENT UNSEEN)".
	NOT - Messages that do not match the specified search key.
	OLD - Messages that do not have the \\Recent flag set. This is functionally equivalent to "NOT RECENT" (as opposed to "NOT NEW").
	ON - Messages whose internal date is within the specified date.
	OR - Messages that match either search key.
	RECENT - Messages that have the \\Recent flag set.
	SEEN - Messages that have the \\Seen flag set.
	SENTBEFORE - Messages whose Date: header is earlier than the specified date.
	SENTON - Messages whose Date: header is within the specified date.
	SENTSINCE - Messages whose Date: header is within or later than the specified date.
	SINCE - Messages whose internal date is within or later than the specified date.
	SMALLER - Messages with an RFC822.SIZE smaller than the specified number of octets.
	SUBJECT - Messages that contain the specified string in the envelope structure's SUBJECT field.
	TEXT - Messages that contain the specified string in the header or body of the message.
	TO - Messages that contain the specified string in the envelope structure's TO field.
	UID - Messages with unique identifiers corresponding to the specified unique identifier set.
	UNANSWERED - Messages that do not have the \\Answered flag set.
	UNDELETED - Messages that do not have the \\Deleted flag set.
	UNDRAFT - Messages that do not have the \\Draft flag set.
	UNFLAGGED - Messages that do not have the \\Flagged flag set.
	UNKEYWORD - Messages that do not have the specified keyword set.
	UNSEEN - Messages that do not have the \\Seen flag set.

Field Name	Description
	This Building Block returns a string containing the IDs of messages that meet the search criteria if successful, an exception if unsuccessful.

NNTP

Dragging an NNTP icon into your Script Tree opens an NNTP Building Block parameters dialog box.

NNTP toolbox items include:

- **NNTP-Connect**: Start an NNTP session.
- **NNTP-GetArticle**: Retrieve articles from the specified news group from the NNTP server.
- **NNTP-GetArticleCount**: Retrieve the number of articles in the specified news group from the NNTP server.
- **NNTP-PostArticle**: Post articles to the specified news group.

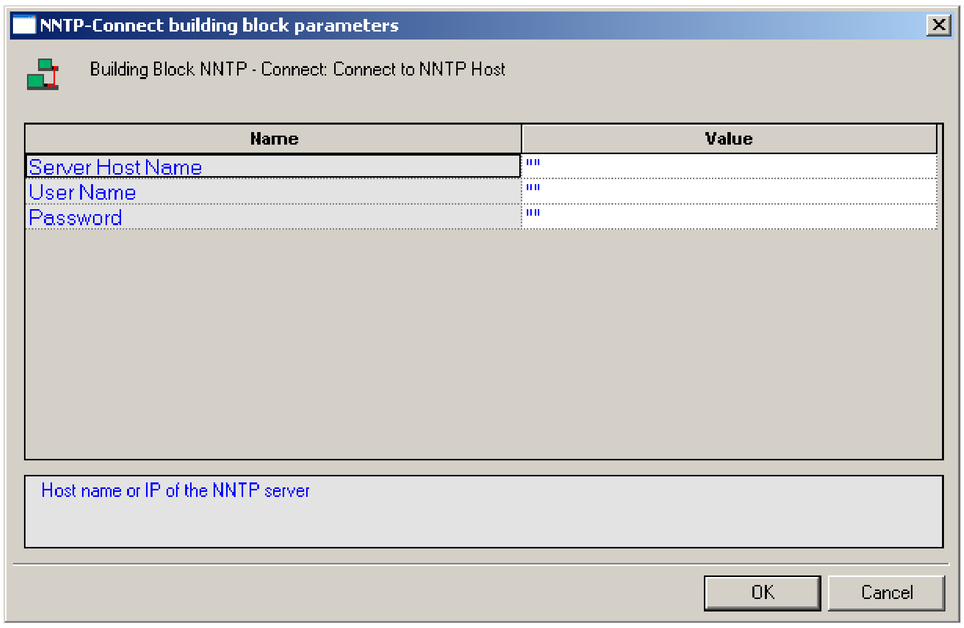
NNTP-Connect

Use the NNTP-Connect Building Block to start an NNTP session. When you connect, you are connecting to a specific.

To enter a value:

1. Drag the **NNTP-Connect** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The NNTP-Connect Building Block parameters dialog box opens.



NNTP-Connect building block parameters

Building Block NNTP - Connect: Connect to NNTP Host

Name	Value
Server Host Name	""
User Name	""
Password	""

Host name or IP of the NNTP server

Figure 164: NNTP-Connect Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Server Host Name field is used to define the NNTP Server Name or IP to be used when logging in to the specified NNTP server. WebLOAD Recorder automatically sends the user-specified name and password to the NNTP server when connecting.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 56.

4. Click **OK**.

   The NNTP-Connect Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, an NNTP connection is opened using the server name, user name, and password specified by the user.

The fields in the NNTP-Connect Building Block parameters dialog box are described in the following table:

Table 56: NNTP-Connect Building Block Parameters Dialog Box Fields

Field Name	Description
Server Host Name	Specify the NNTP server name or IP number. Type the NNTP server name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The NNTP host is identified either through an IP number or a full name string. A server name string must be enclosed within quotation marks.
User Name	Specify an NT user ID for the NNTP connection. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The user name must be enclosed within quotation marks.
Password	Specify an NT password for authentication during the NNTP connection. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The password must be enclosed within quotation marks.

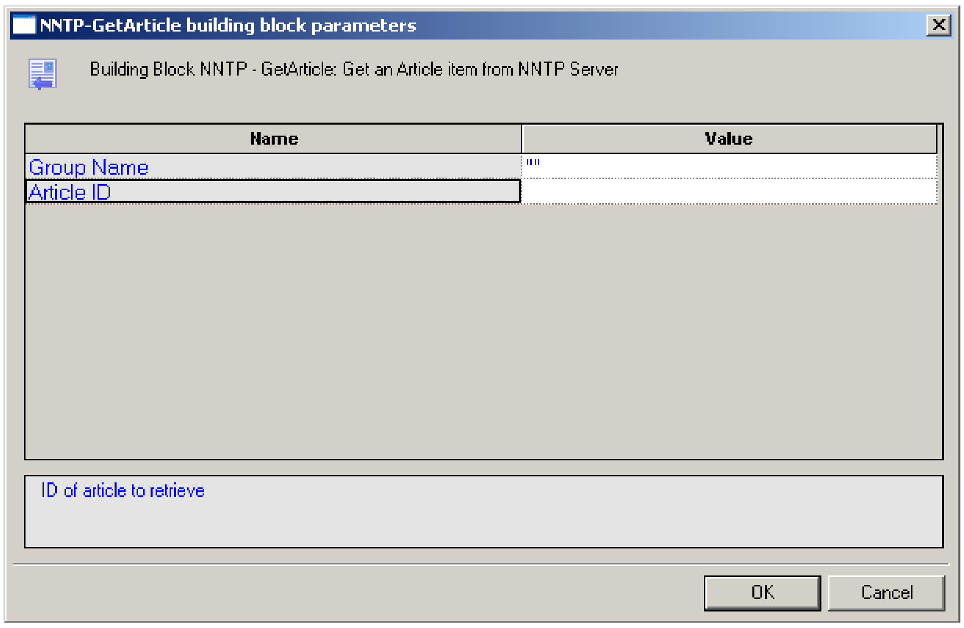
NNTP-GetArticle

Use the NNTP-GetArticle Building Block to retrieve articles from the specified news group from the NNTP server.

To enter a value:

1. Drag the **NNTP-GetArticle** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The NNTP-GetArticle Building Block parameters dialog box opens.



Building Block NNTP - GetArticle: Get an Article item from NNTP Server

Name	Value
Group Name	""
Article ID	

ID of article to retrieve

Figure 165:NNTP-GetArticle Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Article ID field contains the ID number of the news article to be retrieved.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 57.

4. Click **OK**.

   The NNTP-GetArticle Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the specified article is retrieved from the specified news group.

The fields in the NNTP-GetArticle Building Block parameters dialog box are described in the following table:

Table 57: NNTP-GetArticle Building Block Parameters Dialog Box Fields

Field Name	Description
Group Name	Specify the name of the news group from which articles should be retrieved. Type the news group name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The news group name must be enclosed within quotation marks.
Article ID	Specify the ID number of the article to be retrieved. Type the ID number into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.

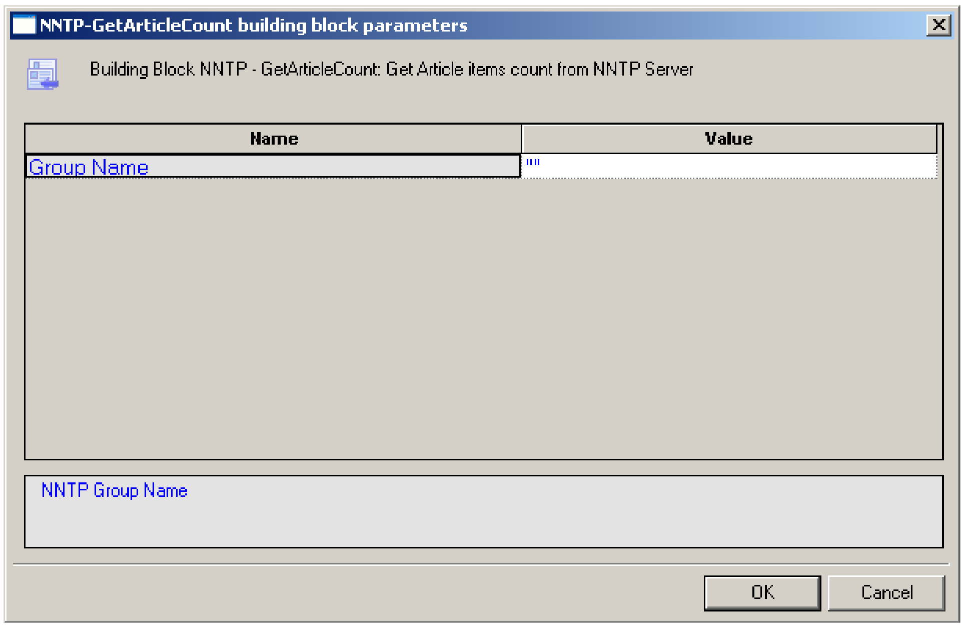
NNTP-GetArticleCount

Use the NNTP-GetArticleCount Building Block to retrieve the number of articles in the specified news group from the NNTP server.

To enter a value:

1. Drag the **NNTP-GetArticleCount** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The NNTP-GetArticleCount Building Block parameters dialog box opens.



NNTP-GetArticleCount building block parameters

Building Block NNTP - GetArticleCount: Get Article items count from NNTP Server

Name	Value
Group Name	""

NNTP Group Name

Figure 166: NNTP-GetArticleCount Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Group Name field contains the name of the news group whose articles are to be counted.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 58.

4. Click **OK**.

   The NNTP-GetArticleCount Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the number of articles appearing in the specified news group is returned.

The field in the NNTP-GetArticleCount Building Block parameters dialog box is described in the following table:

Table 58: NTTP-GetArticleCount Building Block Parameters Dialog Box Field

Field Name	Description
Group Name	Specify the name of the news group from which articles should be counted. Type the news group name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The news group name must be enclosed within quotation marks.

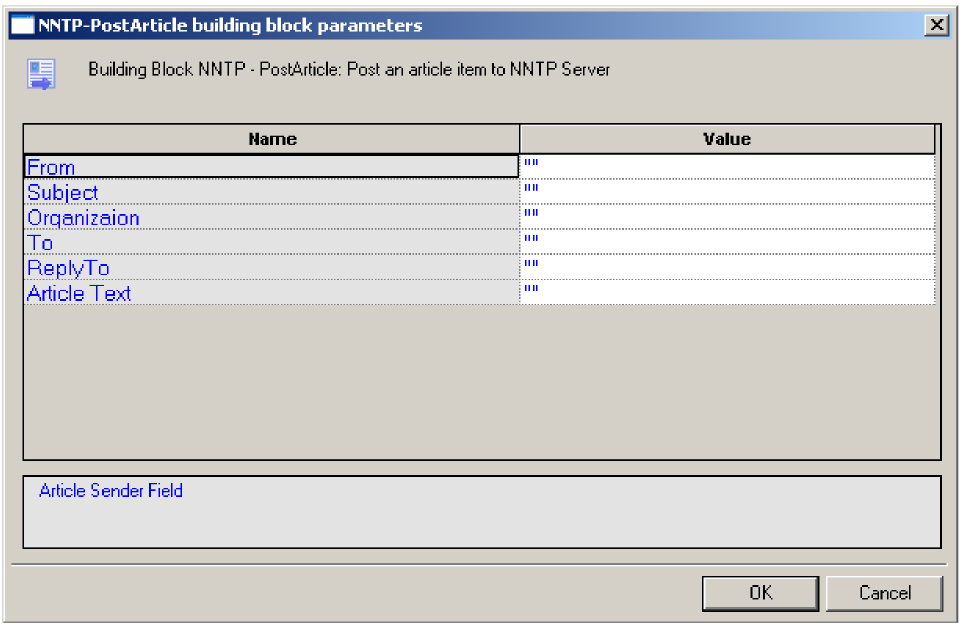
NNTP-PostArticle

Use the NNTP-PostArticle Building Block to post articles to the specified news group.

To enter a value:

1. Drag the **NNTP-PostArticle** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The NNTP-PostArticle Building Block parameters dialog box opens.



Building Block NNTP - PostArticle: Post an article item to NNTP Server

Name	Value
From	""
Subject	""
Organizaion	""
To	""
ReplyTo	""
Article Text	""

Article Sender Field

Figure 167: NNTP-PostArticle Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the From field contains the name of the person sending the news article to be posted on the news group.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 59.
4. Click **OK**.

   The NNTP-PostArticle Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the article text is posted to the specified news group.

The fields in the NNTP-PostArticle Building Block parameters dialog box are described in the following table:

Table 59: NNTP-PostArticle Building Block Parameters Dialog Box Fields

Field Name	Description
From	Specify the name of the person sending the email. Type the sender's name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The name must be enclosed within quotation marks.
Subject	Enter a short text line that appears as the subject line for the email being sent. Type the subject line into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The subject text must be enclosed within quotation marks.
Organization	Specify the name of the organization to which the recipient belongs.
To	Specify the name of the person to whom the email should be sent. Type the receiver's name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The name must be enclosed within quotation marks.
ReplyTo	Specify the name of the person to whom the recipient should reply. Type the name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The name must be enclosed within quotation marks.
Article Text	Enter the message text of the email being sent. Type the message text into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The message text must be enclosed within quotation marks.

TCP

Dragging a TCP icon into your Script Tree opens a TCP Building Block parameters dialog box.

TCP toolbox items include:

- **TCP-Connect**: Open a TCP connection.
- **TCP-Send**: Send a TCP request.
- **TCP-Receive**: Return all responses from the TCP host since the last TCP-Send action.
- **TCP-Erase**: Clear the contents of the TCP document object.

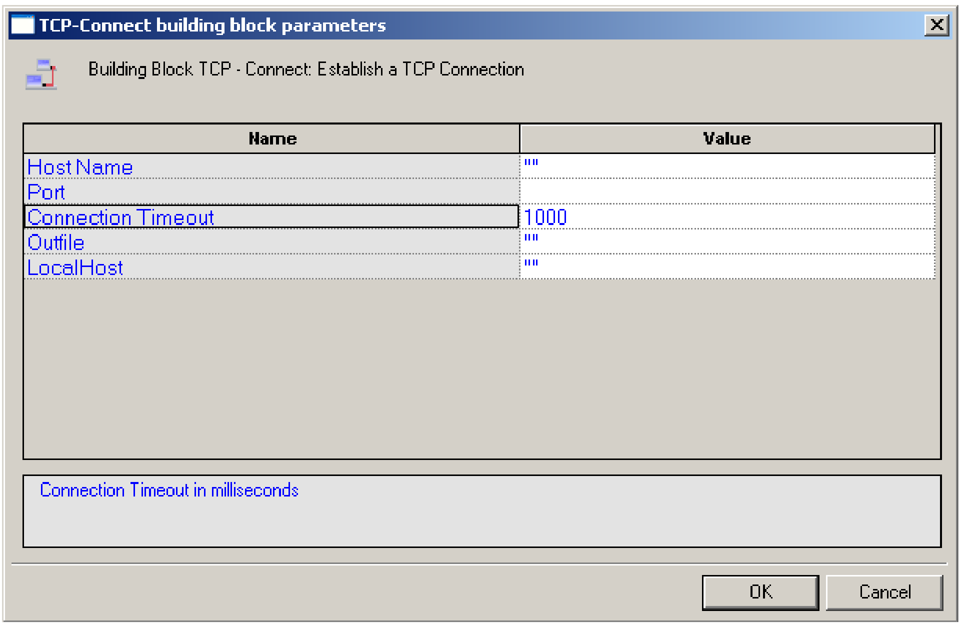
TCP-Connect

Use the TCP-Connect Building Block to open a TCP connection.

To enter a value:

1. Drag the **TCP-Connect** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The TCP-Connect Building Block parameters dialog box opens.



TCP-Connect building block parameters

Building Block TCP - Connect: Establish a TCP Connection

Name	Value
Host Name	""
Port	
Connection Timeout	1000
Outfile	""
LocalHost	""

Connection Timeout in milliseconds

Figure 168: TCP-Connect Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

For example, in the preceding figure, the comment area explains that the Connection Timeout field is used to set the amount of time the system will wait for a TCP connection to be established before timing out. Time is defined in milliseconds.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 60.
4. Click **OK**.

   The TCP-Connect Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, a TCP connection is opened using the host names specified by the user.

The fields in the TCP-Connect Building Block parameters dialog box are described in the following table:

Table 60: TCP-Connect Building Block Parameters Dialog Box Fields

Field Name	**Description**
Host Name	Specify the name of the TCP destination host. Type the TCP Host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The TCP host is identified either through a DNS number or a full name string. A host name string must be enclosed within quotation marks.
Port	Specify the port to which you are connecting. Type the port number into the input field. If you do not specify a value, the default TCP port is used.
Connection Timeout	Specify the amount of time the system will wait for a TCP connection to be established before timing out. Type the timeout value in the input field. Time is defined in milliseconds.
Outfile	Specify the name of the file into which the TCP output stream should be stored. Type the Outfile name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The file name string must be enclosed within quotation marks.

Field Name	Description
LocalHost	Specify the name of the local host. Type the local host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The local host is identified either through a DNS number or a full name string. A host name string must be enclosed within quotation marks.

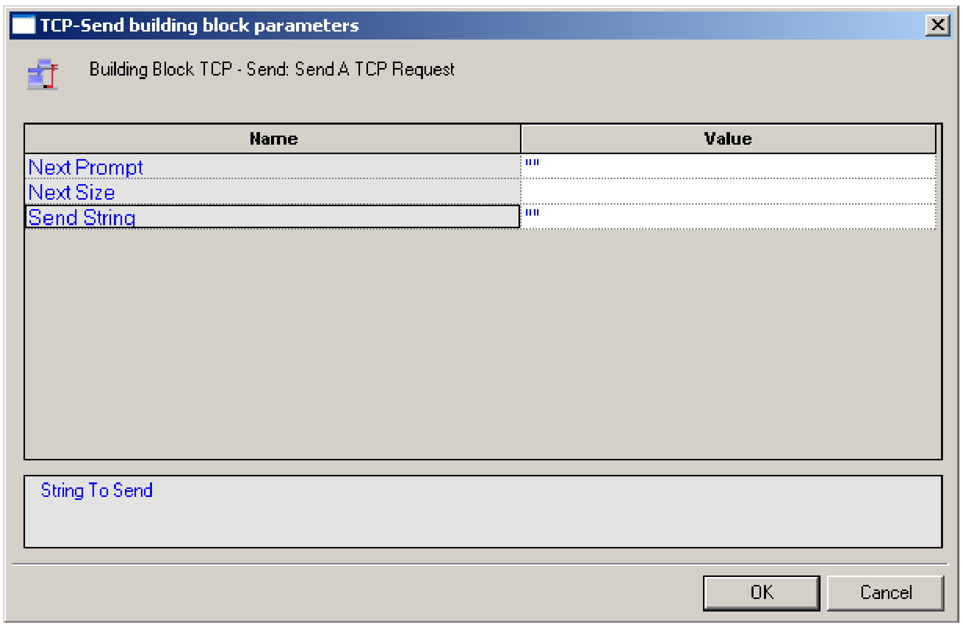
TCP-Send

Use the TCP-Send Building Block to send a TCP request.

To enter a value:

1. Drag the **TCP-Send** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The TCP-Send Building Block parameters dialog box opens.


TCP-Send building block parameters
Building Block TCP - Send: Send A TCP Request

Name	Value
Next Prompt	""
Next Size	
Send String	""

String To Send

Figure 169: TCP-Send Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Send String designates the text string to be sent.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 61.

4. Click **OK**.

The TCP-Send Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the TCP-Send Building Block parameters dialog box are described in the following table:

Table 61: TCP-Send Building Block Parameters Dialog Box Fields

Field Name	Description
Next Prompt	Specify a distinctive text string to be identified in the next string received from the host. If used, this string must appear in all communications received from the TCP host. Type the prompt string into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The string must be enclosed within quotation marks.
Next Size	Specify the size, in bytes, of the expected data. If used, this size specification limits the length of all communications received from the TCP host. Type the size value in the input area for this field.
Send String	Enter the text being sent to the TCP host. Type the string text into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The message text must be enclosed within quotation marks.

TCP-Receive

Use the TCP-Receive Building Block to return all responses from the TCP host since the last TCP-Send action. A TCP-Receive action returns to the script when the NextPrompt, NextSize, or Timeout conditions set with a previous TCP-Send action are met. If more than one of these properties is specified, the method returns to the script when the first one is met. Subsequent uses of TCP-Receive find the next instance of the limiting property, returning additional information from the buffer. The content returned depends upon which of the three limiting properties triggered the return.

To enter a value:

- Drag the **TCP-Receive** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

  The TCP-Receive Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

TCP-Erase

Use the TCP-Erase Building Block to clear the contents of the TCP document object.

To enter a value:

- Drag the **TCP-Erase** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

  The TCP-Erase Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

TELNET

Dragging a TELNET icon into your Script Tree opens a TELNET Building Block parameters dialog box.

TELNET toolbox items include:

- **TELNET-Connect**: Open a TELNET connection.
- **TELNET-Receive**: Receive a TELNET communication.
- **TELNET-Send**: Send a TELNET communication.
- **TELNET-Erase**: Clear the contents of the TELNET document object.

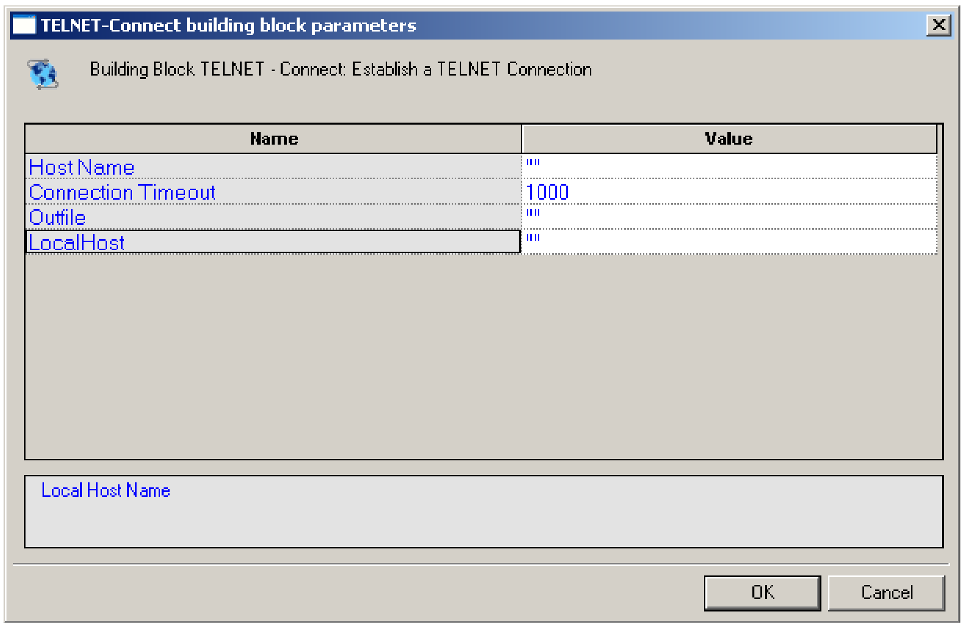
TELNET-Connect

Use the TELNET-Connect Building Block to open a TELNET connection.

To enter a value:

1. Drag the **TELNET-Connect** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The TELNET-Connect Building Block parameters dialog box opens.



Building Block TELNET - Connect: Establish a TELNET Connection

Name	Value
Host Name	""
Connection Timeout	1000
Outfile	""
LocalHost	""

Local Host Name

Figure 170: TELNET-Connect Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Local Host field is used to define the name of the local host for this TELNET session.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 62.

4. Click **OK**.

   The TELNET-Connect Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, a TELNET connection is opened using the host names specified by the user.

The fields in the TELNET-Connect Building Block parameters dialog box are described in the following table:

Table 62: TELNET-Connect Building Block Parameters Dialog Box Fields

Field Name	Description
Host Name	Specify the name of the TELNET destination host. Type the TELNET Host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The TELNET host is identified either through a DNS number or a full name string. A host name string must be enclosed within quotation marks.
Connection Timeout	Specify the amount of time the system will wait for a TELNET connection to be established before timing out. Type the timeout value in the input field. Time is defined in milliseconds.
Outfile	Specify the name of the file into which the TELNET output stream should be stored. Type the Outfile name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The file name string must be enclosed within quotation marks.
LocalHost	Specify the name of the local host. Type the local host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The local host is identified either through a DNS number or a full name string. A host name string must be enclosed within quotation marks.

TELNET-Receive

Use the TELNET-Receive Building Block to receive a TELNET communication.

To enter a value:

1. Drag the **TELNET-Receive** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The TELNET-Receive Building Block parameters dialog box opens.

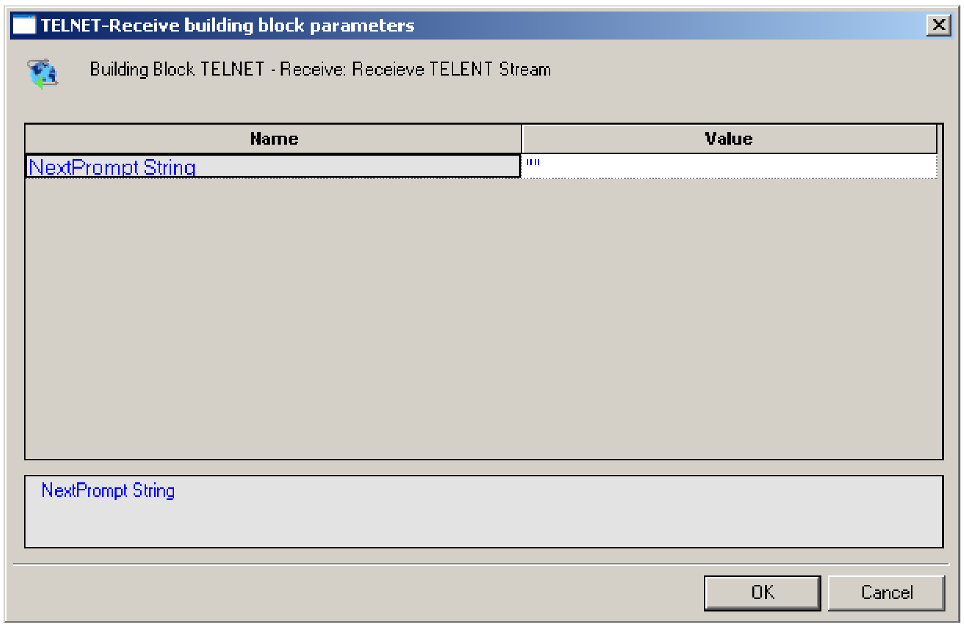

TELNET-Receive building block parameters
Building Block TELNET - Receive: Receieve TELENT Stream
Name
Value
NextPrompt String
""
NextPrompt String
OK
Cancel

Figure 171: TELNET-Receive Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the NextPrompt String designates the text string that must be found and identified in the next communication received via TELNET.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 63.

4. Click **OK**.

   The TELNET-Receive Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the TELNET-Receive Building Block parameters dialog box are described in the following table:

Table 63: TELNET-Receive Building Block Parameters Dialog Box Fields

Field Name	Description
NextPrompt String	Specify a distinctive text string to be identified in the next string received from the host. If used, this string must appear in all communications received from the TELNET host. Type the prompt string into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The string must be enclosed within quotation marks.

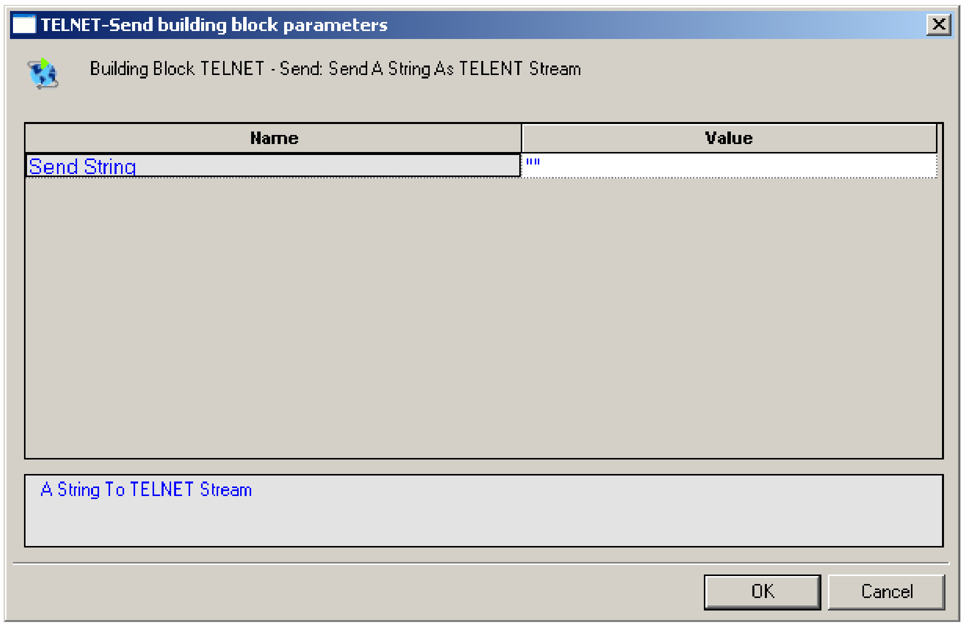
TELNET-Send

Use the TELNET-Send Building Block to send a TELNET communication.

To enter a value:

1. Drag the **TELNET-Send** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The TELNET-Send Building Block parameters dialog box opens.



TELNET-Send building block parameters

Building Block TELNET - Send: Send A String As TELENT Stream

Name	Value
Send String	""

A String To TELNET Stream

Figure 172: TELNET-Send Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Send String designates the text string to be sent.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 64.

4. Click **OK**.

   The TELNET-Send Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The field in the TELNET-Send Building Block parameters dialog box is described in the following table:

Table 64: TELNET-Send Building Block Parameters Dialog Box Field

Field Name	Description
Send String	Enter the text being sent to the TELNET host. Type the string text into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The message text must be enclosed within quotation marks.

TELNET-Erase

Use the TELNET-Erase Building Block to clear the contents of the TELNET document object.

To enter a value:

- Drag the **TELNET-Erase** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

  The TELNET-Erase Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

UDP

Dragging a UDP icon into your Script Tree opens a UDP Building Block parameters dialog box.

UDP toolbox items include:

- **UDP-Bind**: Create a connection to a UDP port.
- **UDP-Broadcast**: Broadcast data to the local net.
- **UDP-Receive**: Return all responses from the host since the last UDP-Send action.
- **UDP-Send**: Send a UDP communication.
- **UDP-Erase**: Clear the contents of the UDP document object.

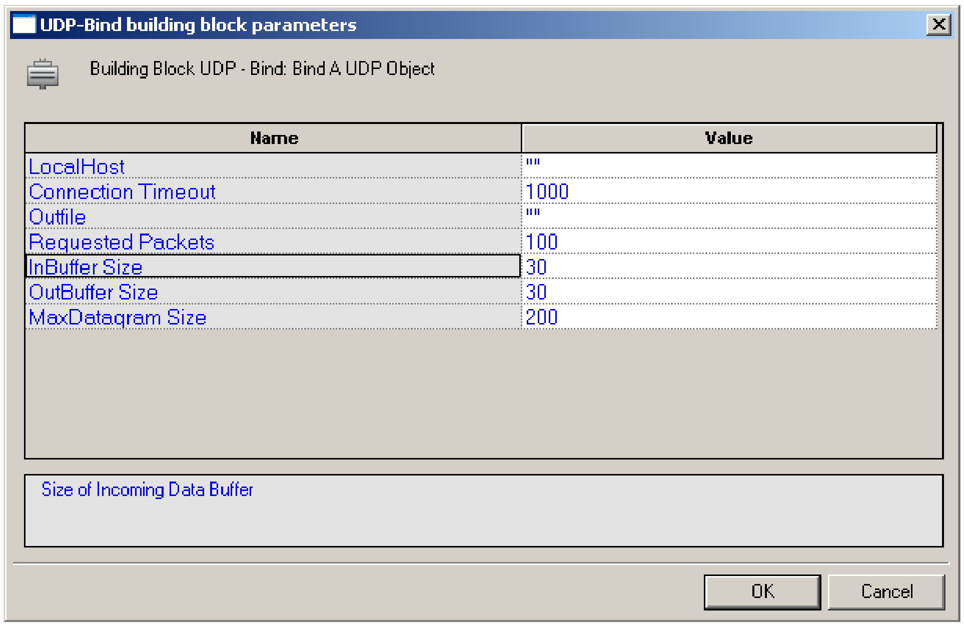
UDP-Bind

Use the UDP-Bind Building Block to create a connection to a UDP port.

To enter a value:

1. Drag the **UDP-Bind** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The UDP-Bind Building Block parameters dialog box opens.


UDP-Bind building block parameters
Building Block UDP - Bind: Bind A UDP Object
OK
Cancel

Name	Value
LocalHost	""
Connection Timeout	1000
Outfile	""
Requested Packets	100
InBuffer Size	30
OutBuffer Size	30
MaxDatagram Size	200

Size of Incoming Data Buffer

Figure 173: UDP-Bind Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the InBuffer Size field is used to define the amount of space allocated to the incoming data buffer for this UDP session.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 65.

4. Click **OK**.

   The UDP-Bind Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the `InitAgenda()` function includes commands to include the WebLOAD Recorder JIPP and UDP library files. The `InitClient()` function includes a command to define a separate UDP object for each client. Within the main body of the script, a UDP connection is opened using the connection parameters specified by the user. The `TerminateClient()` function automatically closes the connection and deletes all objects created for clients during test sessions.

The fields in the UDP-Bind Building Block parameters dialog box are described in the following table:

Table 65: UDP-Bind Building Block Parameters Dialog Box Fields

Field Name	Description
LocalHost	Specify the name of the local host. Type the local host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The local host is identified either through a DNS number or a full name string. A host name string must be enclosed within quotation marks.
Connection Timeout	Specify the amount of time the system will wait for a UDP connection to be established before timing out. Type the timeout value in the input field. Time is defined in milliseconds.
Outfile	Specify the name of the file into which the UDP output stream should be stored. Type the Outfile name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The file name string must be enclosed within quotation marks.
Requested Packets	Specify the number of requested packets per UDP communication. Type the number of requested packets per communication for this session in the Value input area. The default value is 100.
InBuffer Size	Specify the amount of space allocated to the incoming data buffer for this UDP session. Type the input buffer size for this session in the Value input area. The default value is 300.
OutBuffer Size	Specify the amount of space allocated to the outgoing data buffer for this UDP session. Type the output buffer size for this session in the Value input area. The default value is 300.
MaxDatagram Size	Specify the maximum datagram size, in bytes, for this UDP session. Type the maximum datagram size for this session in the Value input area. The default value is 200.

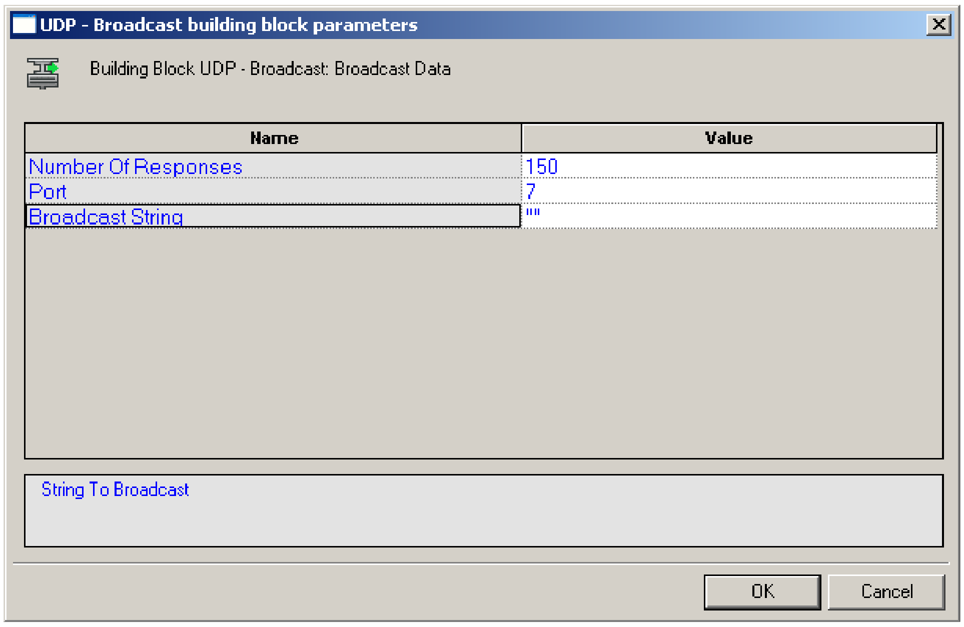
UDP-Broadcast

Use the UDP-Broadcast Building Block to broadcast data to the local net.

To enter a value:

1. Drag the **UDP-Broadcast** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The UDP-Broadcast Building Block parameters dialog box opens.



Building Block UDP - Broadcast: Broadcast Data

Name	Value
Number Of Responses	150
Port	7
Broadcast String	""

String To Broadcast

Figure 174: UDP-Broadcast Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Broadcast String field is used to define the string to be broadcast.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 66.

4. Click **OK**.

   The **UDP-Broadcast** Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   In the script, the string defined by the user is broadcast via the specified port.

The fields in the UDP-Broadcast Building Block parameters dialog box are described in the following table:

Table 66: UDP-Broadcast Building Block Parameters Dialog Box Fields

Field Name	Description
Number of Responses	Specify the number of responses the testing machine waits for before proceeding. Use this property to make sure that all network hosts have responded. To specify an unlimited number of responses, specify a Number of Responses value of zero. Type the timeout value in the input field.
Port	Specify the port to which you are connecting. Type the port number into the input field. If you do not specify a value, the default TCP port is used.
Broadcast String	Enter the text to be broadcast on the net. Type the string text into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The text must be enclosed within quotation marks.

UDP-Receive

Use the UDP-Receive Building Block to return all responses from the host since the last UDP-Send action. A UDP-Receive action is completed when either the RequestedPackets or Timeout conditions set when the UDP connection was first established is met. Subsequent uses of UDP-Receive find the next instance of the limiting property, returning additional information from the buffer.

To enter a value:

- Drag the **UDP-Receive** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

  The UDP-Receive Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

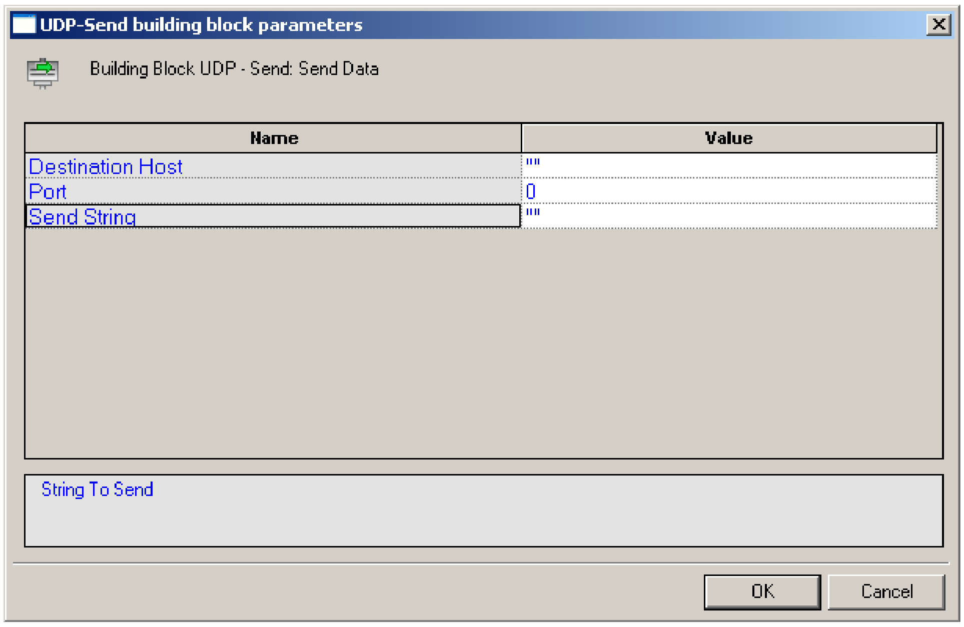
UDP-Send

Use the UDP-Send Building Block to send a UDP communication.

To enter a value:

1. Drag the **UDP-Send** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

   The UDP-Send Building Block parameters dialog box opens.



UDP-Send building block parameters

Building Block UDP - Send: Send Data

Name	Value
Destination Host	""
Port	0
Send String	""

String To Send

Figure 175: UDP-Send Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Send String designates the text string to be sent.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 67.

4. Click **OK**.

   The UDP-Send Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the UDP-Send Building Block parameters dialog box are described in the following table:

Table 67: UDP-Send Building Block Parameters Dialog Box Fields

Field Name	Description
Destination Host	Specify the name of the destination host. Type the destination host name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The destination host is identified either through a DNS number or a full name string. A host name string must be enclosed within quotation marks.
Port	Specify the port to which you are connecting. Type the port number into the input field. If you do not specify a value, the default port is used.
Send String	Enter the text being sent to the specified host. Type the string text into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The message text must be enclosed within quotation marks.

UDP-Erase

Use the UDP-Erase Building Block to clear the contents of the UDP document object.

To enter a value:

- Drag the **UDP-Erase** icon from the Internet Protocols toolbox into the Script Tree at the desired location.

  The UDP-Erase Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The WebLOAD Recorder IoT Protocols Toolbox

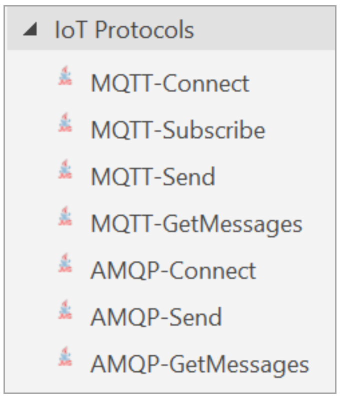
Use the WebLOAD Recorder IoT Protocols Building Blocks to simply and easily add IoT protocols functionality to your test session script without having to write numerous lines of code.

To add IoT Protocols Building Blocks to a test script directly through the WebLOAD Recorder GUI:

- Drag the selected IoT Protocols icon from the IoT Protocols toolbox and drop it into the Script Tree at the appropriate point.

  WebLOAD Recorder automatically adds the appropriate JavaScript code to your test session script.

The following are the available IoT Protocols building blocks:


IoT Protocols
MQTT-Connect
MQTT-Subscribe
MQTT-Send
MQTT-GetMessages
AMQP-Connect
AMQP-Send
AMQP-GetMessages

Each Building Block opens a different dialog box. Enter the required values in the Value field. Explanations are provided at the bottom of the dialog box for each parameter as it is selected in the dialog box.

The field descriptions in this section assume a basic familiarity with IoT protocols terminology.

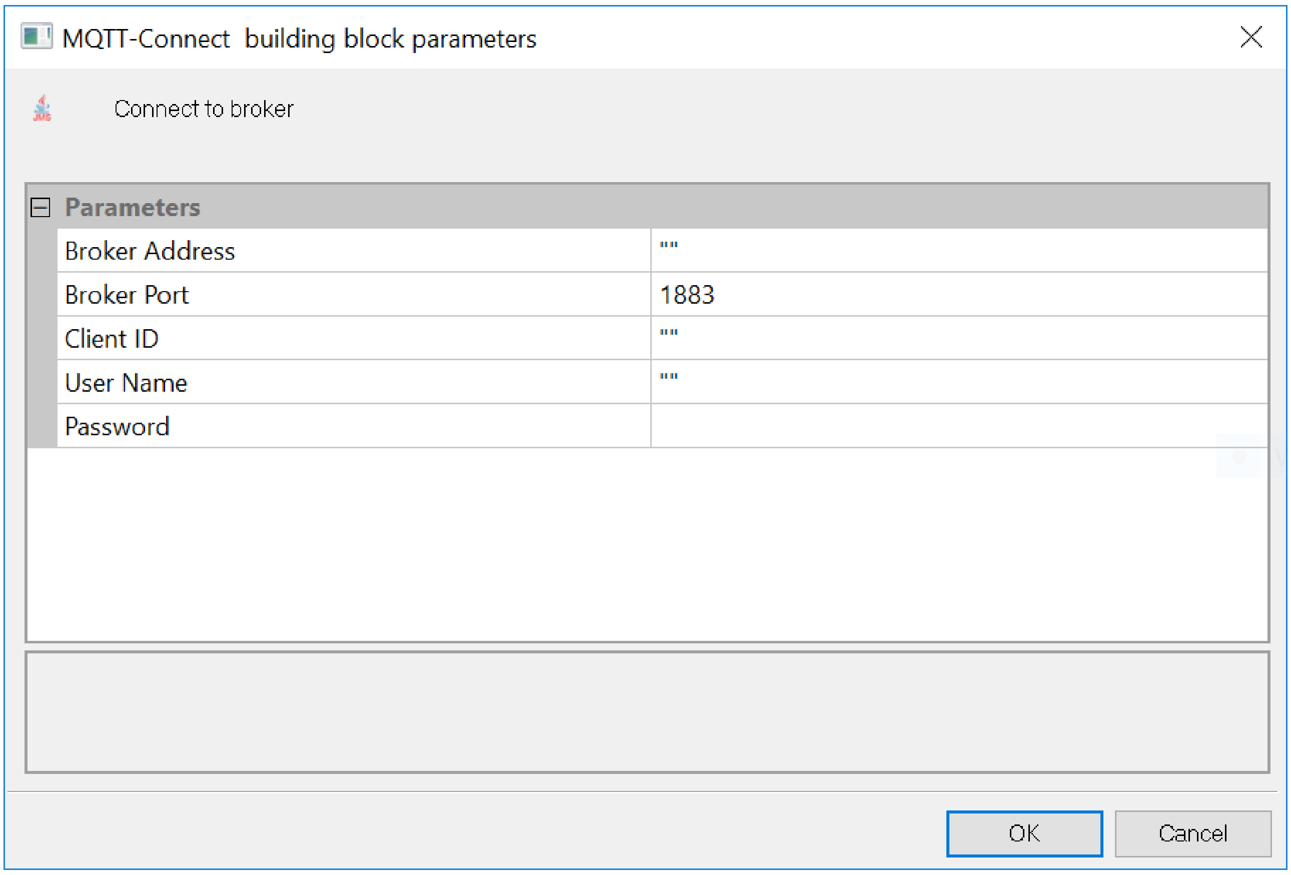
MQTT-Connect

The MQTT-Connect Building Block enables you to connect to the broker (i.e., the MQTT server).

To insert an MQTT-Connect Building Block:

1. Drag the MQTT-Connect icon from the IoT toolbox into the Script Tree at the desired location.

The MQTT-Connect Node Building Block parameters dialog box opens.



Parameters	
Broker Address	""
Broker Port	1883
Client ID	""
User Name	""
Password	

Figure 176: MQTT-Connect Node Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 68: MQTT-Connect Node Building Block Parameters Dialog Box Fields

Field Name	Description
Broker Address	The address of the MQTT broker. For example: “mqtt.example.com”
Broker Port	The MQTT Broker port number. For example: 1883.
Client ID	The MQTT client ID. In some cases this can be random text, such as “random123”.
User Name	The MQTT user name.
Password	The MQTT Client password.

3. Click **OK**.

The MQTT-Connect node is added to the Script Tree.

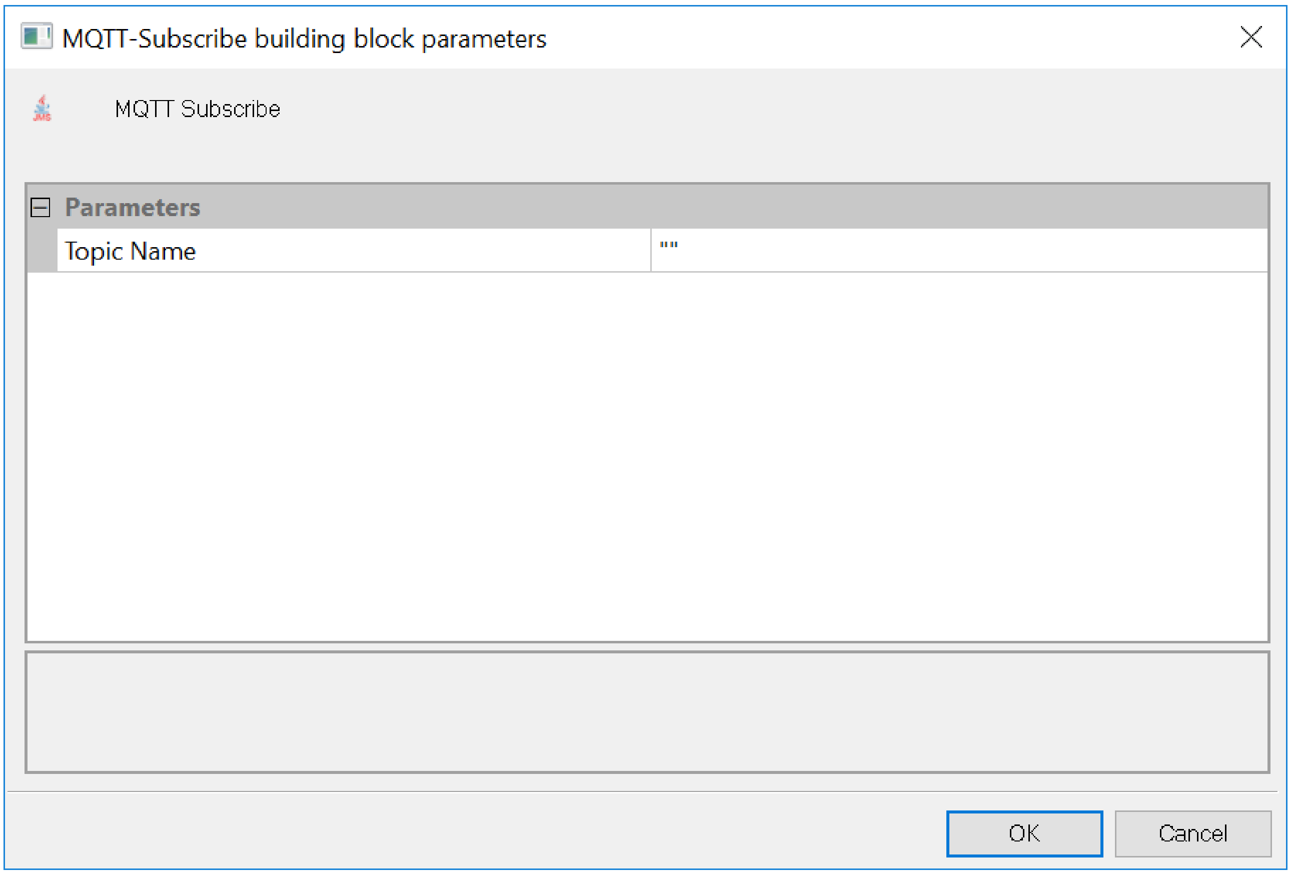
MQTT-Subscribe

The MQTT-Subscribe Building Block enables you to subscribe to a topic.

To insert an MQTT-Subscribe Building Block:

1. Drag the MQTT-Subscribe icon from the IoT toolbox into the Script Tree at the desired location.

   The MQTT-Subscribe Node Building Block parameters dialog box opens.



Topic Name	""

Figure 177: MQTT-Subscribe Node Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 69: MQTT-Subscribe Node Building Block Parameters Dialog Box Fields

Field Name	Description
Topic Name	The MQTT topic name. For example: "any topic\my topic".

3. Click **OK**.

The MQTT-Subscribe node is added to the Script Tree.

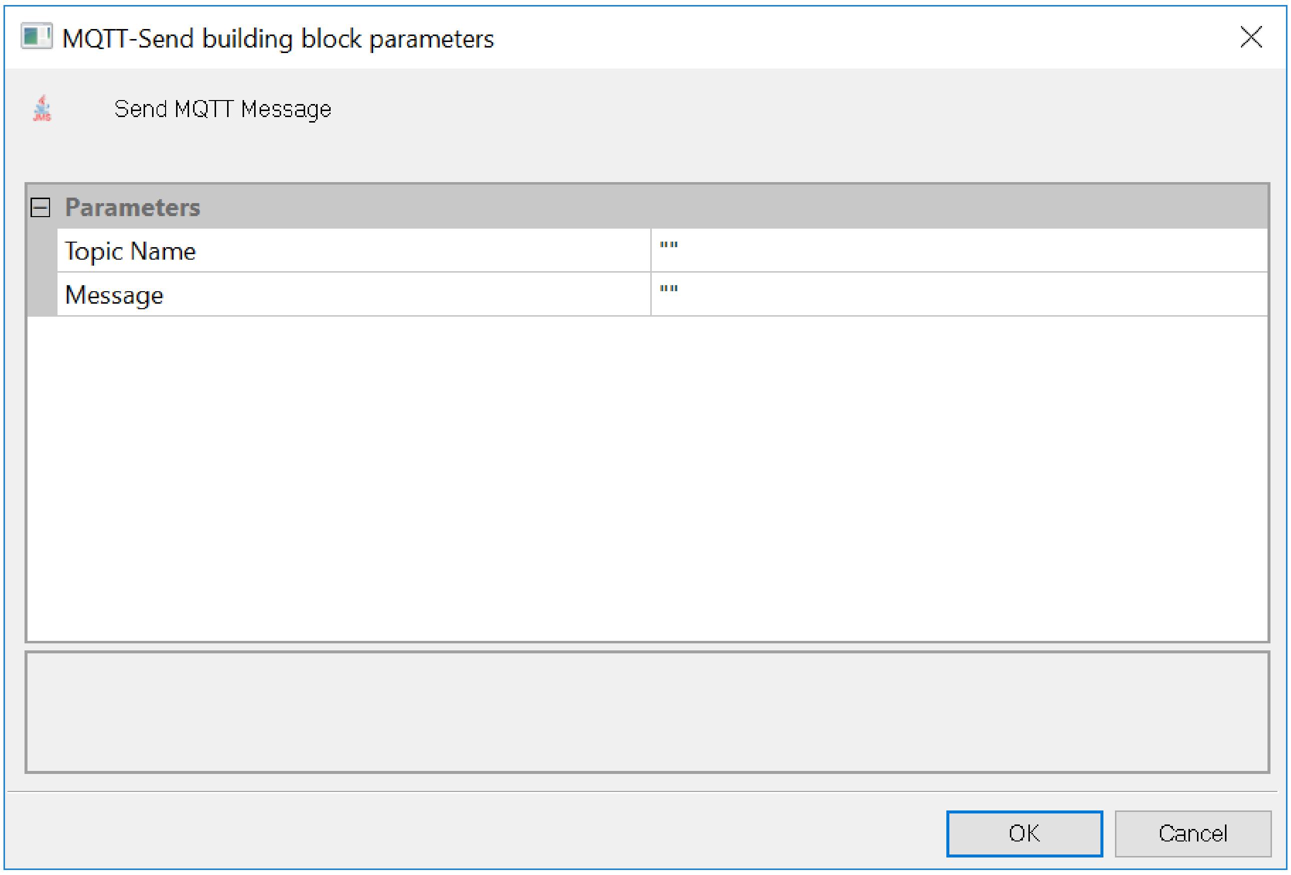
MQTT-Send

The MQTT-Send Building Block enables you to send MQTT messages.

To insert an MQTT-Send Building Block:

1. Drag the MQTT-Send icon from the IoT toolbox into the Script Tree at the desired location.

   The MQTT-Send Node Building Block parameters dialog box opens.


MQTT-Send building block parameters

Send MQTT Message

Parameters	
Topic Name	""
Message	""

Figure 178: MQTT-Send Node Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 70: MQTT-Send Node Building Block Parameters Dialog Box Fields

Field Name	Description
Topic Name	The MQTT topic name.
Message	The MQTT message to publish.

3. Click **OK**.

The MQTT-Send node is added to the Script Tree.

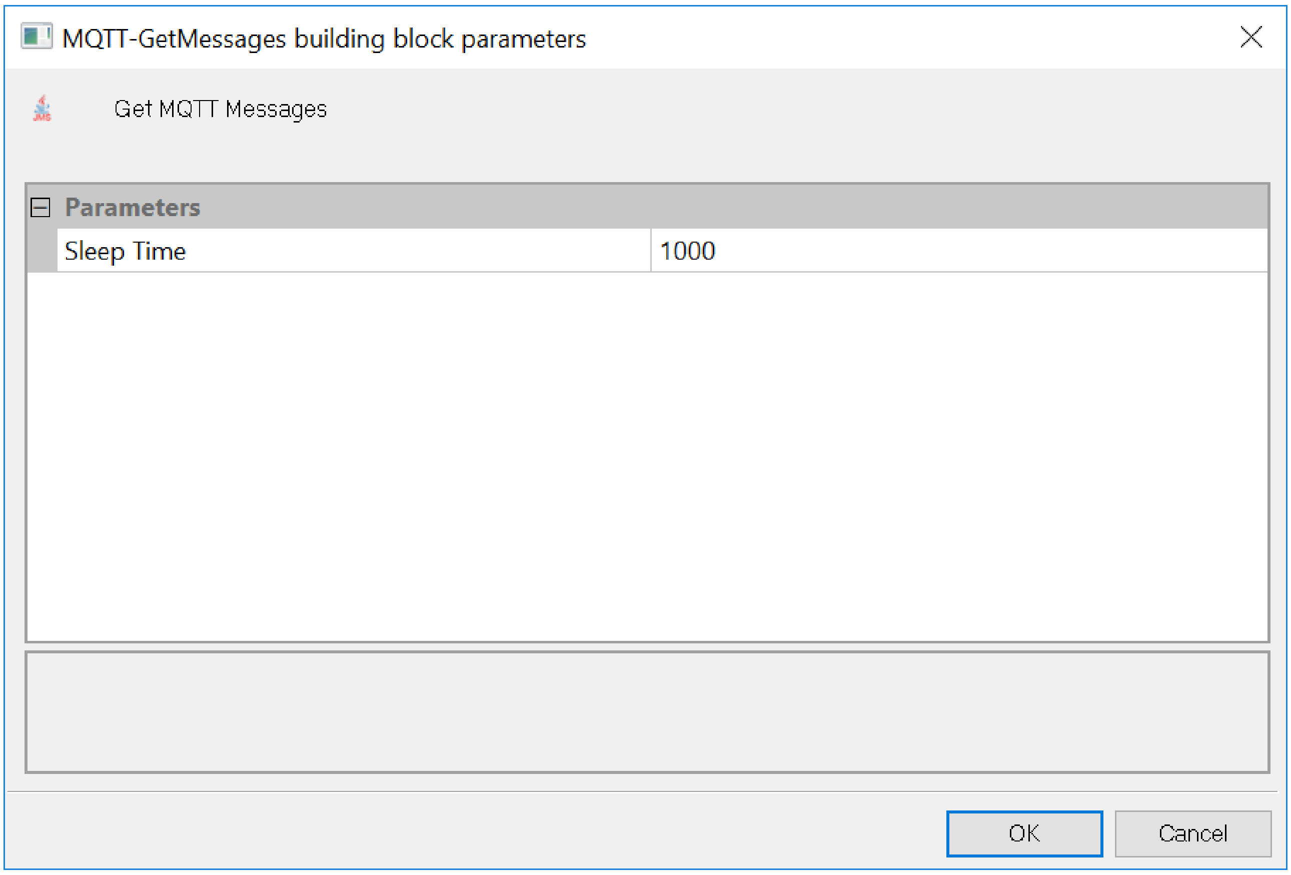
MQTT-GetMessages

The MQTT-GetMessages Building Block enables you to get MQTT messages.

To insert an MQTT-GetMessages Building Block:

1. Drag the MQTT-GetMessages icon from the IoT toolbox into the Script Tree at the desired location.

   The MQTT-GetMessages Node Building Block parameters dialog box opens.



Sleep Time	1000

Figure 179: MQTT-GetMessages Node Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 71: MQTT-GetMessages Node Building Block Parameters Dialog Box Fields

Field Name	Description
Sleep Time	The default sleep time.

3. Click **OK**.

The MQTT-GetMessages node is added to the Script Tree.

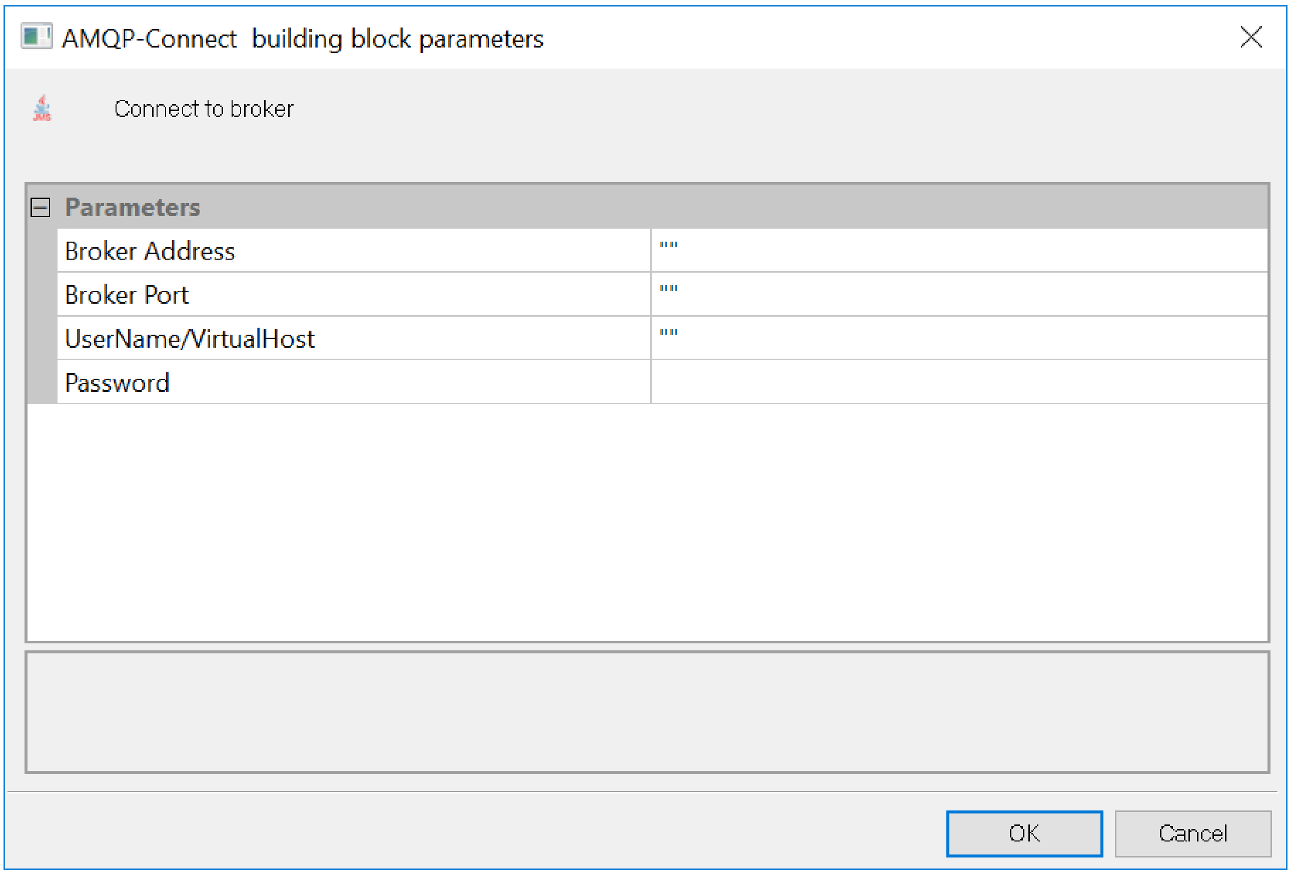
AMQP-Connect

The AMQP-Connect Building Block enables you to connect to the broker.

To insert an AMQP-Connect Building Block:

1. Drag the AMQP-Connect icon from the IoT toolbox into the Script Tree at the desired location.

   The AMQP-Connect Node Building Block parameters dialog box opens.



AMQP-Connect building block parameters

Connect to broker

Parameters	
Broker Address	""
Broker Port	""
UserName/VirtualHost	""
Password	

Figure 180: AMQP-Connect Node Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 72: AMQP-Connect Node Building Block Parameters Dialog Box Fields

Field Name	Description
Broker Address	The address of the AMQP broker.
Broker Port	The AMQP port number.
UserName/VirtualHost	The AMQP Client ID.
Password	The AMQP Client password.

3. Click **OK**.

The AMQP-Connect node is added to the Script Tree.

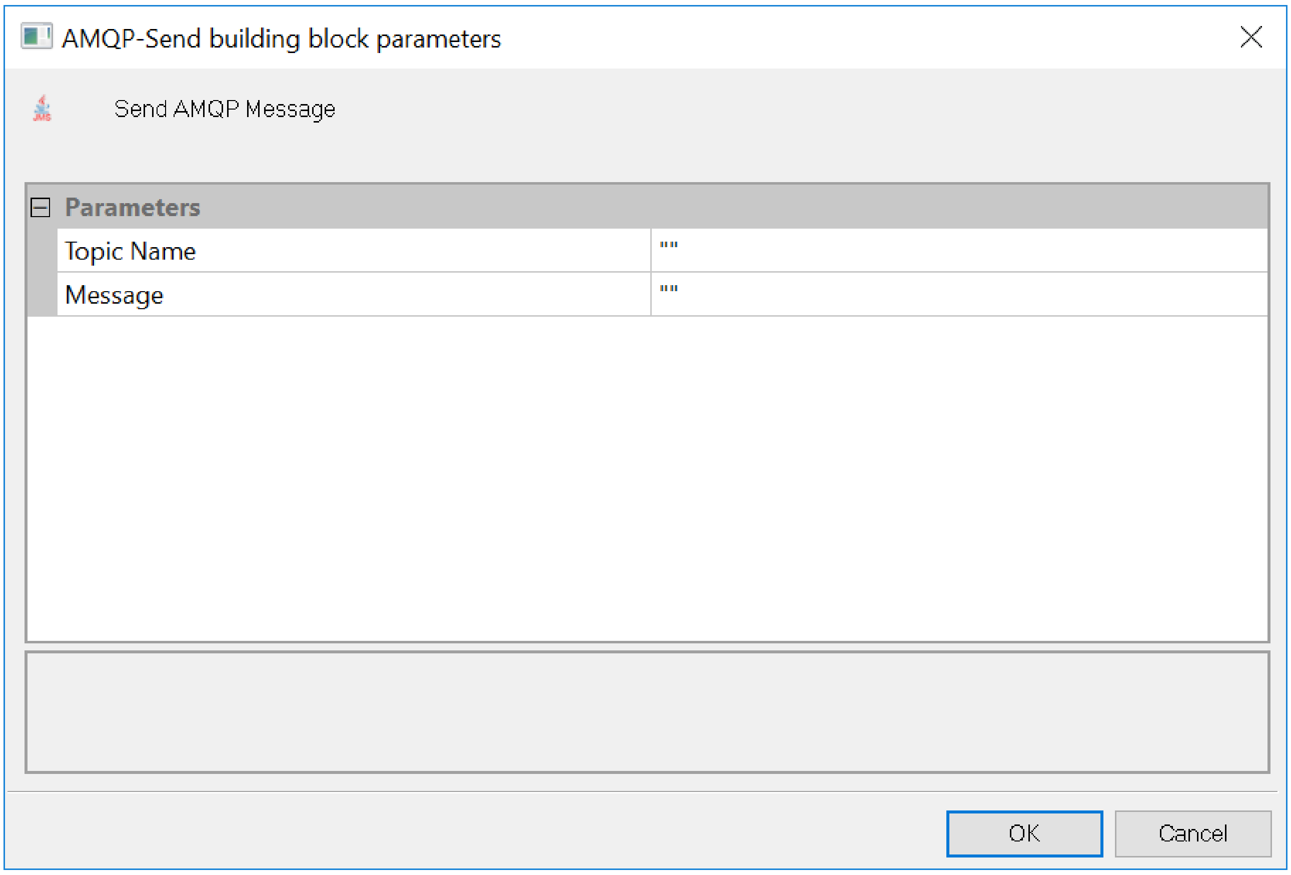
AMQP-Send

The AMQP-Send Building Block enables you to send AMQP messages.

To insert an AMQP-Send Building Block:

1. Drag the AMQP-Send icon from the IoT toolbox into the Script Tree at the desired location.

   The AMQP-Send Node Building Block parameters dialog box opens.



Parameters	
Topic Name	""
Message	""

Figure 181: AMQP-Send Node Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 73: AMQP-Send Node Building Block Parameters Dialog Box Fields

Field Name	Description
Topic Name	The AMQP topic name.

Field Name	Description
Message	The AMQP message to publish.

3. Click **OK**.

The AMQP-Send node is added to the Script Tree.

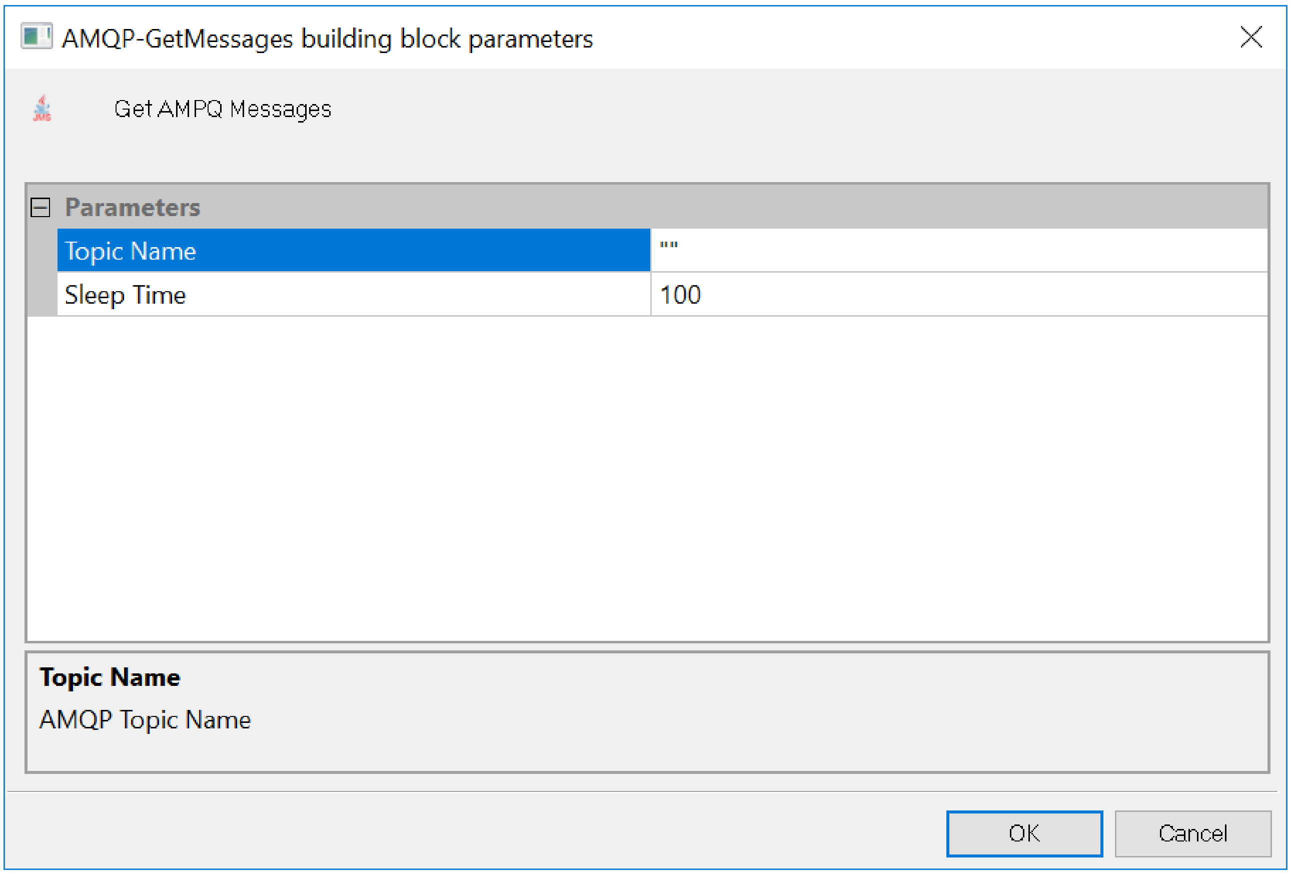
AMQP-GetMessages

The AMQP-GetMessages Building Block enables you to get AMQP messages.

To insert an AMQP-GetMessages Building Block:

1. Drag the AMQP-GetMessages icon from the IoT toolbox into the Script Tree at the desired location.

   The AMQP-GetMessages Node Building Block parameters dialog box opens.


AMQP-GetMessages building block parameters

Get AMPQ Messages

Parameters

Topic Name	""
Sleep Time	100

Topic Name
AMQP Topic Name

Figure 182: AMQP-GetMessages Node Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 74: AMQP-GetMessages Node Building Block Parameters Dialog Box Fields

Field Name	Description
Topic Name	The AMQP Topic name.
Sleep Time	The sleep time.

3. Click **OK**.

The AMQP-GetMessages node is added to the Script Tree.

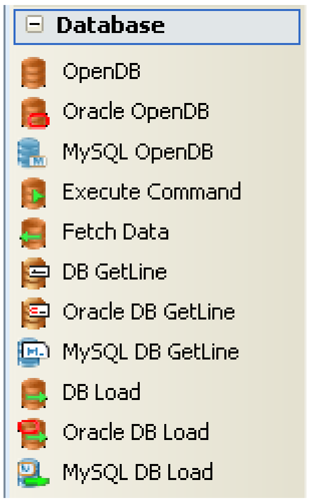
The WebLOAD Recorder Database Toolbox

The WebLOAD Recorder Database Toolbox includes a complete set of database Building Blocks. Use the WebLOAD Recorder database Building Blocks to simply and easily add database activities to your test session script.

To add database Building Blocks to a test script directly through WebLOAD Recorder:

- Drag the selected database Building Block from the Database toolbox and drop it into the Script Tree at the appropriate point.

The following are the database Building Blocks available in WebLOAD Recorder:



Each database Building Block opens a different dialog box. Enter the required values in the Value field. Explanations are provided at the bottom of the dialog box for each parameter as it is selected in the dialog box.

Note: The values that appear in the Wizard's Value area are the default values for each field. In most cases, the default value for string variables is an empty string, indicated in the Value area by a set of empty quotation marks. If you are entering your own value for a string field, the new string must also be enclosed within quotation marks. Fields that were not assigned a value in the dialog box are left as empty fields in the script code.

Once you have finished defining the new database Building Block, the new activity is reflected in the Script Tree. A database Building Block is added to the Script Tree for each database Building Block defined. WebLOAD Recorder automatically adds the corresponding JavaScript code to your test session script.

To see the complete sequence of JavaScript code for all the Database Building Blocks that have been added to the Script Tree, click the Agenda root node in the Script Tree and select the JavaScript View tab.

Notes: The JavaScript code for each of the Database Building Blocks can be found in the `DBBuildingBlocks.js` library file, `which` is part of the `Include` directory under the WebLOAD installation directory. The JavaScript code that implements these Database Building Blocks is automatically inserted to the appropriate locations within the script. Code lines may be added to the initialization phase (within the `InitAgenda()` function), in the main body of the script, or to the termination phase (within the `TerminateAgenda()` function).

The JavaScript code for that object can be edited, as described in *Using the JavaScript Editor* (on page 75).

The field descriptions in this section assume a basic familiarity with database terminology. To take full advantage of the Database Building Blocks, testers must understand how to work with ADO objects and have a basic knowledge of SQL command syntax. WebLOAD Recorder automatically inserts into the test session script the appropriate JavaScript code to implement the database commands that the tester specifies. However, it is the tester's responsibility to specify valid database commands.

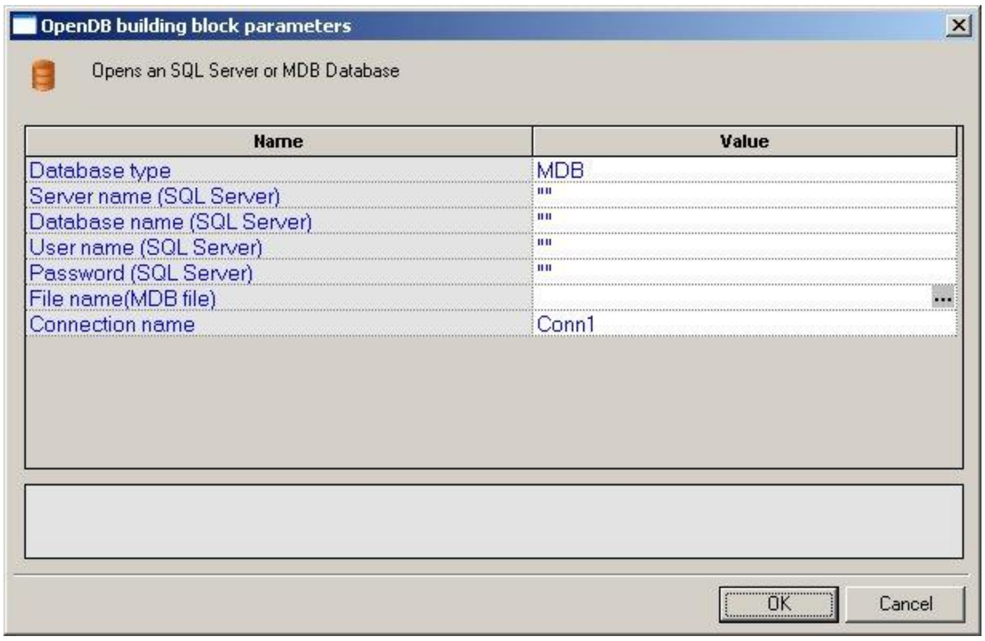
OpenDB

Use the OpenDB Building Block to open and close a specified database.

To enter a value:

1. Drag the **OpenDB** icon from the Database toolbox into the Script Tree at the desired location.

   The OpenDB Building Block parameters dialog box opens.


OpenDB building block parameters

Opens an SQL Server or MDB Database

Name	Value
Database type	MDB
Server name (SQL Server)	""
Database name (SQL Server)	""
User name (SQL Server)	""
Password (SQL Server)	""
File name(MDB file)	
Connection name	Conn1

Figure 183: OpenDB Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Database Type field is used to specify the type of database to be opened.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 75.

Note: The Database toolbox is currently available only for database activities through ADO under a Windows operating system.

4. Click **OK**.

The OpenDB Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Note: The OpenDB Building Block automatically adds the JavaScript code required to both *open* and *close* the specified database. No “CloseDB” Building Block is necessary.

The fields in the OpenDB Building Block parameters dialog box are described in the following table:

Table 75: OpenDB Building Block Parameters Dialog Box Fields

Field Name	Description
Database type	Specify the type of database to be opened. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include MS-Access and SQL Server databases.
Server name (SQL Server)	Specify the name of the machine where the database is running. Type the appropriate server name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
Database name (SQL Server)	Specify the name of the database on the SQL server. Type the appropriate database name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
User name (SQL Server)	Specify a user ID for authentication against the database. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
Password (SQL Server)	Specify a password for authentication against the database. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
File name (MDB File)	Specify the full path for an MDB file. Select the appropriate file from the Browser window that appears when you click ... to the right of the Value input area for this field. Relevant for MDB databases only.
Connection name	Specify the name of the connection variable. Type the connection name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. This connection name variable is used throughout the script file to access and work with this database.

Oracle OpenDB

Use the Oracle OpenDB Building Block to open and close an Oracle database.

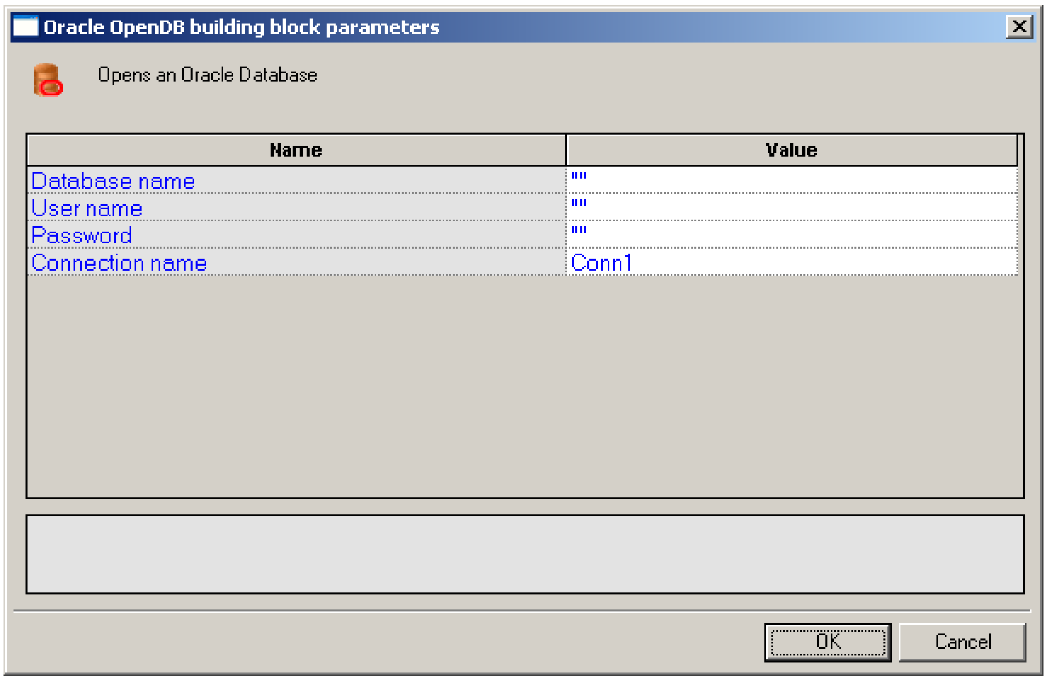
Note: For specific pre-requisites regarding the Oracle Open DB Building Blocks, refer to the *WebLOAD Installation Guide*.

Note: To use the Oracle OpenDB Building Block you must first install the Oracle Client. The Oracle Client must be installed on the same machine as the WebLOAD Recorder.

To enter a value:

1. Drag the **Oracle OpenDB** icon from the Database toolbox into the Script Tree at the desired location.

   The OpenDB Building Block parameters dialog box opens.



Name	Value
Database name	""
User name	""
Password	""
Connection name	Conn1

Figure 184: Oracle OpenDB Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Database Type field is used to specify the type of database to be opened.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 76.

Note: The Database toolbox is currently available only for database activities through ADO under a Windows operating system.

4. Click **OK**.

The Oracle OpenDB Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Note: The Oracle OpenDB Building Block automatically adds the JavaScript code required to both *open* and *close* the specified database. No "CloseDB" Building Block is necessary.

The fields in the Oracle OpenDB Building Block parameters dialog box are described in the following table:

Table 76: Oracle OpenDB Building Block Parameters Dialog Box Fields

Field Name	**Description**
Database name	Specify the name of the database on the Oracle server. Type the appropriate database name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for Oracle Server databases only.
User name	Specify a user ID for authentication against the database. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for Oracle Server databases only.
Password	Specify a password for authentication against the database. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for Oracle Server databases only.
Connection name	Specify the name of the connection variable. Type the connection name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. This connection name variable is used throughout the script file to access and work with this database.

MySQL OpenDB

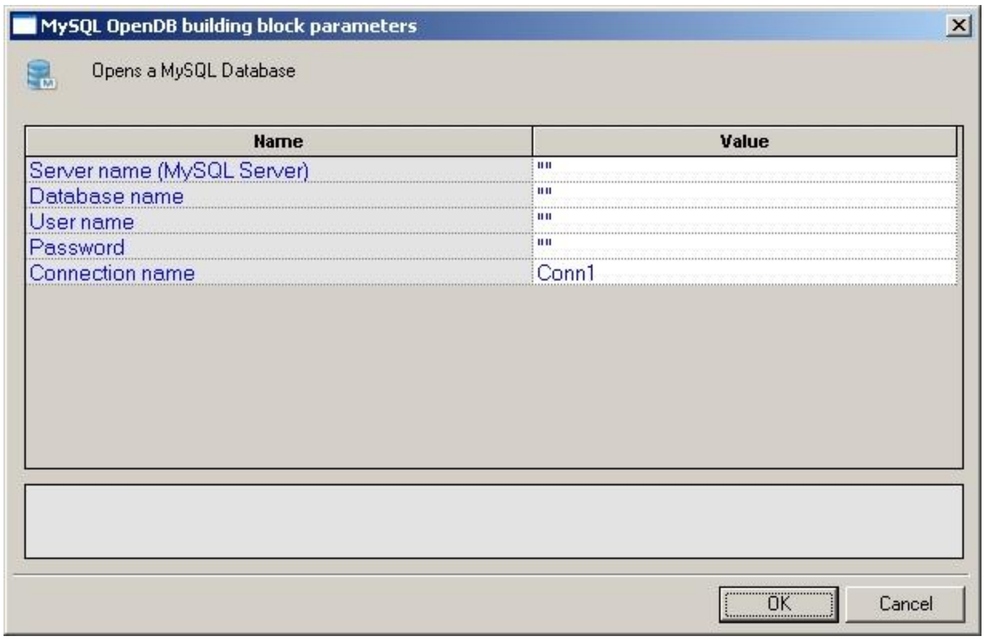
Use the MySQL OpenDB Building Block to open and close a MySQL database.

Note: Before connecting to the MySQL database, verify that the MySQL connector/ODBC 3.5.1 Driver is installed on your computer. In addition, verify that the MySQL port (3306) is open on the local/remote firewall and that the MySQL database has the right grant permission for the user and IP address from which you are connecting.

To enter a value:

1. Drag the **MySQL OpenDB** icon from the Database toolbox into the Script Tree at the desired location.

   The MySQLOpenDB Building Block parameters dialog box opens.



Name	Value
Server name (MySQL Server)	""
Database name	""
User name	""
Password	""
Connection name	Conn1

Figure 185: MySQL OpenDB Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.
3. Enter the appropriate field value into the Value column next to the field name, as described Table 77.
4. Click **OK**.

   The MySQL Open DB Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the MySQL OpenDB Building Block parameters dialog box are described in the following table:

Table 77: MySQL OpenDB Building Block Parameters Dialog Box Fields

Field Name	Description
Server name (MySQL Server)	Specify the name of the MySQL Database server.
Database name	Specify the name of the database on the MySQL Database server. Type the appropriate database name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
User name	Specify a user ID for authentication against the database. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
Password	Specify a password for authentication against the database. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
Connection name	The connection to use to open the database.

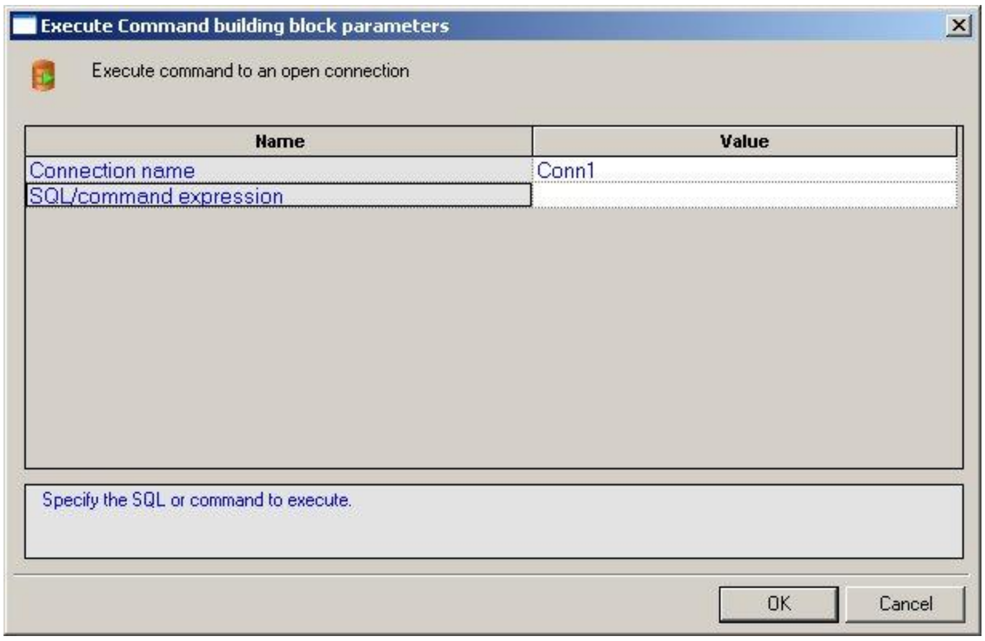
Execute Command

Use the Execute Command Building Block to add simple database commands to your test session script. The database is identified using the Connection Name variable defined through the Oracle OpenDB and OpenDB Building Blocks. The Execute Command Building Block is used for database commands that do not involve getting a return value, such as Insert, Update, and Delete.

To enter a value:

1. Drag the **Execute Command** icon from the Database toolbox into the Script Tree at the desired location.

   The Execute Command Building Block parameters dialog box opens.


Execute Command building block parameters

Execute command to an open connection

Name	Value
Connection name	Conn1
SQL/command expression	

Specify the SQL or command to execute.

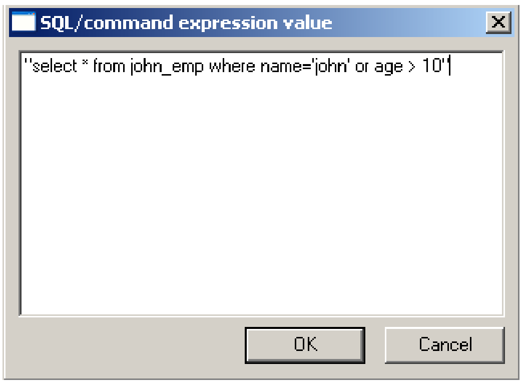
Figure 186: Execute Command Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, the comment area in the preceding figure explains that the SQL/Command Expression field is used to enter the command to be executed.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 78.

   For example, to enter a text string, type the complete text into the input-text window that appears when you click the small arrow to the right of the Value input area for the field, as illustrated in the preceding figure.


SQL/command expression value

```
"select * from john_emp where name='john' or age > 10"
```

Figure 187: Input Text Window

4. Click **OK**.

The Execute Command Building Block is added to the Script Tree and the JavaScript code is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

Note: The database connection is identified with the variable defined in the OpenDB and Oracle OpenDB Building Block parameters dialog boxes.

The fields in the Execute Command Building Block parameters dialog box are described in the following table:

Table 78: Execute Command Building Block Parameters Dialog Box Fields

Field Name	Description
Connection name	Specify the name of the connection variable. Type the connection name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The connection name variable must match the name of a database connection that was previously opened with the OpenDB Building Block. The same connection name is used throughout the script file to access and work with this database.
SQL/command expression	Specify the SQL command to be executed. Type the complete command into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.

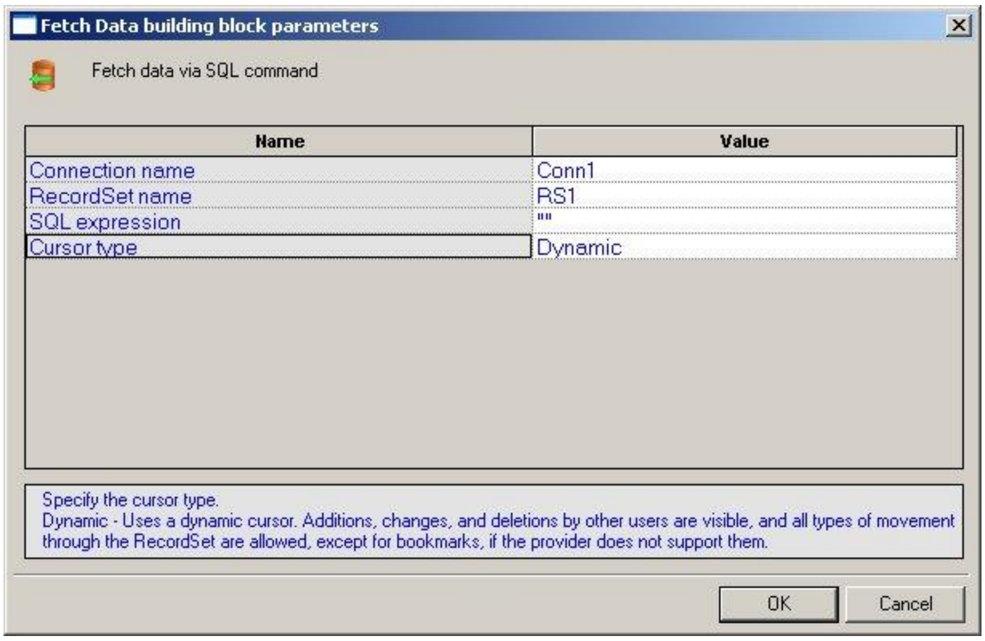
Fetch Data

Use the Fetch Data Building Block to add database commands that return data values to the script. The database is identified using the Connection Name variable defined through the OpenDB and Oracle OpenDB Building Blocks.

To enter a value:

1. Drag the **Fetch Data** icon from the Database toolbox into the Script Tree at the desired location.

   The Fetch Data Building Block parameters dialog box opens.


Fetch Data building block parameters
Fetch data via SQL command
OK
Cancel

Name	Value
Connection name	Conn1
RecordSet name	RS1
SQL expression	""
Cursor type	Dynamic

Specify the cursor type.
Dynamic - Uses a dynamic cursor. Additions, changes, and deletions by other users are visible, and all types of movement through the RecordSet are allowed, except for bookmarks, if the provider does not support them.

Figure 188: Fetch Data Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Cursor Type field is used to define the level of access and visibility requested for this database Building Block.

3. Enter the appropriate field value into the Value column next to the field name, as described Table 79.

   For example, to select a value from a pre-defined list, select the Cursor Type choice from the list of options displayed in the drop-down list box that appears when you click the small arrow to the right of the Value input area for this field.

4. Click **OK**.

   The Fetch Data Building Block is added to the Script Tree. The JavaScript code, including the `TerminateClient()` function, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the Fetch Data Building Block parameters dialog box are described in the following table:

Table 79: Fetch Data Building Block Parameters Dialog Box Fields

Field Name	**Description**
Connection name	Specify the name of the connection variable. Type the connection name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The connection name variable must match the name of a database connection that was previously opened with the OpenDB Building Block. The same connection name is used throughout the script file to access and work with this database. By default, the connection name defined in the most recent OpenDB Building Block appears in this field.
RecordSet name	Specify the name of the database RecordSet variable. Type the RecordSet name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. The same RecordSet name is used throughout the script file to access and work with records from this database.
SQL expression	Specify the SQL command to be executed. Type the complete command into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
Cursor type	Specify the level of access and visibility requested for this database Building Block. Select the Cursor Type choice from the list of options displayed in the drop-down list box that appears when you click the small arrow to the right of the Value input area for this field.

DB GetLine

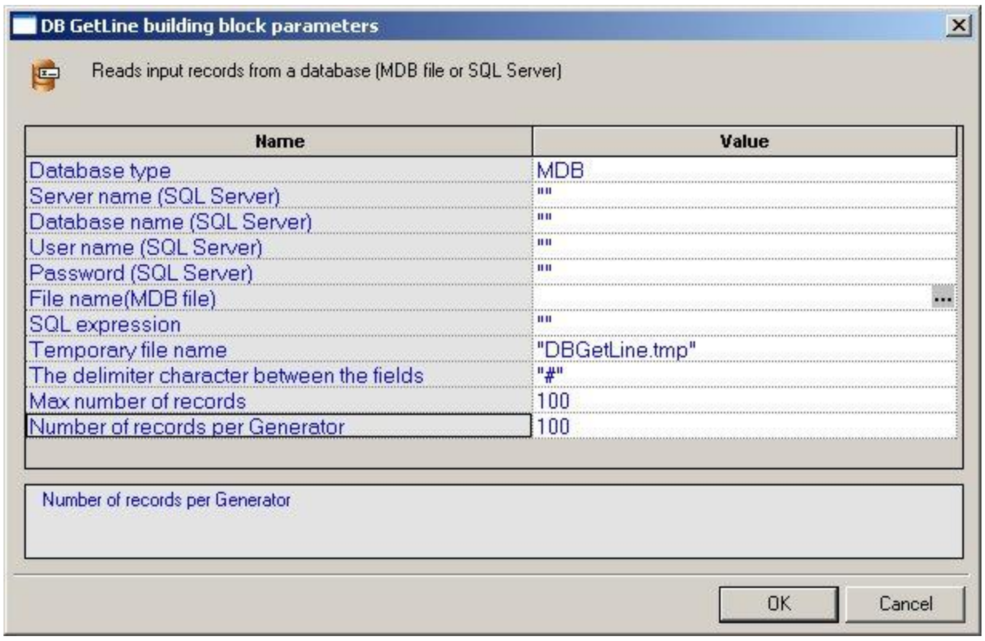
When running large load tests, the user input is usually read automatically from an input file. To read many rows of input data from a simple text file, WebLOAD uses the `GetLine()` I/O command. To read large amounts of input data from an MS-Access or SQL Server database, WebLOAD uses the equivalent DB GetLine database Building Block.

The DB GetLine Building Block reads complete records, one by one, from a specified database table. The database table is exported into a temporary file, from which the records are read, one record per line.

To enter a value:

1. Drag the **DB GetLine** icon from the Database toolbox into the Script Tree at the desired location.

   The DB GetLine Building Block parameters dialog box opens.



DB GetLine building block parameters

Reads input records from a database (MDB file or SQL Server)

Name	Value
Database type	MDB
Server name (SQL Server)	""
Database name (SQL Server)	""
User name (SQL Server)	""
Password (SQL Server)	""
File name(MDB file)	
SQL expression	""
Temporary file name	"DBGetLine.tmp"
The delimiter character between the fields	"#"
Max number of records	100
Number of records per Generator	100

Number of records per Generator

Figure 189: DB GetLine Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.
3. Enter the appropriate field value into the Value column next to the field name, as described Table 80.
4. Click **OK**.

   The DB GetLine Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()` and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the DB GetLine Building Block parameters dialog box are described in the following table:

Table 80: DB GetLine Building Block Parameters Dialog Box Fields

Field Name	Description
Database type	Specify the type of database to be opened. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include MS-Access and SQL Server databases.
Server name (SQL Server)	Specify the name of the machine where the database is running. Type the appropriate server name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
Database name (SQL Server)	Specify the name of the database on the SQL server. Type the appropriate database name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
User name (SQL Server)	Specify a user ID for authentication against the database. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
Password (SQL Server)	Specify a password for authentication against the database. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
File name (MDB File)	Specify the full path for an MDB file. Select the appropriate file from the Browser window that appears when you click the ... button to the right of the Value input area for this field. Relevant for MDB databases only.

Field Name	Description
SQL expression	Specify the SQL command to be executed. Type the complete command into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. **Note:** To take full advantage of the Database Building Blocks, testers must understand how to work with ADO objects and have a basic knowledge of SQL command syntax. WebLOAD Recorder automatically inserts into the test session script the appropriate JavaScript code to implement specified database commands. However, it is the tester who must specify the database commands to be inserted. WebLOAD Recorder cannot correct a tester's SQL syntax errors.
Temporary file name	Name to use for the temporary file that will contain the output data from the SQL statement. This temporary file will serve as an input file to the test session script. Data from this file is read line by line, where each data record is a separate line.
The delimiter character between the fields	Delimiter character that separates between the fields in each record. This delimiter character must not appear as valid character within any of the data fields. The default delimiter character is a pound sign (#). Type in a different character as needed.
Max number of records	Specifies the maximum number of records to read from the database.
Number of records per Generator	Specifies the maximum number of records to be read from the database by each generator. This field is intended for instances of testing by a network of generators. Multiple generators do not merge or share data. Database access is synchronized between generators, similar to WebLOAD Recorder's synchronization of Global Parameters. Each generator is allowed access to a specific number of records, enabling all the generators to work with the database in parallel. Record access is divided evenly between generators. The total number of records allocated to all generators must be equal to the Maximum Number of Records field value. For example, 1000 records may be divided between 10 Load Generators, with 100 records allocated per generator.

Oracle DB GetLine

When running large load tests, the user input is usually read automatically from an input file. To read many rows of input data from a simple text file, WebLOAD uses the Oracle `GetLine()` I/O command. To read large amounts of input data from an Oracle database, WebLOAD uses the equivalent Oracle DBGetLine database Building Block.

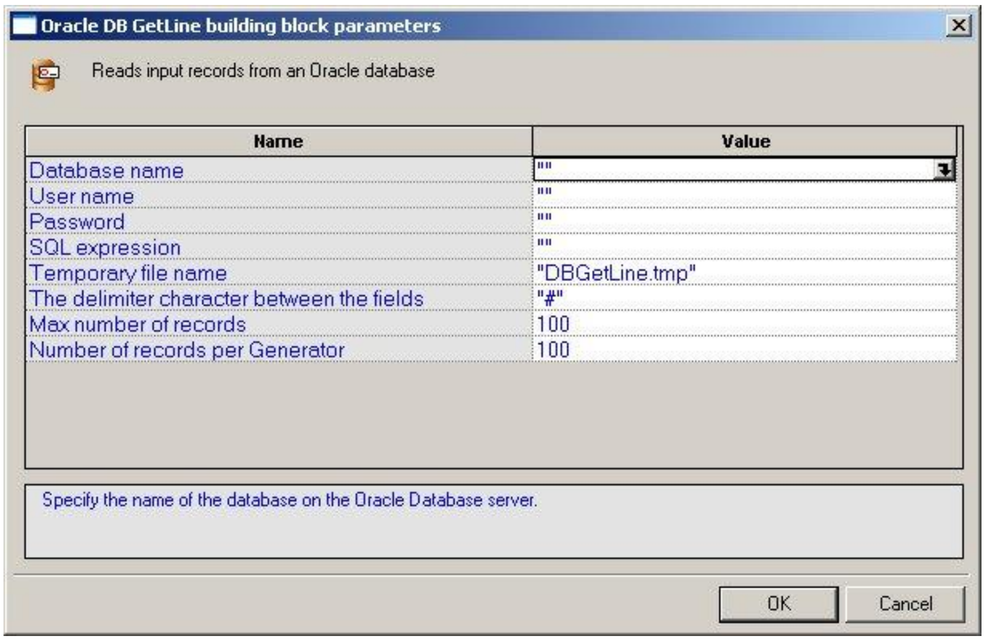
The Oracle DB GetLine Building Block reads complete records, one by one, from a specified database table. The database table is exported into a temporary file, from which the records are read, one record per line.

Note: To use the Oracle DB GetLine Building Block you must first install the Oracle Client. The Oracle Client must be installed on the same machine as the WebLOAD Recorder.

To enter a value:

1. Drag the **Oracle DB GetLine** icon from the Database toolbox into the Script Tree at the desired location.

   The Oracle DB GetLine Building Block parameters dialog box opens.



Oracle DB GetLine building block parameters

Reads input records from an Oracle database

Name	Value
Database name	""
User name	""
Password	""
SQL expression	""
Temporary file name	"DBGetLine.tmp"
The delimiter character between the fields	"#"
Max number of records	100
Number of records per Generator	100

Specify the name of the database on the Oracle Database server.

OK Cancel

Figure 190: Oracle DB GetLine Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.
3. Enter the appropriate field value into the Value column next to the field name, as described Table 81.

4. Click **OK**.

   The Oracle DB GetLine Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()` and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the **Oracle DB GetLine** Building Block parameters dialog box are described in the following table:

Table 81: Oracle DB GetLine Building Block Parameters Dialog Box Fields

Field Name	Description
Database name	Specify the name of the database on the Oracle Database server. Type the appropriate database name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
User name	Specify a user ID for authentication against the database. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
Password	Specify a password for authentication against the database. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
SQL expression	Specify the SQL command to be executed. Type the complete command into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. **Note:** To take full advantage of the Database Building Blocks, testers must understand how to work with ADO objects and have a basic knowledge of SQL command syntax. WebLOAD Recorder automatically inserts into the test session script the appropriate JavaScript code to implement specified database commands. However, it is the tester who must specify the database commands to be inserted. WebLOAD Recorder cannot correct a tester's SQL syntax errors.
Temporary file name	Name to use for the temporary file that will contain the output data from the SQL statement. This temporary file will serve as an input file to the test session script. Data from this file is read line by line, where each data record is a separate line.
The delimiter character between the fields	Delimiter character that separates between the fields in each record. This delimiter character must not appear as valid character within any of the data fields. The default delimiter character is a pound sign (#). Type in a different character as needed.

Field Name	Description
Max number of records	Specifies the maximum number of records to read from the database.
Number of records per Generator	Specifies the maximum number of records to be read from the database by each generator. This field is intended for instances of testing by a network of generators. Multiple generators do not merge or share data. Database access is synchronized between generators, similar to WebLOAD Recorder's synchronization of Global Parameters. Each generator is allowed access to a specific number of records, allowing all the generators to work with the database in parallel. Record access is divided evenly between generators. The total number of records allocated to all generators must be equal to the Maximum Number of Records field value. For example, 1000 records may be divided between 10 Load Generators, with 100 records allocated per generator.

MySQL DB GetLine

When running large load tests, the user input is usually read automatically from an input file. To read many rows of input data from a simple text file, WebLOAD uses the MySQL `GetLine()` I/O command. To read large amounts of input data from a MySQL database, WebLOAD uses the equivalent MySQL DBGetLine database Building Block.

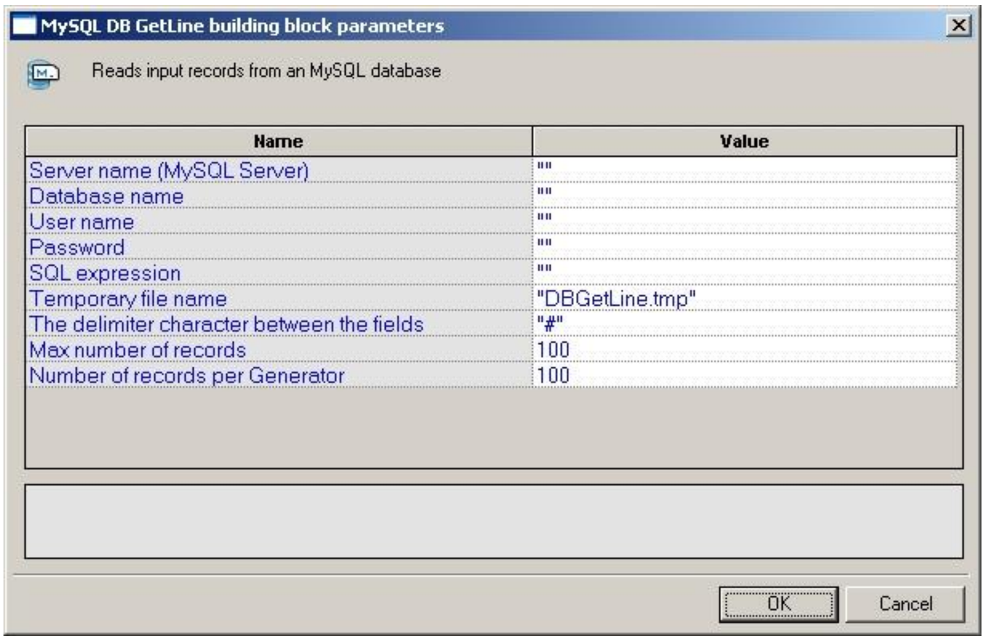
The MySQL DB GetLine Building Block reads complete records, one by one, from a specified database table. The database table is exported into a temporary file, from which the records are read, one record per line.

Note: Before connecting to the MySQL database, verify that the MySQL connector/ODBC 3.5.1 Driver is installed on your computer. In addition, verify that the MySQL port (3306) is open on the local/remote firewall and that the MySQL database has the right grant permission for the user and IP address from which you are connecting.

To enter a value:

1. Drag the **MySQL DB GetLine** icon from the Database toolbox into the Script Tree at the desired location.

   The MySQL DB GetLine Building Block parameters dialog box opens.


MySQL DB GetLine building block parameters

Reads input records from an MySQL database

Name	Value
Server name (MySQL Server)	""
Database name	""
User name	""
Password	""
SQL expression	""
Temporary file name	"DBGetLine.tmp"
The delimiter character between the fields	"#"
Max number of records	100
Number of records per Generator	100

Figure 191: MySQL DB GetLine Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.
3. Enter the appropriate field value into the Value column next to the field name, as described Table 82.
4. Click **OK**.

   The MySQL DB GetLine Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the MySQL DB GetLine Building Block parameters dialog box are described in the following table:

Table 82: MySQL DB GetLine Building Block Parameters Dialog Box Fields

Field Name	Description
Server name (MySQL Server)	Specify the name of the MySQL Database server.
Database name	Specify the name of the database on the MySQL Database server. Type the appropriate database name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
User name	Specify a user ID for authentication against the database. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.

Field Name	Description
Password	Specify a password for authentication against the database. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
SQL expression	Specify the SQL command to be executed. Type the complete command into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. **Note:** To take full advantage of the Database Building Blocks, testers must understand how to work with ADO objects and have a basic knowledge of SQL command syntax. WebLOAD Recorder automatically inserts into the test session script the appropriate JavaScript code to implement specified database commands. However, it is the tester who must specify the database commands to be inserted. WebLOAD Recorder cannot correct a tester's SQL syntax errors.
Temporary file name	Name to use for the temporary file that will contain the output data from the SQL statement. This temporary file will serve as an input file to the test session script. Data from this file is read line by line, where each data record is a separate line.
The delimiter character between the fields	Delimiter character that separates between the fields in each record. This delimiter character must not appear as valid character within any of the data fields. The default delimiter character is a pound sign (#). Type in a different character as needed.
Max number of records	Specifies the maximum number of records to read from the database.
Number of records per Generator	Specifies the maximum number of records to be read from the database by each generator. This field is intended for instances of testing by a network of generators. Multiple generators do not merge or share data. Database access is synchronized between generators, similar to WebLOAD Recorder's synchronization of Global Parameters. Each generator is allowed access to a specific number of records, allowing all the generators to work with the database in parallel. Record access is divided evenly between generators. The total number of records allocated to all generators must be equal to the Maximum Number of Records field value. For example, 1000 records may be divided between 10 Load Generators, with 100 records allocated per generator.

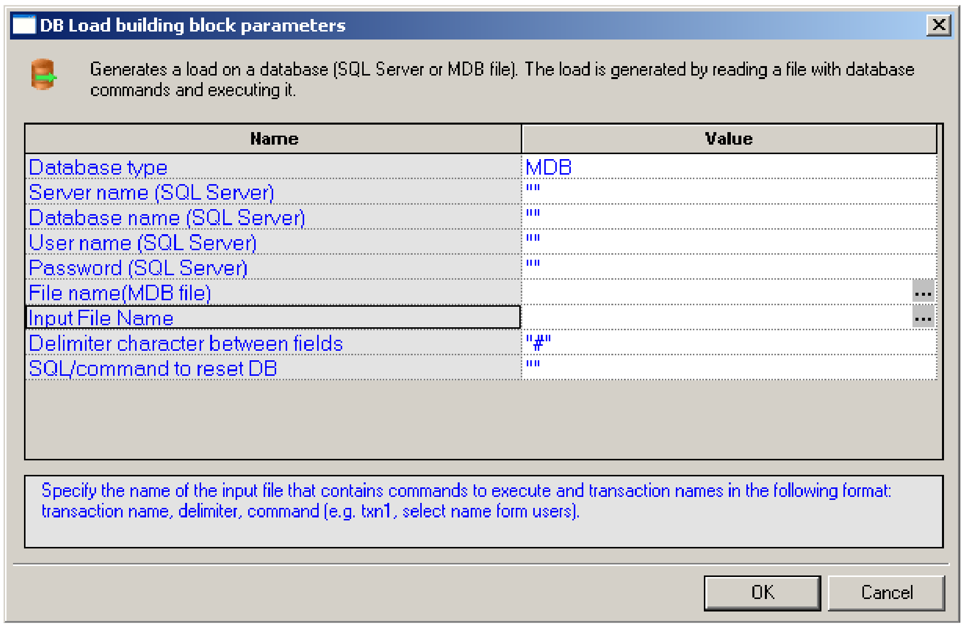
DB Load

Use the DB Load Building Block to generate a load test for the specified database. The load is generated by executing multiple iterations of database commands read from an input file. Load testing though the WebLOAD testing suite is usually scheduled only after functional testing is completed with WebLOAD Recorder.

To enter a value:

1. Drag the **DB Load** icon from the Database toolbox into the Script Tree at the desired location.

   The DB Load Building Block parameters dialog box opens.



DB Load building block parameters

Generates a load on a database (SQL Server or MDB file). The load is generated by reading a file with database commands and executing it.

Name	Value
Database type	MDB
Server name (SQL Server)	""
Database name (SQL Server)	""
User name (SQL Server)	""
Password (SQL Server)	""
File name(MDB file)	
Input File Name	
Delimiter character between fields	"#"
SQL/command to reset DB	""

Specify the name of the input file that contains commands to execute and transaction names in the following format: transaction name, delimiter, command (e.g. txn1, select name form users).

Figure 192: DB Load Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.
3. Enter the appropriate field value into the Value column next to the field name, as described Table 83.
4. Click **OK**.

   The DB Load Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the DB Load Building Block parameters dialog box are described in the following table:

Table 83: DB Load Building Block Parameters Dialog Box Fields

Field Name	Description
Database type	Specify the type of database to be opened. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include MS-Access and SQL Server databases.
Server name (SQL Server)	Specify the name of the machine where the database is running. Type the appropriate server name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
Database name (SQL Server)	Specify the name of the database on the SQL server. Type the appropriate database name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
User name (SQL Server)	Specify a user ID for authentication against the database. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
Password (SQL Server)	Specify a password for authentication against the database. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field. Relevant for SQL Server databases only.
File name (MDB file)	Specify the full path for an MDB file. Select the appropriate file from the Browser window that appears when you click ... to the right of the Value input area for this field. Relevant for MDB databases only.

Field Name	Description

Input File Name	Name of the input file that contains a set of SQL commands and transactions to be completed during this load test. Select the appropriate file from the Browser window that appears when you click **...** to the right of the Value input area for this field. Relevant for MDB databases only.
	
	A typical SQL command input file may look like the following:
	
	`Select1#select * from john_emp`
	
	`Select2#select * from john_emp where name='john'`
	
	`Select3#select * from john_emp where name='john' or age > 10`
	
	`Update#update john_emp set age = 20 where name='john'`
	
	`Insert#insert into john_emp values (99, 'zzz', 2)`
	
	`Delete#delete from john_emp where id=99`
	
	The input file consists of rows of SQL commands. As with all load testing, the commands in the input file are executed in sequence, with WebLOAD looping through the file repeatedly until the test is completed. Each SQL command line in the input file is preceded by a name identifying the transaction in which the command will be located. A pound sign (#) separates the transaction name field from the SQL command field in each row.
	
	Each SQL command is defined as a distinct HTTP transaction, enclosed in the script body within a `BeginTransaction()/EndTransaction()` set and identified by the transaction name. These transactions, like all transactions, are tracked automatically by the built-in WebLOAD timers and counters. Statistics on the performance of each transaction appear in the WebLOAD output reports, with each transaction identified in the report by name.

Delimiter character between fields	Delimiter character that separates between the fields in each record. This delimiter character must not appear as valid character within any of the data fields. The default delimiter character is a pound sign (#). Type in a different character as needed.
SQL/command to reset DB	Specify an SQL command to be executed at the end of a testing round to reset the database. This field is reserved for any “cleanup” commands that may be required in order to continue using the database for multiple iterations. Type the complete command into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.

Oracle DB Load

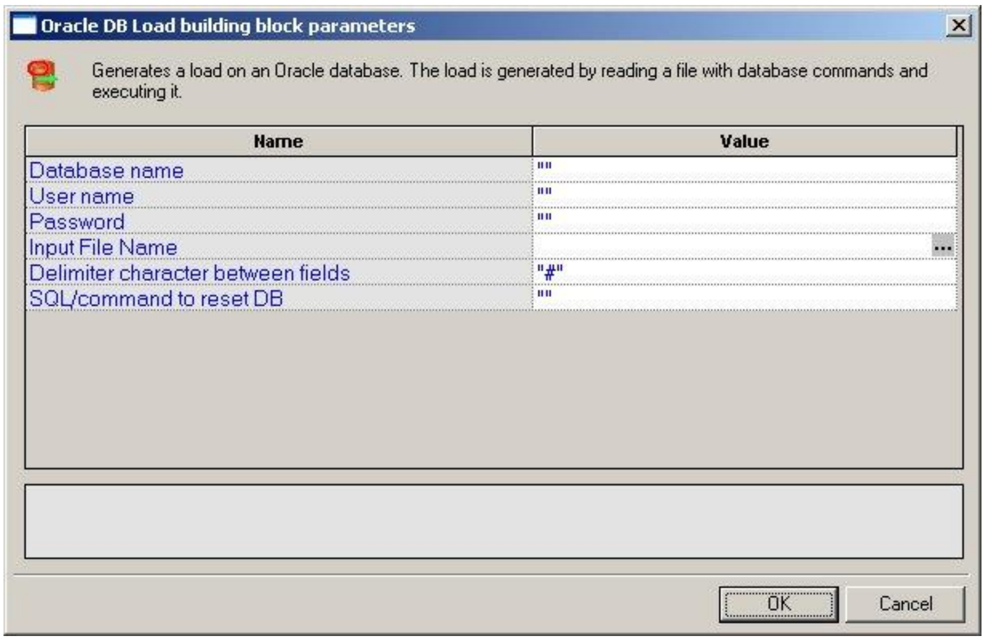
Use the Oracle DB Load Building Block to generate a load test for the specified Oracle database. The load is generated by executing multiple iterations of database commands read from an input file. Load testing though the WebLOAD testing suite is usually scheduled only after functional testing is completed with WebLOAD Recorder.

Note: To use the Oracle DB Load Building Block you must first install the Oracle Client. The Oracle Client must be installed on the same machine as the WebLOAD Recorder.

To enter a value:

1. Drag the **Oracle DB Load** icon from the Database toolbox into the Script Tree at the desired location.

   The Oracle DB Load Building Block parameters dialog box opens.



Oracle DB Load building block parameters

Generates a load on an Oracle database. The load is generated by reading a file with database commands and executing it.

Name	Value
Database name	""
User name	""
Password	""
Input File Name	
Delimiter character between fields	"#"
SQL/command to reset DB	""

Figure 193: Oracle DB Load Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.
3. Enter the appropriate field value into the Value column next to the field name, as described in Table 84.
4. Click **OK**.

   The Oracle DB Load Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()`

functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the Oracle DB Load Building Block parameters dialog box are described in the following table:

Table 84: Oracle DB Load Building Block Parameters Dialog Box Fields

Field Name	Description
Database name	Specify the name of the database on the Oracle Database server. Type the appropriate database name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
User name	Specify a user ID for authentication against the database. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
Password	Specify a password for authentication against the database. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.

Input File Name	Name of the input file that contains a set of SQL commands and transactions to be completed during this load test. Select the appropriate file from the Browser window that appears when you click the ... button to the right of the Value input area for this field. Relevant for MDB databases only.
	
	A typical SQL command input file may look like the following:
	
	`Select1#select * from john_emp`
	
	`Select2#select * from john_emp where name='john'`
	
	`Select3#select * from john_emp where name='john' or age > 10`
	
	`Update#update john_emp set age = 20 where name='john'`
	
	`Insert#insert into john_emp values (99, 'zzz', 2)`
	
	`Delete#delete from john_emp where id=99`
	
	The input file consists of rows of SQL commands. As with all load testing, the commands in the input file are executed in sequence, with WebLOAD looping through the file repeatedly until the test is completed. Each SQL command line in the input file is preceded by a name identifying the transaction in which the command will be located. A pound sign (#) separates the transaction name field from the SQL command field in each row.

Field Name	Description
Input File Name (continued)	Each SQL command is defined as a distinct HTTP transaction, enclosed in the script body within a `BeginTransaction()/EndTransaction()` set and identified by the transaction name. These transactions, like all transactions, are tracked automatically by the built-in WebLOAD timers and counters. Statistics on the performance of each transaction appear in the WebLOAD output reports, with each transaction identified in the report by name.
Delimiter character between fields	Delimiter character that separates between the fields in each record. This delimiter character must not appear as valid character within any of the data fields. The default delimiter character is a pound sign (#). Type in a different character as needed.
SQL/command to reset DB	Specify an SQL command to be executed at the end of a testing round to reset the database. This field is reserved for any "cleanup" commands that may be required in order to continue using the database for multiple iterations. Type the complete command into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.

MySQL DB Load

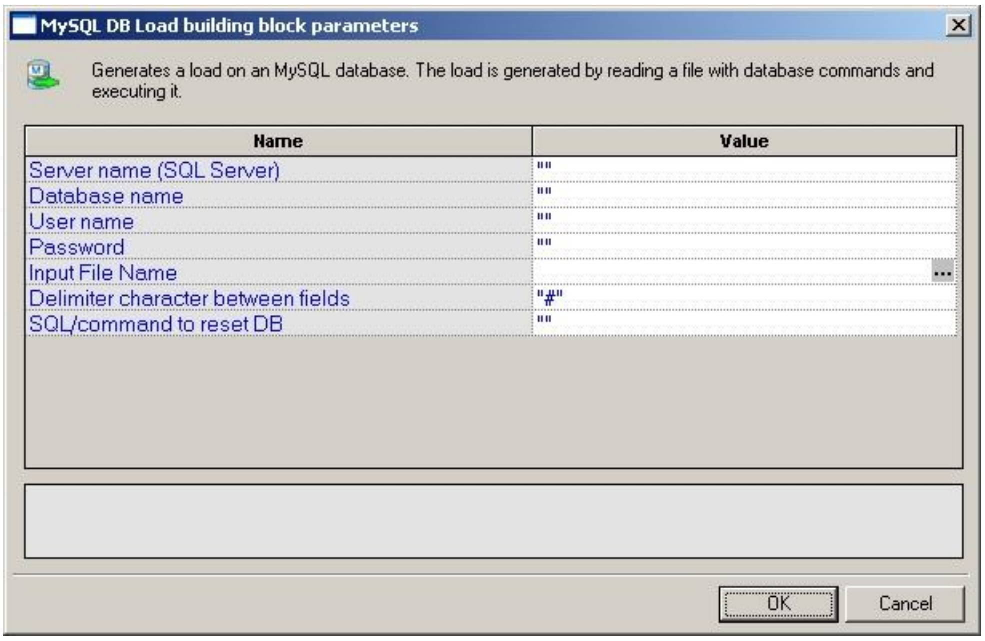
Use the MySQL DB Load Building Block to generate a load test for the specified MySQL database. The load is generated by executing multiple iterations of database commands read from an input file. Load testing though the WebLOAD testing suite is usually scheduled only after functional testing is completed with WebLOAD Recorder.

Note: Before connecting to the MySQL database, verify that the MySQL connector/ODBC 3.5.1 Driver is installed on your computer. In addition, verify that the MySQL port (3306) is open on the local/remote firewall and that the MySQL database has the right grant permission for the user and IP address from which you are connecting.

To enter a value:

1. Drag the **MySQL DB Load** icon from the Database toolbox into the Script Tree at the desired location.

   The MySQL DB Load Building Block parameters dialog box opens.


MySQL DB Load building block parameters

Generates a load on an MySQL database. The load is generated by reading a file with database commands and executing it.

Name	Value
Server name (SQL Server)	""
Database name	""
User name	""
Password	""
Input File Name	
Delimiter character between fields	"#"
SQL/command to reset DB	""

Figure 194: MySQL DB Load Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.
3. Enter the appropriate field value into the Value column next to the field name, as described Table 85.
4. Click **OK**.

   The MySQL DB Load Building Block is added to the Script Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

The fields in the MySQL DB Load Building Block parameters dialog box are described in the following table:

Table 85: MySQL DB Load Building Block Parameters Dialog Box Fields

Field Name	Description
Server name (MySQL server)	Specify the name of the MySQL Database server.
Database name	Specify the name of the database on the MySQL Database server. Type the appropriate database name into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.

Field Name	Description
User name	Specify a user ID for authentication against the database. Type the user ID into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.
Password	Specify a password for authentication against the database. Type the password into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.

Input File Name	Name of the input file that contains a set of SQL commands and transactions to be completed during this load test. Select the appropriate file from the Browser window that appears when you click the ... button to the right of the Value input area for this field. Relevant for MDB databases only.
	
	A typical SQL command input file may look like the following:
	
	`Select1#select * from john_emp`
	
	`Select2#select * from john_emp where name='john'`
	
	`Select3#select * from john_emp where name='john' or age > 10`
	
	`Update#update john_emp set age = 20 where name='john'`
	
	`Insert#insert into john_emp values (99, 'zzz', 2)`
	
	`Delete#delete from john_emp where id=99`
	
	The input file consists of rows of SQL commands. As with all load testing, the commands in the input file are executed in sequence, with WebLOAD looping through the file repeatedly until the test is completed. Each SQL command line in the input file is preceded by a name identifying the transaction in which the command will be located. A pound sign (#) separates the transaction name field from the SQL command field in each row.
	
	Each SQL command is defined as a distinct HTTP transaction, enclosed in the script body within a `BeginTransaction()/EndTransaction()` set and identified by the transaction name. These transactions, like all transactions, are tracked automatically by the built-in WebLOAD timers and counters. Statistics on the performance of each transaction appear in the WebLOAD output reports, with each transaction identified in the report by name.

Delimiter character between fields	Delimiter character that separates between the fields in each record. This delimiter character must not appear as valid character within any of the data fields. The default delimiter character is a pound sign (#). Type in a different character as needed.

Field Name	Description
SQL/command to reset DB	Specify an SQL command to be executed at the end of a testing round to reset the database. This field is reserved for any "cleanup" commands that may be required in order to continue using the database for multiple iterations. Type the complete command into the input-text window that appears when you click the small arrow to the right of the Value input area for this field.

The WebLOAD Recorder Verifications Toolbox

The following table describes the purpose of each of the WebLOAD Recorder Verifications Toolbox items:

Table 86: Verifications Toolbox Items

Script Item	Purpose
WS-Single	Verifies the value of the first element returned by the query to the Web service (WS).
WS-Multiple	Verifies the value of every element returned by the query to the Web service (WS).
Flex:Verify-Ext	Verifies data in the AMF data response.
Flex:Extract-Ext	Extracts data from AMF data response

To add Verifications Building Blocks to a test script directly through the WebLOAD Recorder:

- Drag the selected verification Building Block from the Verifications toolbox and drop it into the Script Tree immediately after the node that represents the response you wish to verify.

Each Verifications Building Block opens a different dialog box. Enter the required values in the Value fields. Explanations are provided at the bottom of the dialog box for each parameter as it is selected in the dialog box.

Note: The values that appear in the dialog box Value area are the default values for each field. In most cases, the default value for string variables is an empty string, indicated in the Value area by a set of empty quotation marks. If you are entering your own value for a string field, the new string must also be enclosed within quotation marks. Fields that were not assigned a value in the dialog box are left as empty fields in the script code.

A Verifications node is added to the Script Tree for each Verifications Building Block defined. WebLOAD Recorder automatically adds the corresponding JavaScript code to your test session script.

WS-Single

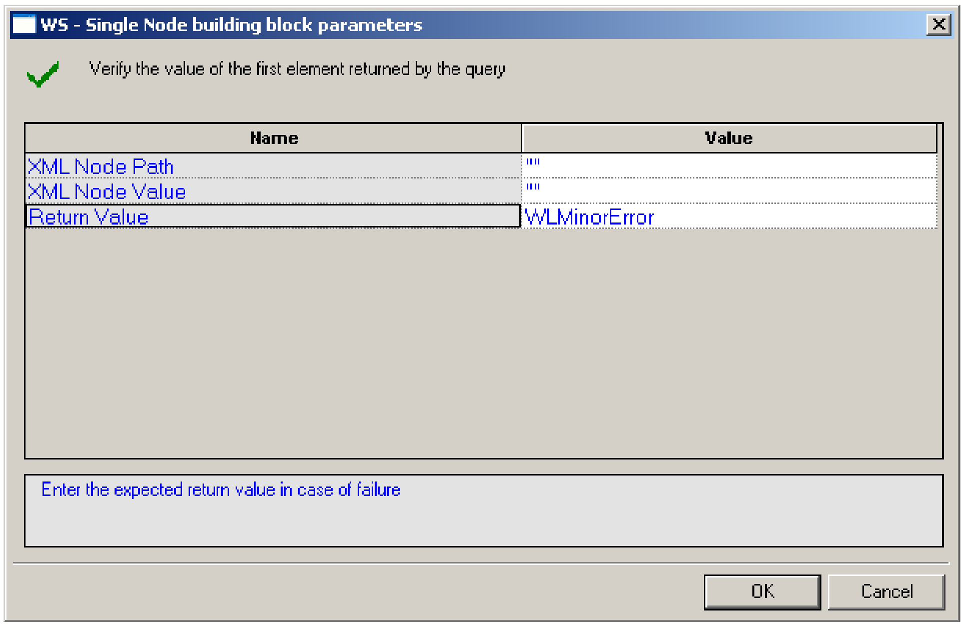
The WS-Single Building Block enables you to automatically generate a verification function of the value of the first element in a Web service's response, in your script. During playback, the results of the verification process (failure or success) are displayed in the Log View window.

If the verification succeeds, a Debug message is written to the Log View (with the element name and actual value) and the function returns WLSuccess. If the verification fails, a Warning message is displayed and the function returns WLMinorError.

To insert a WS-Single Building Block:

5. Drag the **WS-Single** icon from the Verifications toolbox into the Script Tree immediately after the node that represents the response you wish to verify.

   The WS-Single Node Building Block parameters dialog box opens.


WS - Single Node building block parameters

Verify the value of the first element returned by the query

Name	Value
XML Node Path	""
XML Node Value	""
Return Value	WLMinorError

Enter the expected return value in case of failure

Figure 195: WS-Single Node Building Block Parameters Dialog Box

6. Edit the dialog box fields according to the following table.

Table 87: WS-Single Node Building Block Parameters Dialog Box Fields

Field Name	Description
XML Node Path	The XPath query string of the object to be verified.
XML Node Value	The desired response of the verification.
Return Value	The return value in case of failure.

7. Click **OK**.

   The WS-Single node is added to the Script Tree. The JavaScript code, including the `InitAgenda()` function, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   For example:

```
function InitAgenda()
{
//Start generation for Building Block WS - Single Node

IncludeFile("wlXmlVerification.js", WLExecuteScript);
wlGlobals.SaveSource = true;
xmlDom  = InitXML();

//End generation for Building Block WS - Single Node
}
/***** WLIDE - WS - Single Node - ID:2 *****/

VerifyXMLNode(document.wlSource, "//Result", "2")

// END WLIDE
```

Note: After the verification function is created in the script, you can duplicate it several times within the script to verify different response values.

WS-Multiple

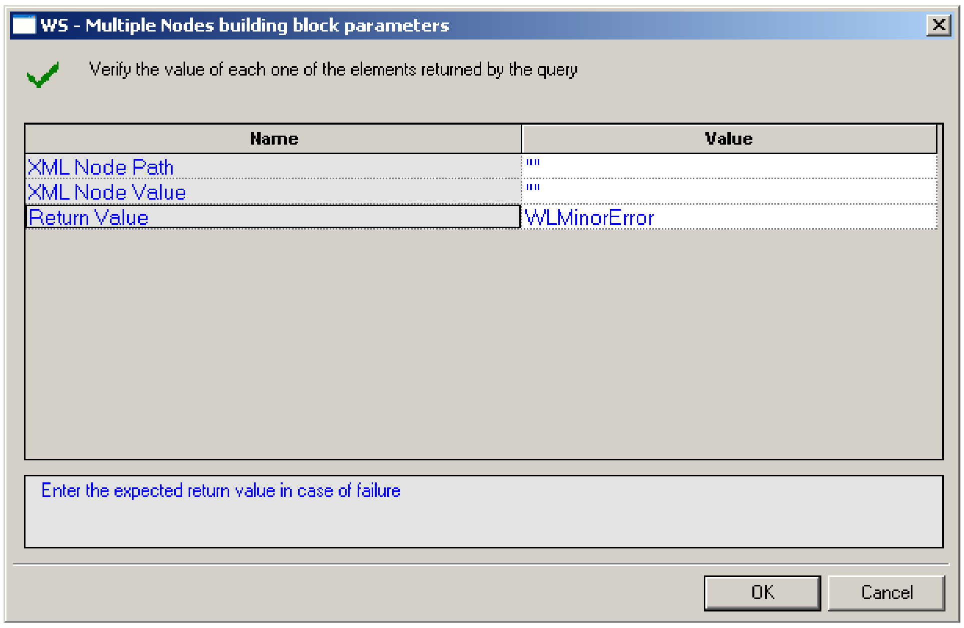
The WS-Multiple Building Block enables you to automatically generate a verification function of the values of every element in a Web service's response, in your script. During playback, the result of the verification process (failure or success) is displayed in the Log View window.

If the verification succeeds, a Debug message is written to the Log View (with the element name and actual value) and the function returns WLSuccess. If the verification fails, a Warning message is displayed and the function returns WLMinorError.

To insert a WS-Multiple Building Block:

1. Drag the **WS-Multiple** icon from the Verifications toolbox into the Script Tree immediately after the node that represents the response you wish to verify.

   The WS-Multiple Nodes Building Block parameters dialog box opens.



Verify the value of each one of the elements returned by the query

Name	Value
XML Node Path	""
XML Node Value	""
Return Value	WLMinorError

Enter the expected return value in case of failure

Figure 196: WS-Multiple Nodes Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 88: WS-Multiple Nodes Building Block Parameters Dialog Box Fields

Field Name	Description
XML Node Path	The XPath query string of the object to be verified.
XML Node Value	The desired response of the verification.
Return Value	Select the return value in case of failure.

3. Click **OK**.

   The WS-Multiple node is added to the Script Tree. The JavaScript code, including the `InitAgenda()` function, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   For example:

```
function InitAgenda(){
//Start generation for Building Block WS - Multiple
Nodes

IncludeFile("wlXmlVerification.js", WLExecuteScript);
wlGlobals.SaveSource = true;
xmlDom  = InitXML();

//End generation for Building Block WS - Multiple Nodes
}
/***** WLIDE - WS - Multiple Nodes - ID:3 *****/

VerifyXMLNodes(document.wlSource, "//Result", "5")

// END WLIDE
```

Note: After the verification function is created in the script, you can duplicate it several times within the script to verify different response values.

Flex:Verify-Ext

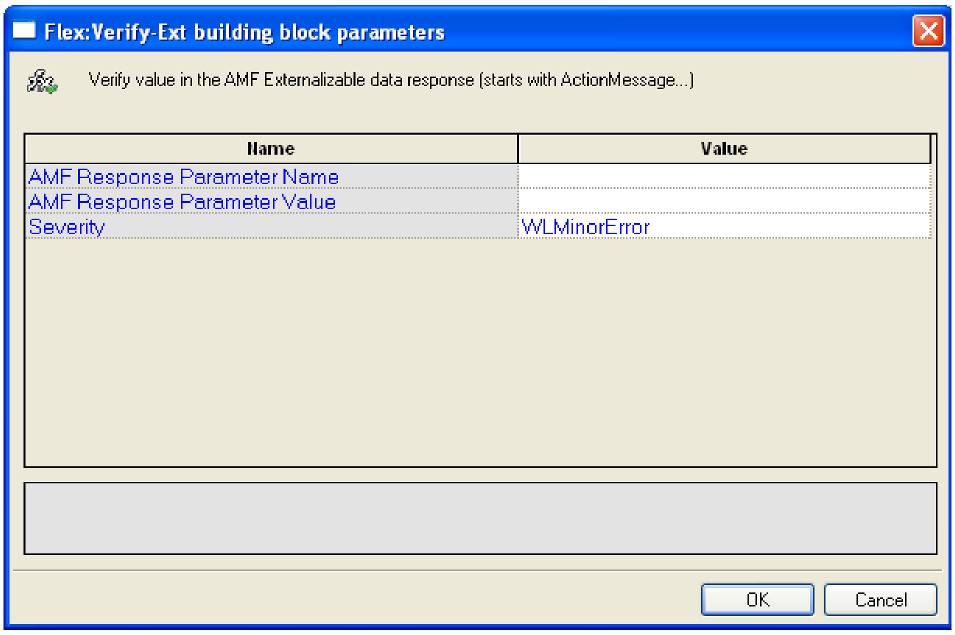
The Flex:Verify-Ext Building Block enables you to automatically generate a verification function of the data in the AMF data response, in your script. During playback, the results of the verification process (failure or success) are displayed in the Log View window.

If the verification succeeds, a Debug message is written to the Log View (with the element name and actual value) and the function returns WLSuccess. If the verification fails, a Warning message is displayed and the function returns WLMinorError.

To insert a Flex:Verify-Ext Building Block:

1. Drag the **Flex:Verify-Ext** icon from the Verifications toolbox into the Script Tree immediately after the node that represents the AMF data response you wish to verify.

The Flex:Verify-Ext Building Block parameters dialog box opens.



Flex:Verify-Ext building block parameters

Verify value in the AMF Externalizable data response (starts with ActionMessage...)

Name	Value
AMF Response Parameter Name	
AMF Response Parameter Value	
Severity	WLMinorError

Figure 197: Flex:Verify-Ext Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 89: Flex:Verify-Ext Building Block Parameters Dialog Box Fields

Field Name	Description
AMF Response Parameter Name	The path to the relevant AMF response element.
AMF Response Parameter Value	The value of the AMF response's parameter.
Severity	Select the return value in case of failure.

3. Click **OK**.

   The Flex:Verify-Ext node is added to the Script Tree. The JavaScript code, including the `InitAgenda()` function, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   For example:

```
function InitAgenda()
{

//Start generation for Building Block Flex:Verify-Ext

        IncludeFile("amfVerification.js");
//End generation for Building Block Flex:Verify-Ext
}
/***** WLIDE - Flex:Verify-Ext - ID:11 *****/

AMFResponse = new
Packages.com.radview.amf.WLAmfMessage(getAmfDataAsJsStri
ng());
VerifyAMFExt(AMFResponse,"ActionMessage.bodies(0).data.b
ody(1).category", "AMF test3", WLMinorError)

// END WLIDE
```

Note: After the verification function is created in the script, you can duplicate it several times within the script to verify different response values.

Flex:Extract-Ext

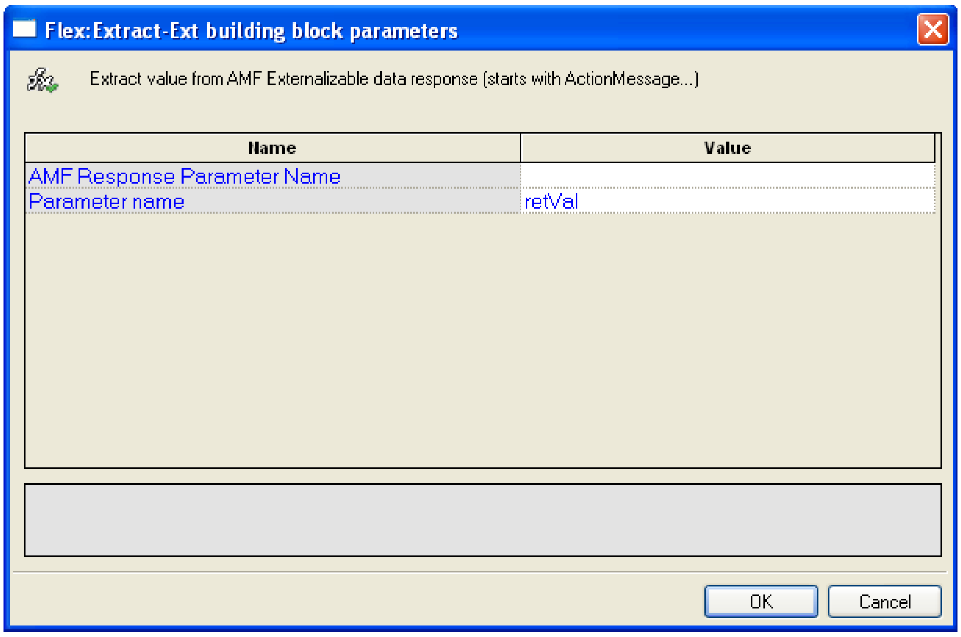
The Flex:Extract-Ext Building Block enables you to automatically extract data from an AMF data response, in your script. During playback, the extracted data is displayed in the Log View window.

If the extraction succeeds, a Debug message is written to the Log View (with the element name and actual value). If the extraction fails, the function returns NULL and a warning message is displayed.

To insert a Flex:Extract -Ext Building Block:

1. Drag the **Flex: Extract-Ext** icon from the Verifications toolbox into the Script Tree immediately after the node that represents the AMF data response from which you wish to extract data.

The Flex:Extract-Ext Building Block parameters dialog box opens.



Extract value from AMF Externalizable data response (starts with ActionMessage...)

Name	Value
AMF Response Parameter Name	
Parameter name	retVal

Figure 198: Flex:Extract-Ext Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 90: Flex:Extract-Ext Building Block Parameters Dialog Box Fields

Field Name	Description
AMF Response Parameter Name	The path to the relevant AMF response element.
Parameter Name	The name of the parameter to assign.

3. Click OK.

   The Flex:Extract-Ext node is added to the Script Tree. The JavaScript code, including the `InitAgenda()` function, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   For example:

```
retVal =
extractAMFValueExt("ActionMessage.bodies(0).data.body(1)
.category");
```

Note: After the extraction function is created in the script, you can duplicate it several times within the script to extract data from different AMF data responses.

The WebLOAD Recorder WebSocket Toolbox

Use the WebLOAD WebSocket Building Blocks to simply and easily add WebSocket functionality to your test session script. For a full description of the available WebSocket functionality, refer to the *WebLOAD™ JavaScript Reference Guide*.

WebSocket Connect

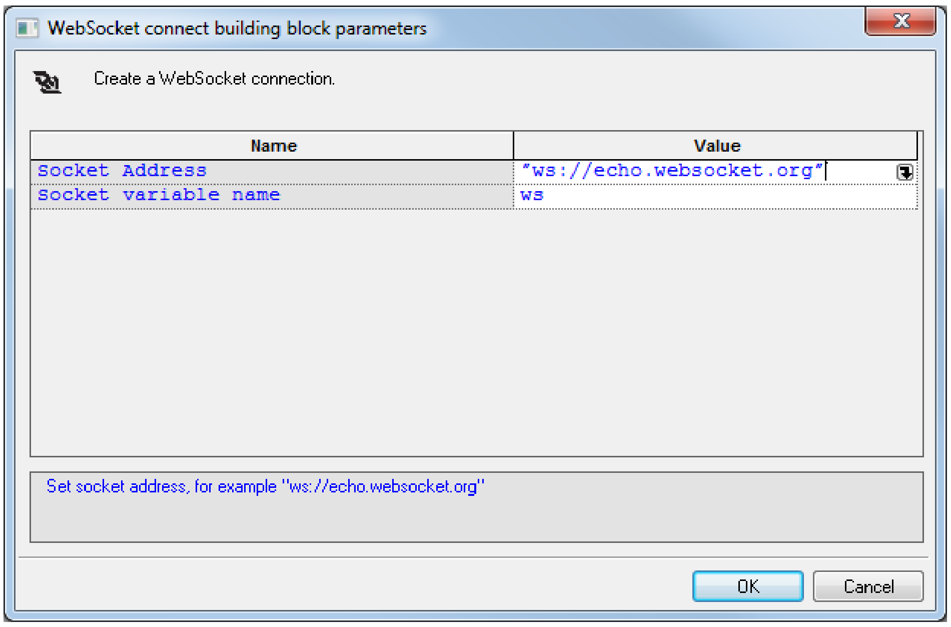
Use the WebSocket Connect Building Block to create a WebSocket connection to a specific URL address.

Note: A WebSocket emits event messages. In order to process received event messages, event handlers needs to be defined for the WebSocket. For instructions, refer to the *WebLOAD™ JavaScript Reference Guide.*

To insert a WebSocket Connect Building Block:

1. Drag the **WebSocket Connect** icon from the WebSocket toolbox into the Script Tree at the desired location.

   The WebSocket Connect Building Block parameters dialog box opens.



WebSocket connect building block parameters

Create a WebSocket connection.

Name	Value
Socket Address	"ws://echo.websocket.org"
Socket variable name	ws

Set socket address, for example "ws://echo.websocket.org"

Figure 199: WebSocket Connect Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Socket Address field is used to define the URL to which to connect.

3. Enter the appropriate field value into the Value column next to the field name, as described in Table 91.

Table 91: WebSocket Connect Building Block Parameters Dialog Box Fields

Field Name	Description
Socket Address	Specify the URL to which to connect.
Socket variable name	Specify a name for this WebSocket object.

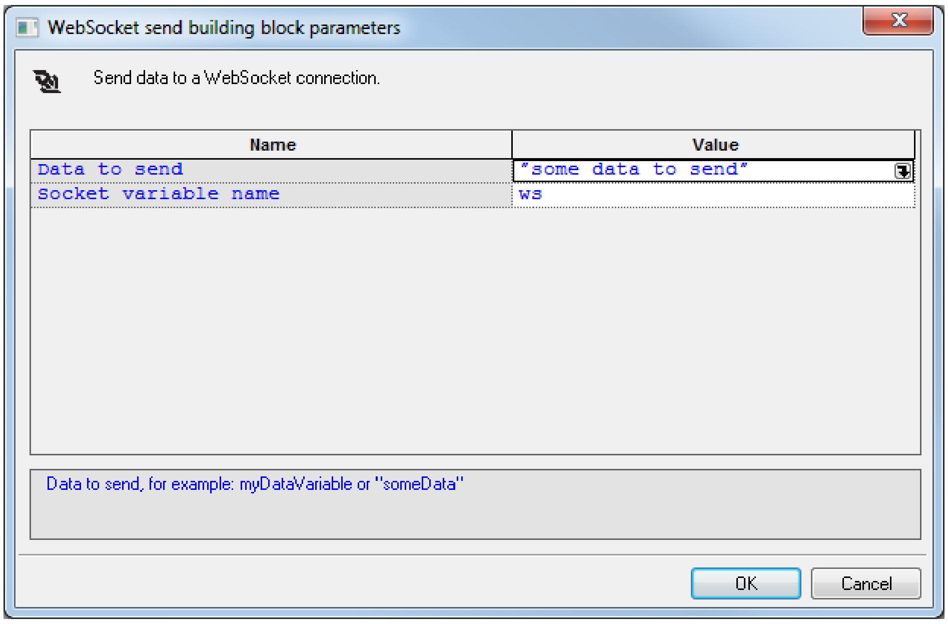
WebSocket Send

Use the WebSocket Send Building Block to send data to a WebSocket connection.

To insert a WebSocket Send Building Block:

1. Drag the **WebSocket Send** icon from the WebSocket toolbox into the Script Tree at the desired location.

   The WebSocket Send Building Block parameters dialog box opens.



WebSocket send building block parameters

Send data to a WebSocket connection.

Name	Value
Data to send	"some data to send"
Socket variable name	ws

Data to send, for example: myDataVariable or "someData"

Figure 200: WebSocket Send Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Data to Send field is used to define the data to be sent.

3. Enter the appropriate field value into the Value column next to the field name, as described in Table 92.

Table 92: WebSocket Send Building Block Parameters Dialog Box Fields

Field Name	Description
Data to send	Specify the data to be sent.
Socket variable name	Specify the name of the WebSocket object to which to send the data.

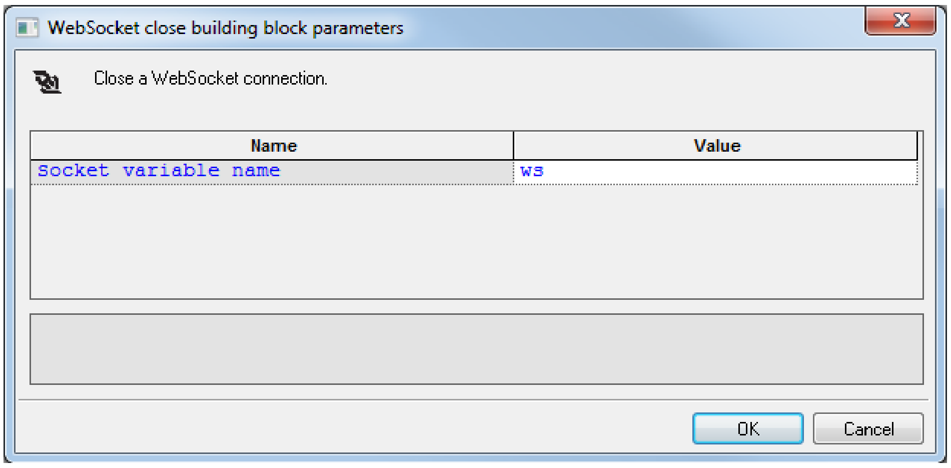
WebSocket Close

Use the WebSocket Close Building Block to close a WebSocket connection.

To insert a WebSocket Close Building Block:

1. Drag the **WebSocket Close** icon from the WebSocket toolbox into the Script Tree at the desired location.

   The WebSocket Close Building Block parameters dialog box opens.



Close a WebSocket connection.

Name	Value
Socket variable name	ws

Figure 201: WebSocket Close Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.
3. Enter the appropriate field value into the Value column next to the field name, as described in Table 93.

Table 93: WebSocket Close Building Block Parameters Dialog Box Fields

Field Name	Description
Socket variable name	Specify the name of the WebSocket connection you wish to close.

The WebLOAD Recorder Web Services Toolbox

Use the WebLOAD Web Services Building Blocks to simply and easily add web service calls or HTTP requests to your test session script.

HTTP

Use the HTTP Building Block to create an HTTP web service call. This building block supports any HTTP method.

To insert an HTTP Web Service Building Block:

1. Drag the **HTTP** icon from the Web Services toolbox into the Script Tree at the desired location.

   The HTTP Web Service dialog box opens.

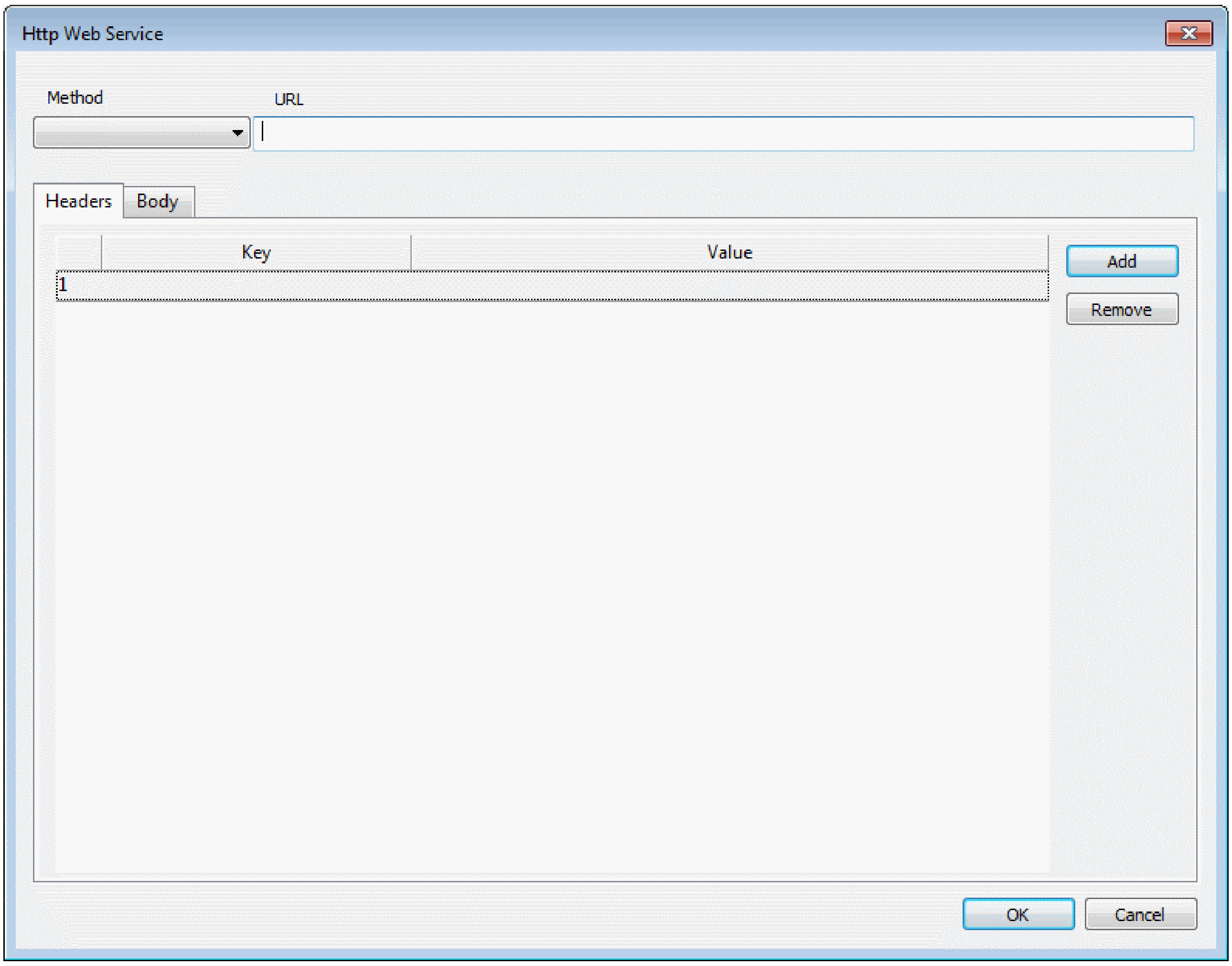
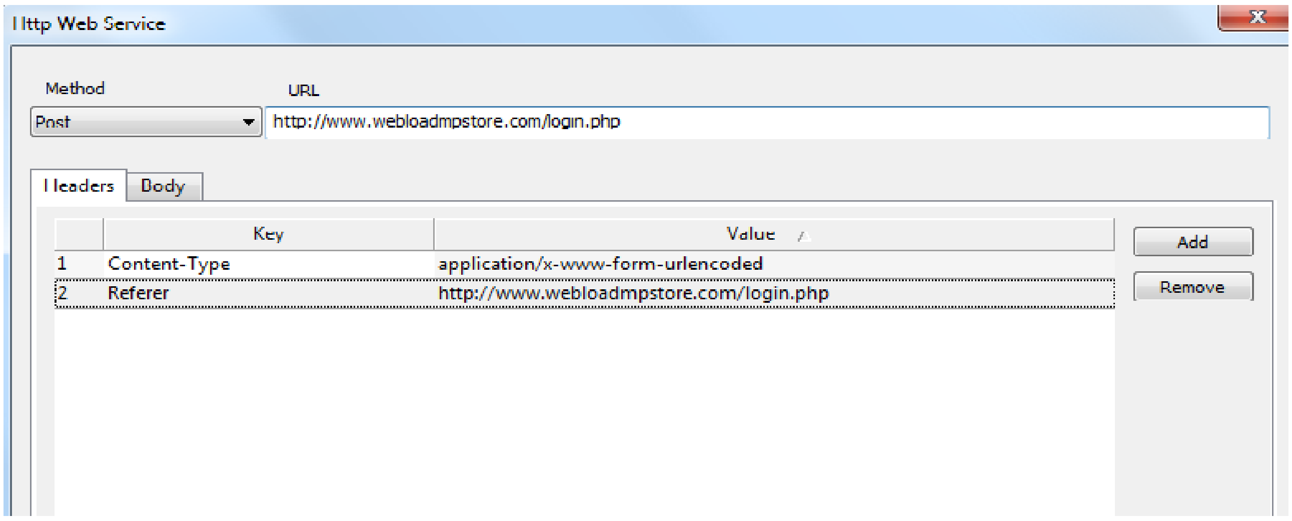


Figure 202: HTTP Web Service Dialog Box

2. Select a method from the **Method** drop-down list: **Get**, **Post**, **Put**, **Delete**, **Copy**, **Head** or **Options**.
3. Enter a URL in the **URL** field.
4. Optionally, use the **Headers** tab to define the web service headers, as follows:
   a. In the **Key** field, select a key from the drop-down list, or enter any key.
   b. In the **Value** field, enter a value for the specified key.
   c. To add additional headers, click **Add**, and repeat steps (a) to (b). To remove a header, select the relevant line and click **Remove**.

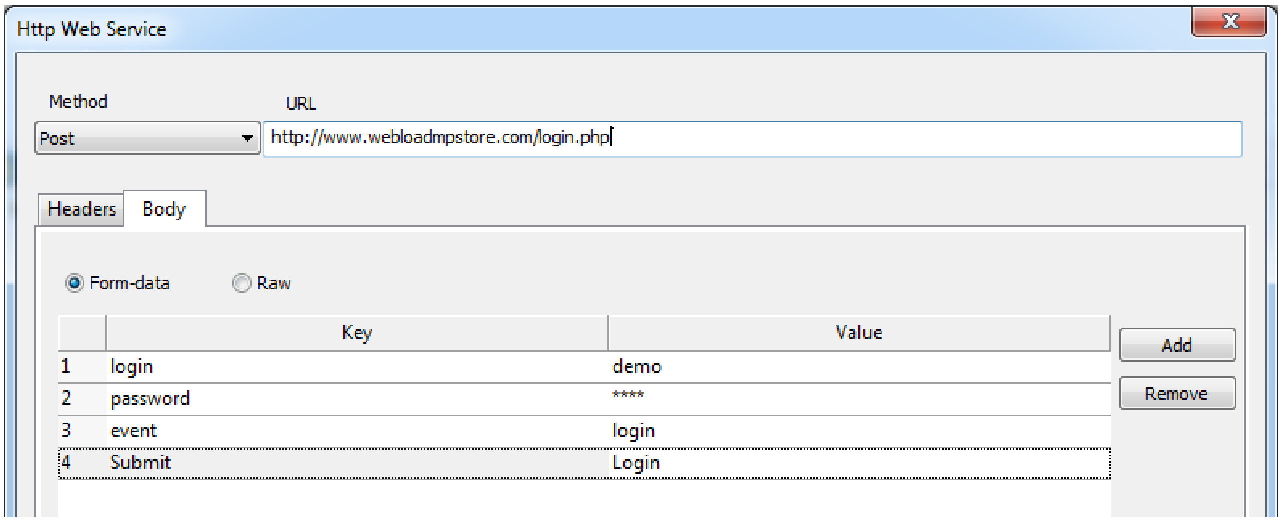


Method: Post
URL: http://www.webloadmpstore.com/login.php

	Key	Value
1	Content-Type	application/x-www-form-urlencoded
2	Referer	http://www.webloadmpstore.com/login.php

Figure 203: Defining Web Service Headers

5. Optionally, use the **Body** tab to define the web service body, using either of the following methods:
   - Select **Form-data**, and define a desired list of Keys and their Values.
     a. In the **Key** field, select a key from the drop-down list, or enter any key.
     b. In the **Value** field, enter a value for the specified key.
     c. To add an additional form data element, click **Add**, and repeat steps (a) to (b). To remove a form data element, select the relevant line and click **Remove**.



Method: Post

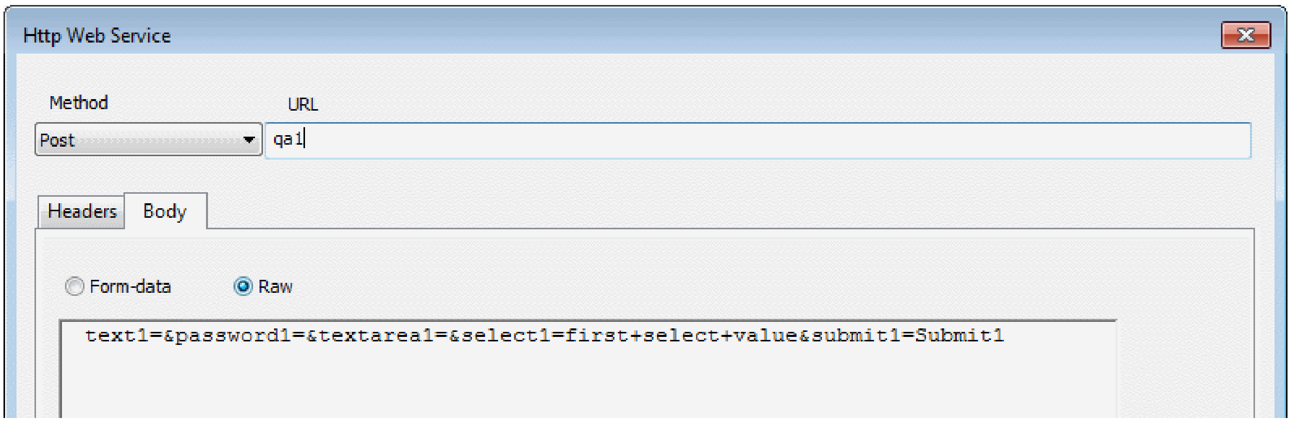
URL: http://www.webloadmpstore.com/login.php

Form-data [✓] Raw

	Key	Value
1	login	demo
2	password	****
3	event	login
4	Submit	Login

Figure 204: Defining Web Service Body using Form-data

- Select **Raw** and enter content into the empty box.



Method: Post

URL: qa1

Form-data Raw [✓]

```
text1=&password1=&textarea1=&select1=first+select+value&submit1=Submit1
```

Figure 205: Defining Web Service Body by Entering Raw Data

6. Click **OK**. The matching code appears in the JavaScript View.

WSDL

Use the WSDL Building Block to create a web service call based on data from a Web Services Description Language (WSDL) file.

To insert a WSDL Building Block:

1. Drag the **WSDL** icon from the Web Services toolbox into the Script Tree at the desired location.

   The WSDL Web Service dialog box opens.

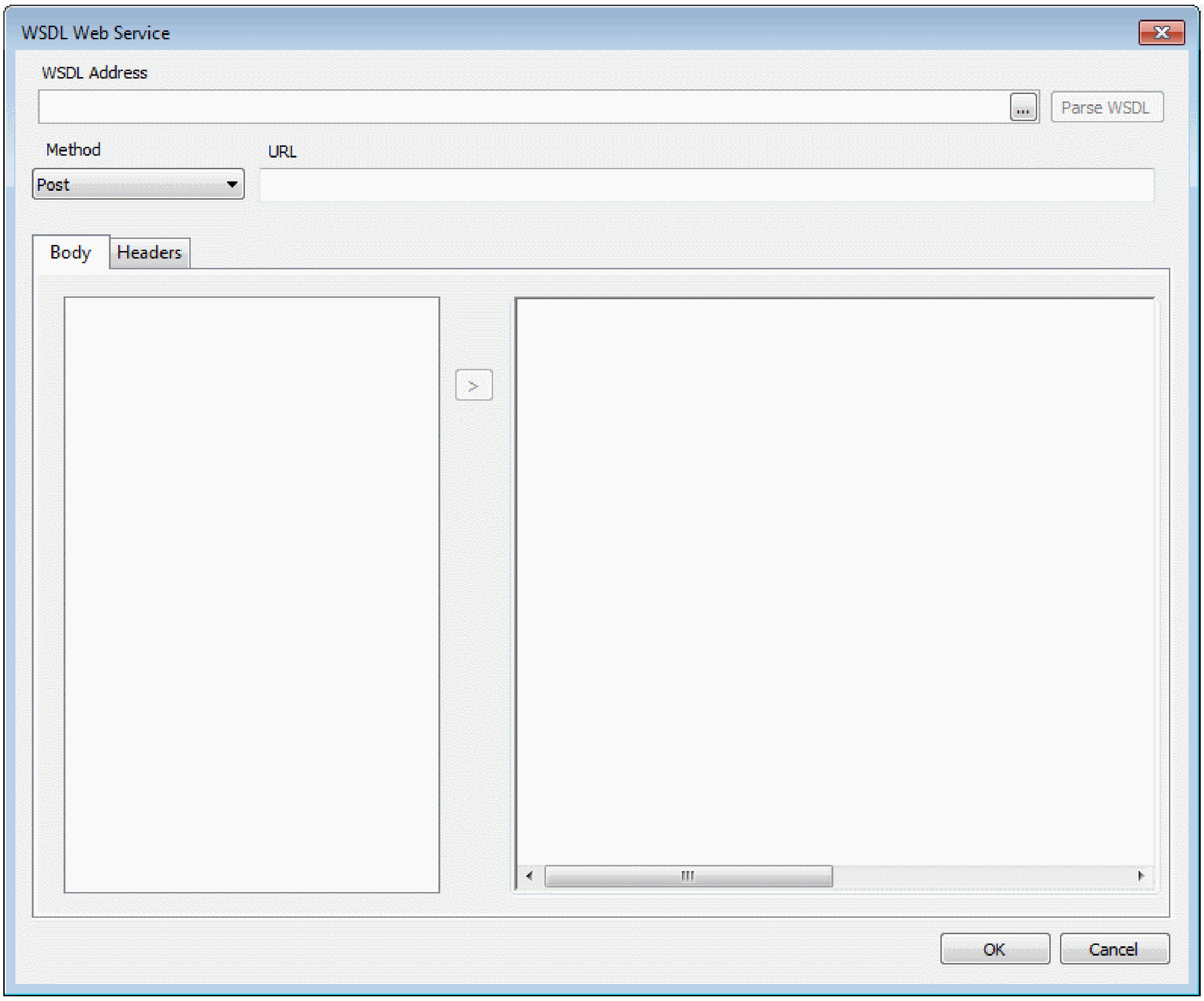
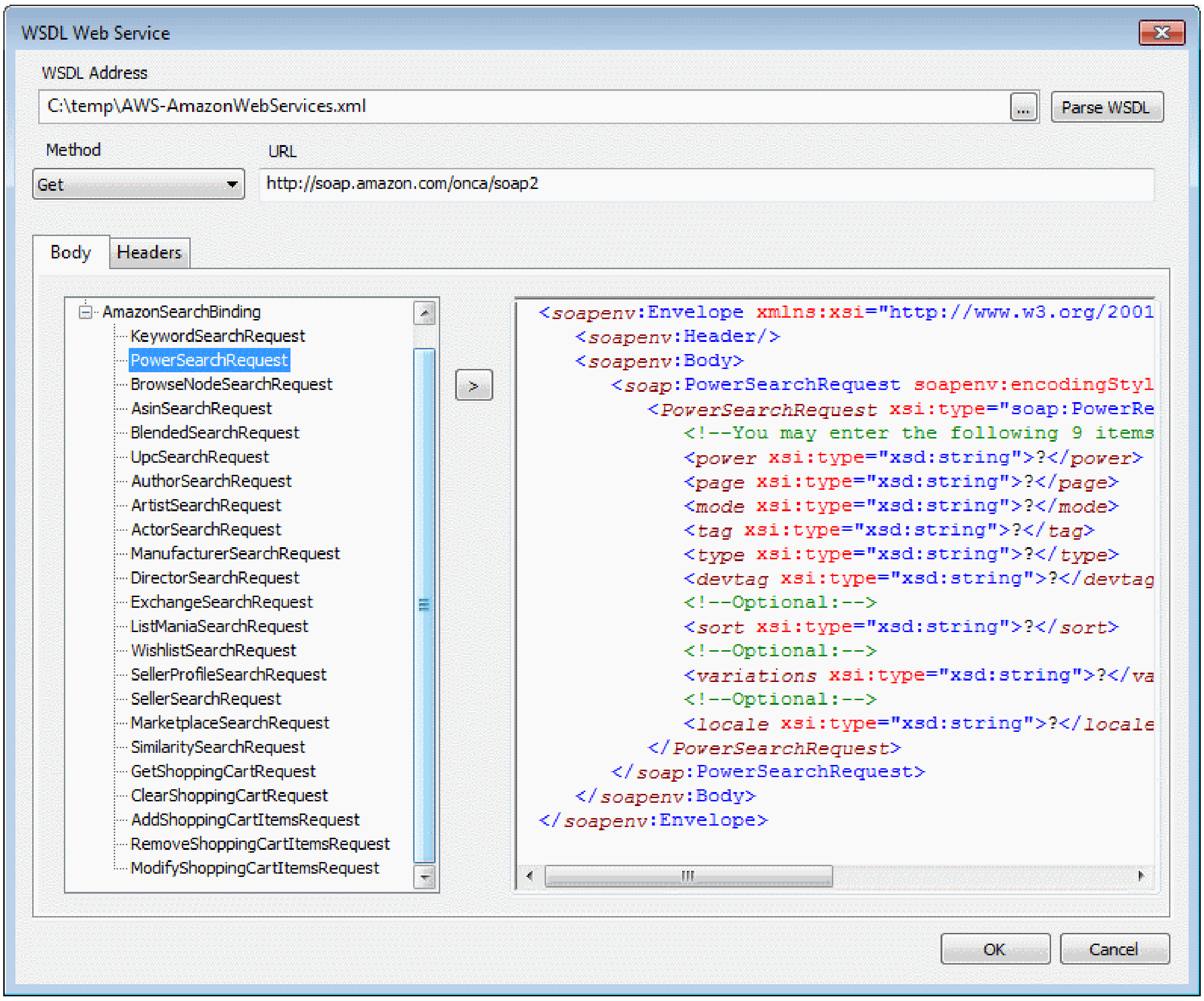


Figure 206: WSDL Web Service Dialog Box

2. In the **WSDL Address** field, enter a WSDL URL, or browse to a WSDL file.
3. Click **Parse WSDL**. The following occurs:
   - The left pane in the **Body** tab is populated with the interfaces, and their methods, of the WSDL file.
4. Define the web service body, as follows:
   a. Select a desired method and click the arrow between the left and right panes.

      The right pane displays the body of the selected method. In addition, WebLOAD automatically creates a default SOAPAction header based on the selected method.



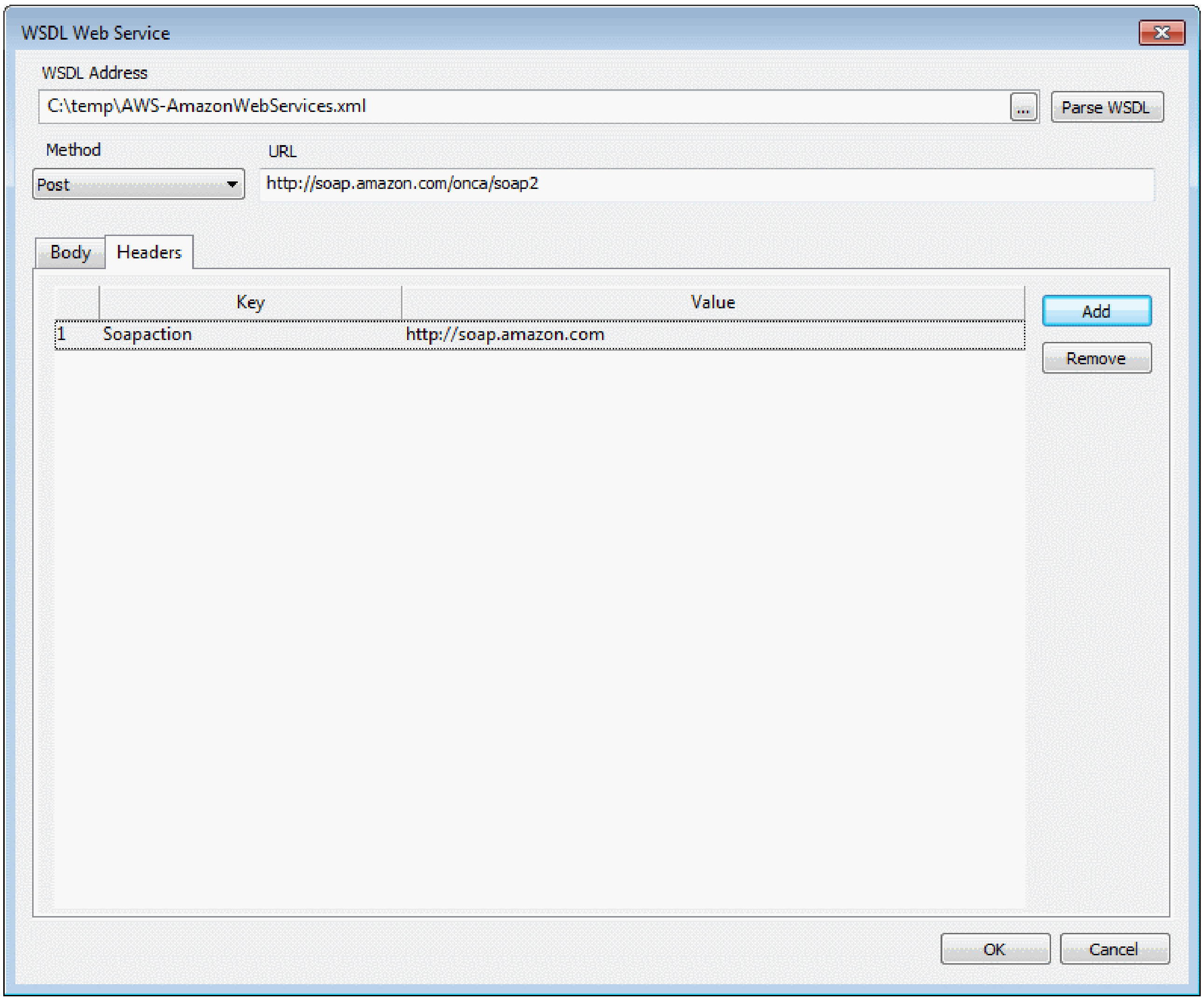
```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001
  <soapenv:Header/>
  <soapenv:Body>
    <soap:PowerSearchRequest soapenv:encodingStyl
      <PowerSearchRequest xsi:type="soap:PowerRe
        <!--You may enter the following 9 items
        <power xsi:type="xsd:string">?</power>
        <page xsi:type="xsd:string">?</page>
        <mode xsi:type="xsd:string">?</mode>
        <tag xsi:type="xsd:string">?</tag>
        <type xsi:type="xsd:string">?</type>
        <devtag xsi:type="xsd:string">?</devtag
        <!--Optional:-->
        <sort xsi:type="xsd:string">?</sort>
        <!--Optional:-->
        <variations xsi:type="xsd:string">?</va
        <!--Optional:-->
        <locale xsi:type="xsd:string">?</locale
      </PowerSearchRequest>
    </soap:PowerSearchRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 207: HTTP Web Service Dialog Box

b. Enter values as desired to replace the "?" in the method body.

5. View and edit the SOAPAction headers as follows:

a. Click the **Headers** tab to view the SOAPAction header that WebLOAD had automatically created when you selected the method.



WSDL Address: C:\temp\AWS-AmazonWebServices.xml

Method: Post

URL: http://soap.amazon.com/onca/soap2

	Key	Value
1	Soapaction	http://soap.amazon.com

Figure 208: HTTP Web Service Dialog Box

b. Optionally edit the header.

c. Optionally add additional headers, as follows:

- Click **Add**. A new empty line is added to the table.
- In the **Key** field, select a key from the drop-down list, or enter any key.
- In the **Value** field, enter a value for the specified key.

d. To remove a header, select the relevant line and click **Remove**.

6. Click **OK**. The matching code appears in the JavaScript View.

The JMS Toolbox

The following table describes the purpose of each of the WebLOAD Recorder JMS Toolbox items:

Table 94: JMS Toolbox Items

Script Item	Purpose
JMS-Connect-JNDI	Connect to a JMS queue, and create consumer and producer objects using JNDI.
JMS-Connect-HornetQ	Connect to a JMS queue, and create consumer and producer objects using HornetQ implementation.
JMS--Send	Send a JMS message using a previously created producer.
JMS-Receive	Receive a JMS message from a previously created consumer.

To use JMS in WebLOAD:

1. Create JMS consumer and producer objects, either through JNDI or using the HornetQ implementation.
2. Use send and/or receive to put or get messages from a queue.

Note: The implementation relies on the WebLOAD Java connection, which enables making changes to the generated code easily. Refer to the "Working with Java" section of the *WebLOAD Scripting Guide* for more details.

JMS-Connect-JNDI

The JMS-Connect-JNDI Building Block creates a JMS connection using JNDI. More specifically, the following objects are used:

- InitialContext – JNDI context
- JMS Connection Factory – looked up using JNDI
- JMS Destination (Queue) – looked up using JNDI
- JMS Session – using the JMS Connection
- JMS Producer and Consumer – using the session and destination

The objects are created in InitClient to be used once per virtual client. If required, some or all of the code can be moved to the script body to repeat the action each round.

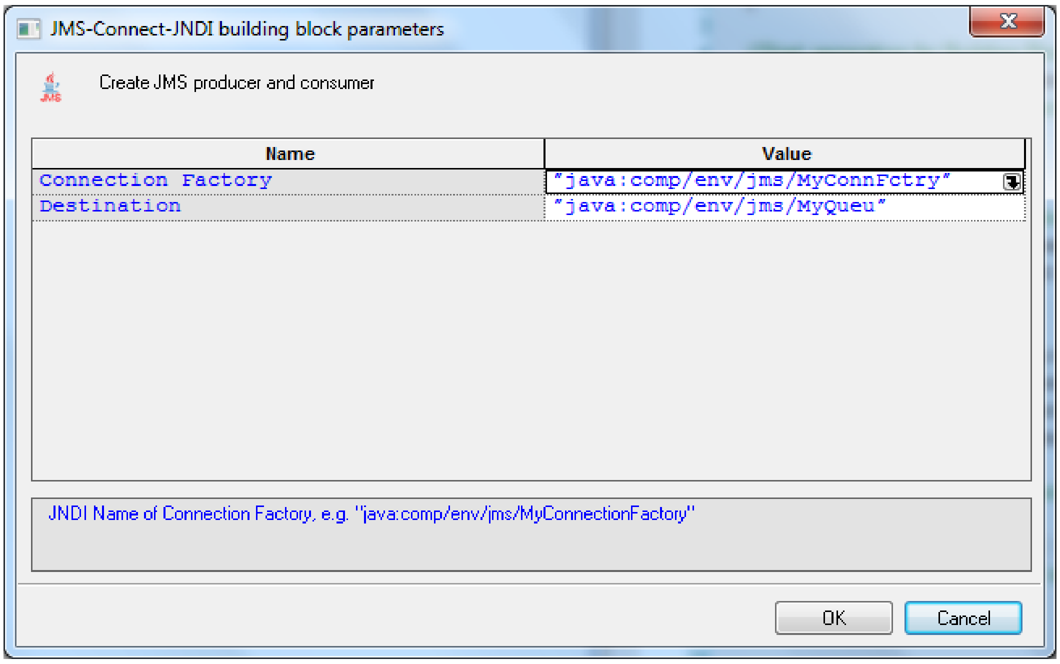
The code can be adjusted as needed, for example by adding properties to the lookup:

```
env = new Packages.java.util.Properties();
env.put("INITIAL_CONTEXT_FACTORY", "my.factory");
env.put("PROVIDER_URL", "http://10.0.1.11");
jndiContext = new Packages.javax.naming.InitialContext(env);
```

To insert a JMS-Connect-JNDI Building Block:

1. Drag the **JMS-Connect-JNDI** icon from the JMS toolbox into the Script Tree.

   The JMS-Connect-JNDI Building Block parameters dialog box opens.



Create JMS producer and consumer

Name	Value
Connection Factory	"java:comp/env/jms/MyConnFctry"
Destination	"java:comp/env/jms/MyQueu"

JNDI Name of Connection Factory, e.g. "java:comp/env/jms/MyConnectionFactory"

Figure 209: JMS-Connect-JNDI Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 95: JMS-Connect-JNDI Building Block Parameters Dialog Box Fields

Field Name	Description
Connection Factory	The JNDI Name of the Connection Factory, for example: `"java:comp/env/jms/MyConnectionFactory"`
Destination	The JNDI name of the destination, , for example: `"java:comp/env/jms/QueueName"`

3. Click **OK**.

   The node is added to the Script Tree. The JavaScript code, including the `InitAgenda()` function, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

   For example:

```
function InitAgenda()
{
```

```
//Start generation for Building Block JMS-Connect-JNDI
IncludeFile("wljms.js",WLExecuteScript);
//End generation for Building Block JMS-Connect-JNDI
}
function InitClient()
{
//Start generation for Building Block JMS-Connect-JNDI
jndiContext = new
Packages.javax.naming.InitialContext();
connectionFactory = jndiContext.lookup("/myConFact");

dest = jndiContext.lookup("/myQueue");
connection = connectionFactory.createConnection();
session = connection.createSession(false,
Packages.javax.jms.Session.AUTO_ACKNOWLEDGE);
producer = session.createProducer(dest);
consumer = session.createConsumer(dest);
connection.start();
//End generation for Building Block JMS-Connect-JNDI
}
/***** WLIDE - JMS-Connect-JNDI - ID:2 *****/
//JMS 'producer' and 'consumer' objects created at
InitClient()

// END WLIDE

function TerminateClient()
{
//Start generation for Building Block JMS-Connect-JNDI
connection.close();
//End generation for Building Block JMS-Connect-JNDI
}
```

JMS-Connect-HornetQ

The JMS-Connect-HornetQ Building Block creates a JMS connection using the HornetQ native implementation.

More specifically, the HornetQJMSClient class is used to create the connection factory and queue, as follows:

- Create JMS Connection Factory using HornetQJMSClient

- Create JMS Queue using HornetQJMSClient
- Create standard JMS objects – consumer, producer, session using the created objects

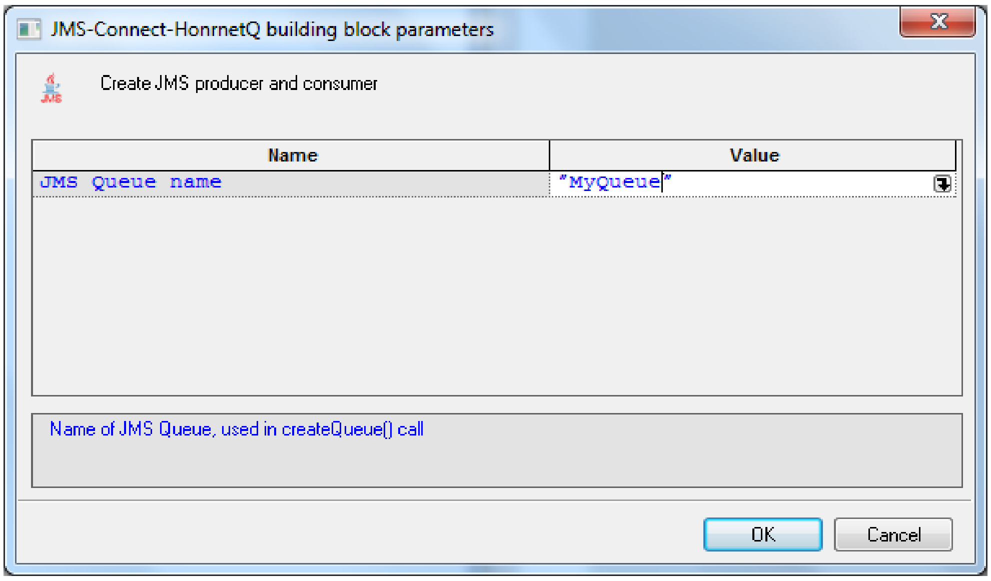
The objects are created in InitClient to be used once per virtual client. If required, some or all of the code can be moved to the script body to repeat the action each round.

Note: HornetQ supports different ways of creating the connection. The code can be adjusted to support methods other that the method described here..

To insert a JMS-Connect-HornetQ Building Block:

1. Drag the **JMS-Connect-HornetQ** icon from the JMS toolbox into the Script Tree.

   The JMS-Connect- HornetQ Building Block parameters dialog box opens.


JMS-Connect-HonrnetQ building block parameters
Create JMS producer and consumer
OK
Cancel

Name	Value
JMS Queue name	"MyQueue"

Name of JMS Queue, used in createQueue() call

Figure 210: JMS-Connect-HornetQ Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 96: JMS-Connect-HornetQ Building Block Parameters Dialog Box Fields

Field Name	Description
JMS Queue name	The name of the JMS Queue, used in createQueue() calls.

3. Click **OK**.

   The node is added to the Script Tree. The JavaScript code, including the `InitAgenda()` function, is added to the script. To see the new JavaScript code, view the script in JavaScript Editing mode.

For example:

```
function InitAgenda()
{

//Start generation for Building Block JMS-Connect-
HonrnetQ
IncludeFile("wljms.js",WLExecuteScript);
//End generation for Building Block JMS-Connect-HonrnetQ
}
function InitClient()
{
//Start generation for Building Block JMS-Connect-
HonrnetQ
transportConfiguration =
    new
Packages.org.hornetq.api.core.TransportConfiguration

("org.hornetq.core.remoting.impl.netty.NettyConnectorFac
tory");
connectionFactory =
Packages.org.hornetq.api.jms.HornetQJMSClient.createConn
ectionFactoryWithoutHA
        (Packages.org.hornetq.api.jms.JMSFactoryType.CF,
[transportConfiguration]);


connection = connectionFactory.createConnection();
session = connection.createSession(false,
Packages.javax.jms.Session.AUTO_ACKNOWLEDGE);
queue =
Packages.org.hornetq.api.jms.HornetQJMSClient.createQueu
e("MyQueue");
producer = session.createProducer(queue);
consumer = session.createConsumer(queue);
connection.start();
//End generation for Building Block JMS-Connect-HonrnetQ
}
/***** WLIDE - JMS-Connect-HonrnetQ - ID:2 *****/
//JMS 'producer' and 'consumer' objects created at
InitClient()


// END WLIDE


function TerminateClient()
{
```

```
//Start generation for Building Block JMS-Connect-
HonrnetQ
connection.close();
//End generation for Building Block JMS-Connect-HonrnetQ
}
```

JMS-Send

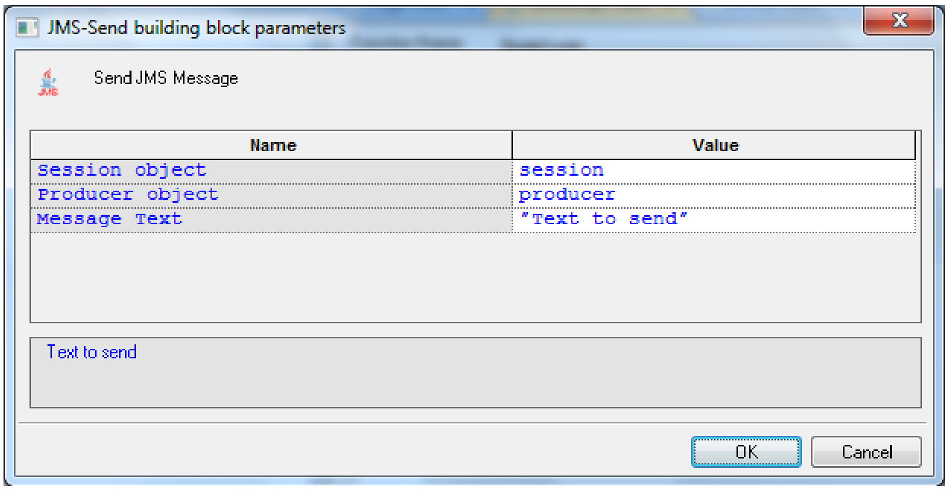
The JMS-Send Building Block sends a JMS message to the specified 'producer' (javax.jms.MessageProducer), using a given 'session' (javax.jms.Session).

The 'producer' and 'session' objects are expected to already have been created, for example by using one of the JMS-Connect building blocks.

To insert a JMS-Send Building Block:

1. Drag the **JMS-Send** icon from the JMS toolbox into the Script Tree.

   The JMS-Send Building Block parameters dialog box opens.



Name	Value
Session object	session
Producer object	producer
Message Text	"Text to send"

Text to send

Figure 211: JMS-Send Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 97: JMS-Send Building Block Parameters Dialog Box Fields

Field Name	Description
Session object	The JMS Session object, created by `connection.createSession()`
Producer object	The JMS Producer object, created by `session.createProducer()`
Message Text	The text to send

3. Click **OK**.

   The node is added to the Script Tree.

   For example:

```
sendJmsMessage(session, producer, "Text to send");
```

JMS-Receive

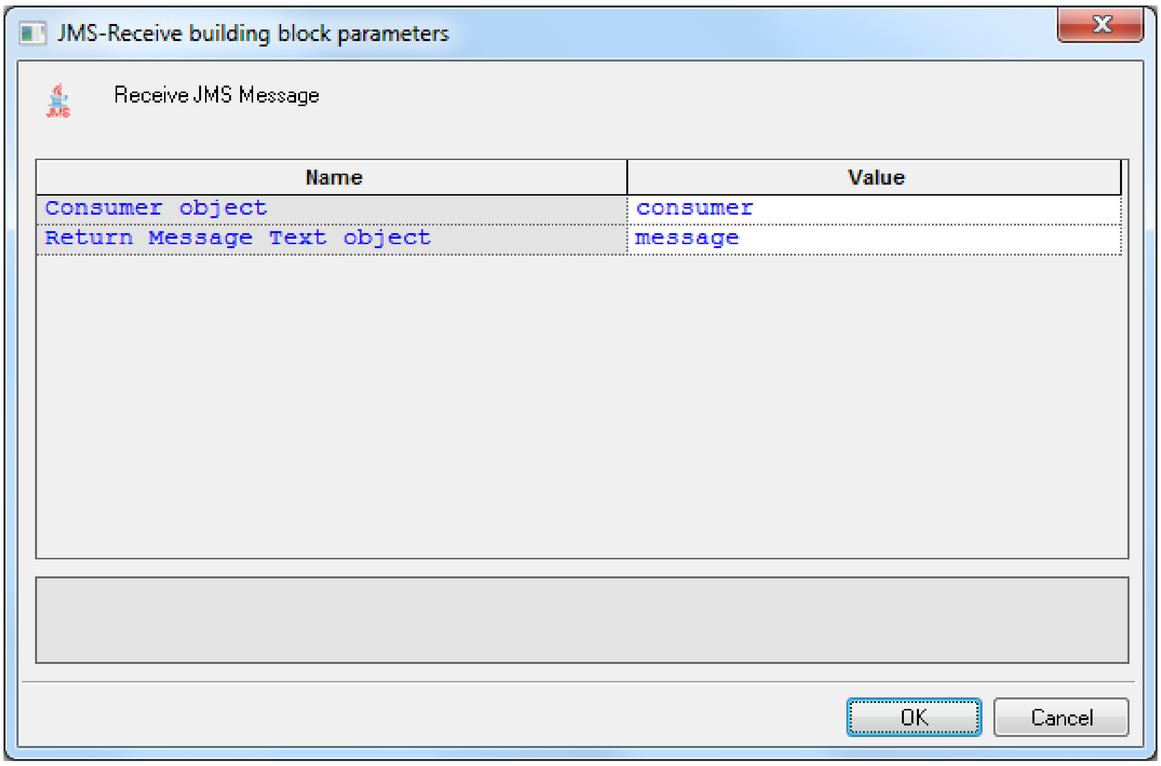
The JMS-Receive Building Block receives a JMS message from the specified 'consumer' (javax.jms.MessageConsumer), into a given variable. The variable can then be used to inspect the message context, using standard JavaScript String methods.

The 'consumer' object is expected to already have been created, for example by using one of the JMS-Connect building blocks.

To insert a JMS-Receive Building Block:

1. Drag the **JMS-Receive** icon from the JMS toolbox into the Script Tree.

   The JMS-Receive Building Block parameters dialog box opens.



Name	Value
Consumer object	consumer
Return Message Text object	message

Figure 212: JMS-Receive Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 98: JMS-Send Building Block Parameters Dialog Box Fields

Field Name	Description
Consumer object	The JMS Consumer object, create by session.createConsumer().
Return Message Text object	The return value name.

3. Click **OK**.

   The node is added to the Script Tree.

   For example:

```
var message = receiveJmsMessage(consumer);

InfoMessage("Got message: " + message);
```

The Real Clients Toolbox

The Real Clients Toolbox includes Selenium, Perfecto Mobile, and JUnit Runner building blocks.

Selenium Building Blocks

Use the WebLOAD Selenium Building Blocks to simply and easily create Selenium actions. For a full description of how to use Selenium scripts in WebLOAD, refer to *Selenium Integration* (on page 389), and to the *Working with Java Selenium Scripts* section in *Chapter 3 Advanced JavaScript Script Features* of the *WebLOAD Scripting Guide*.

To call Selenium actions using the Selenium building blocks:

1. Use the **Selenium driver** building block to instruct WebLOAD to create a Selenium Java WebDriver (org.openqa.selenium.WebDriver).
2. Where desired, enter commands to be executed by the Selenium Java WebDriver. For information about using the WebDriver, refer to the Selenium documentation (http://docs.seleniumhq.org/docs/03_webdriver.jsp#selenium-webdriver-api-commands-and-operations).
3. Where desired, use the **Selenium Report Statistics** building block to insert a command that instructs WebLOAD to collect Selenium statistics from the current page.

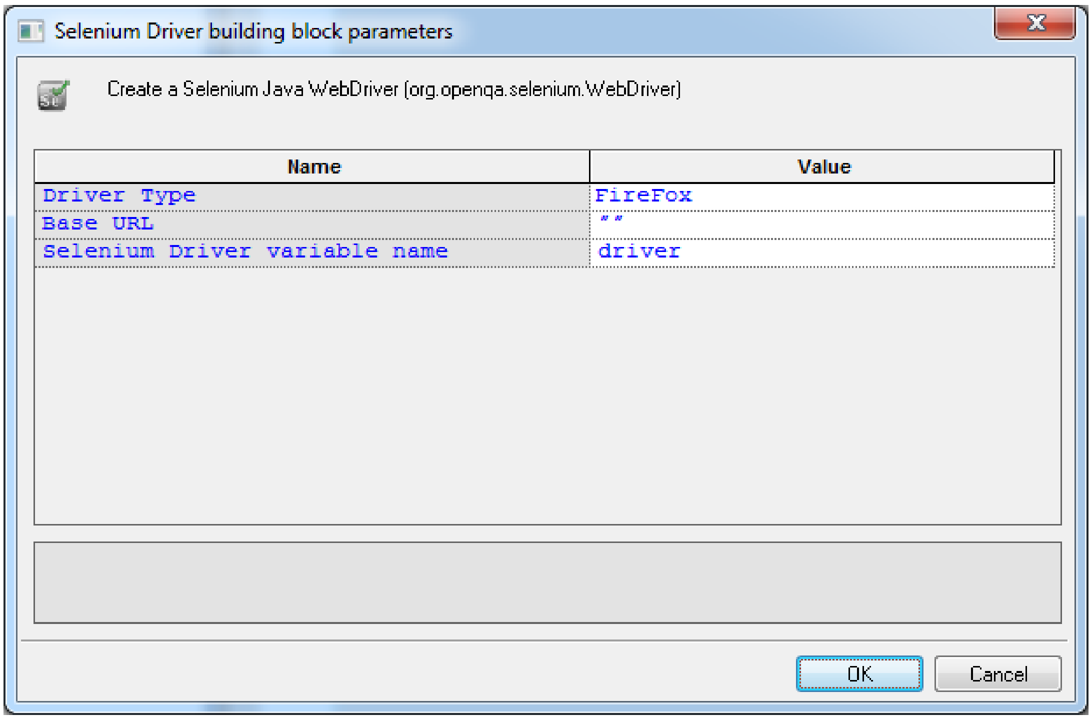
Selenium driver

Use the Selenium Driver Building Block to instruct WebLOAD to create a Selenium Java WebDriver (org.openqa.selenium.WebDriver).

To insert a Selenium Driver Building Block:

1. Drag the **Selenium Driver** icon from the Real Clients toolbox into the Script Tree at the desired location.

   The Selenium Driver Building Block parameters dialog box opens.



Selenium Driver building block parameters

Create a Selenium Java WebDriver (org.openqa.selenium.WebDriver)

Name	Value
Driver Type	FireFox
Base URL	""
Selenium Driver variable name	driver

Figure 213: Selenium Driver Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Driver Type field is used to specify which Selenium driver type to use.

3. Enter the appropriate field value into the Value column next to the field name, as described in Table 99.

Table 99: Selenium Driver Building Block Parameters Dialog Box Fields

Field Name	Description
Driver Type	Specify what type of Selenium driver to create. The possible values are: • Firefox • Chrome • Internet Explorer
Base URL	Specify the first URL to which the driver will connect.
Selenium Driver variable name	Specify a name for this WebLOAD object.

4. Click **OK**.

   The node is added to the Script Tree.

Selenium Report Statistics

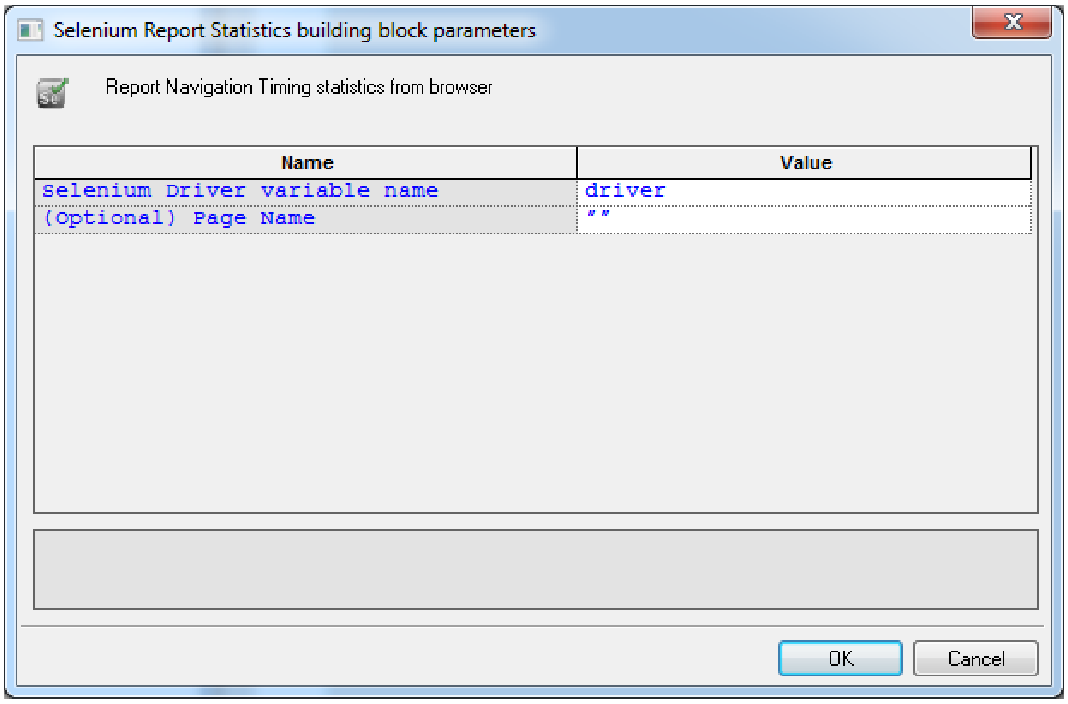
Use the Selenium Report Statistics Block to instruct WebLOAD to collect and report navigation timing statistics for the current page.

For a full list and description of the navigation timing statistics, see Table 104.

To insert a Selenium Report Statistics Building Block:

1. Drag the **Selenium Report Statistics** icon from the Real Clients toolbox into the Script Tree at the desired location.

   The Selenium Report Statistics Building Block parameters dialog box opens.


Selenium Report Statistics building block parameters
Report Navigation Timing statistics from browser
OK
Cancel

Name	Value
Selenium Driver variable name	driver
(Optional) Page Name	""

Figure 214: Selenium Report Statistics Building Block Parameters Dialog Box

2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   For example, in the preceding figure, the comment area explains that the Selenium Driver field is used to specify the Selenium Driver object.

3. Enter the appropriate field value into the Value column next to the field name, as described in Table 100.

Table 100: Selenium Report Statistics Building Block Parameters Dialog Box Fields

Field Name	Description
Selenium Driver	Specify the name of the Selenium Driver object. Navigation timing statistics will be collected for actions executed by this Selenium Driver.
Page Name	By default, WebLOAD reports navigation timing statistics as an average across all pages accessed by the Selenium Driver. If you wish to also display separately the statistics for a particular page, enter the page name in this parameter. The resultant measurements will be called <Measurement Name>-<Page Name>. This parameter is optional.

4. Click **OK**.

   The node is added to the Script Tree.

For example, if you enter **driver** as the name of the Selenium driver, and **my Page name** as the name of a desired page, the following is added to the Script Tree:

```
reportStatistics(driver, "my Page name");
```

JUnit Runner Building Block for running JUnit and other unit testing frameworks

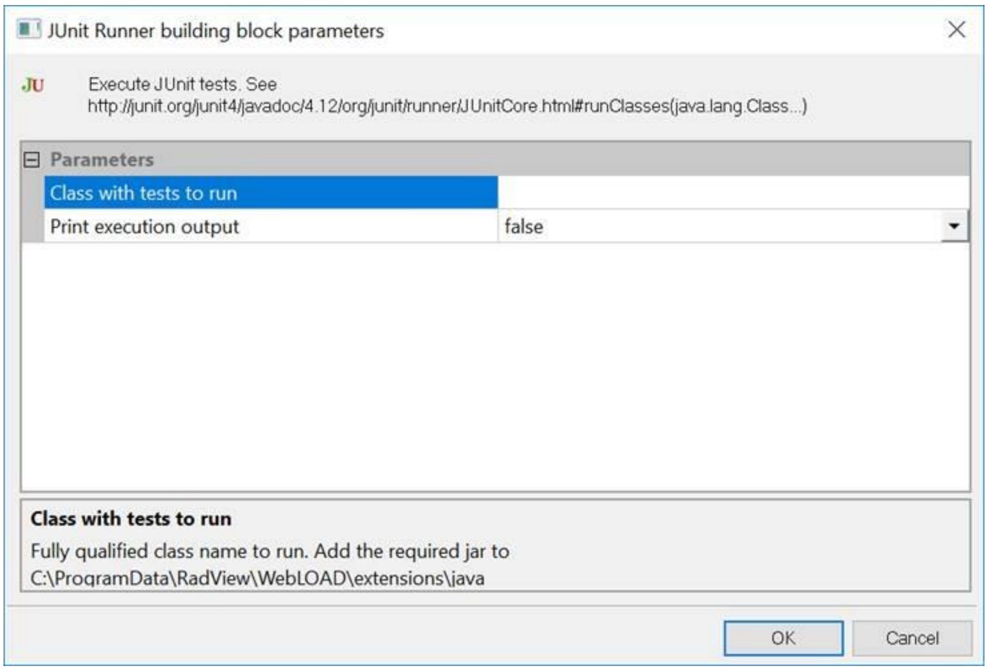
Use the JUnit Runner building block to run all unit tests of specified classes, as part of the load.

This building block is relevant not only for JUnit but for unit tests of any kind inserted in the script. For example, if your script includes Cucumber unit tests, then adding the JUnit Runner building block will run the Cucumber unit tests.

To insert a JUnit Runner Building Block:

1. Drag the **JUnit Runner** icon from the JMS toolbox into the Script Tree.

   The JUnit Runner Building Block parameters dialog box opens.



Execute JUnit tests. See http://junit.org/junit4/javadoc/4.12/org/junit/runner/JUnitCore.html#runClasses(java.lang.Class...)

Parameters

Class with tests to run	
Print execution output	false

Class with tests to run
Fully qualified class name to run. Add the required jar to C:\ProgramData\RadView\WebLOAD\extensions\java

Figure 215: JUnit Runner Building Block Parameters Dialog Box

2. Edit the dialog box fields according to the following table.

Table 101: JUnit Runner Building Block Parameters Dialog Box Fields

Field Name	Description
Class with tests to run	A comma-separated list of the classes on which to run the unit tests.
Print execution output	Whether to display in the WebLOAD Recorder and WebLOAD Console log window, the unit testing execution results.

3. Click **OK**.

   The node is added to the Script Tree.

Perfecto Mobile Building Blocks

Use the Perfecto Mobile building block to instruct WebLOAD to create a Perfecto Mobile script. For a full description of how to create and use Perfecto Mobile scripts in WebLOAD, refer to *Integrating with Perfecto* Mobile (on page 383).

Appendix B

WebLOAD Recorder File Types

The following is a list of files associated with a WebLOAD Recorder project.

Table 102: WebLOAD Recorder Project Files

WebLOAD Recorder Extension	WebLOAD Recorder File Type
`.WLP` Files	WebLOAD Recorder Project Files
`.WLS` Files	WebLOAD Recorder Session Files
`.WLA` Files	Actual Repository Files
`.WLE` Files	Expected Repository Files
`.LOG` Files	Saved Log Window Files

Appendix C

Launching WebLOAD Recorder Testing through the Command Line Interface

This section provides instructions and examples for using Command Line Interface (CLI) to launch WebLOAD Recorder testing.

Running WebLOAD Recorder Testing through the CLI

You can also initiate WebLOAD Recorder testing directly through the CLI. You can enter the WebLOAD Recorder launch command into a batch file or into an external script and WebLOAD Recorder will run directly, without user intervention, using the parameters specified.

To run WebLOAD Recorder testing through the CLI:

Enter the `webloadIDE.exe` command together with a series of optional parameters (described below) into your external script to automatically launch a WebLOAD Recorder test. When your script runs, the executable file will invoke WebLOAD Recorder and run the specified test according to the specified parameters.

Syntax

Use the following syntax to define the parameters for running a WebLOAD Recorder test through a Command Line Interface:

```
webloadide.exe [<flags>][<project or session name to open>]
[<session name to save to>][<Number of rounds to run>]
```

To run more than one session, append all relevant parameters at the end of the syntax. See examples 2 and 3 in *Examples* (on page 370).

Parameters

When running a test invoked by the executable, you can specify the following parameters:

Table 103: Parameters for Test Invoked by the Executable

Parameter	Description
Flags	/a - auto run Automatically run the WebLOAD Recorder test without waiting for user input. If this flag is not specified, WebLOAD Recorder is opened with the specified project / session but the test is not automatically run. The system waits for user input.
Project or session name to open	The name of the `.wlp` file or `.wls` file (Project file or Session file) to run.
Session name to save to	The name of the `.wls` file containing the test data. This file will be saved in the current directory unless otherwise specified.
Number of rounds to run	The number of iterations to run during runtime. The default value is 1.

Parameters are all optional. If no parameters are entered, the executable launches WebLOAD Recorder and does not run a test. If the autorun flag </a> flag is not set, the < Session name to save to >, and the < Number of rounds to run > parameters are ignored.

Examples

Example 1:

```
webloadide.exe test1.wlp
```

This command opens WebLOAD Recorder with the `test1` project file and waits for user input.

Example 2:

```
webloadide.exe /a test1.wlp test2.wlp 3
```

This command:

- Opens WebLOAD Recorder and automatically runs a test using the `test1.wlp` project file.

- Runs the project for three iterations.
- Saves the test results in the WebLOAD Recorder session file `test1.wls`, which includes all of the test data and results.

Example 3:

```
webloadide.exe /a test1.wlp test1.wls 3 /a test2.wlp test2.wls 2
```

This command:

- Opens WebLOAD Recorder and automatically runs a test using the `test1.wlp` project file.
- Runs the project test1.wlp for three iterations.
- Saves the test results in the WebLOAD Recorder session file `test1.wls`, which includes all of the test data and results.
- Opens the WebLOAD Recorder project file `test2.wlp`.
- Runs the project test2.wlp for two iterations.
- Saves the test results in the WebLOAD Recorder session file `test2.wls`, which includes all of the test data and results.

Appendix D

Converting Certificate Files

WebLOAD Recorder supports the use of SSL Client Certificates. WebLOAD Recorder requires that the certificate file be in `*.pem` format. If the certificate file is in `*.pfx` or `*.p12` format, use the Certificate Conversion Wizard application to convert the file to `*.pem` format.

Note: You can use your web browser to export certificates to *.pfx or *.p12 format.

To convert certificate files:

1. Select **Start** > **Programs** > **RadView** > **WebLOAD** > **Utilities** > **Certificate Conversion Wizard**. The Certificate Conversion Wizard appears.

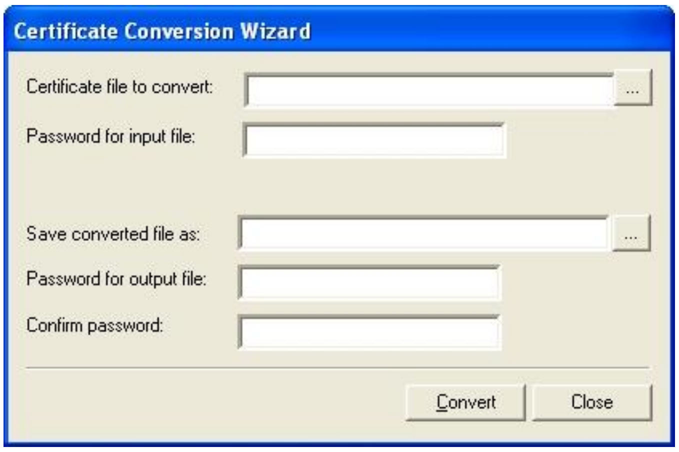

Certificate Conversion Wizard
Certificate file to convert:
...
Password for input file:
Save converted file as:
...
Password for output file:
Confirm password:
Convert
Close

Figure 216: Certificate Conversion Wizard

2. In the Certificate file to convert field, enter the path and file name of a certificate file to convert.

   -Or

   Click ... and browse to the file.

3. In the Password for input file field, enter the password for the certificate file.

Note: If you do not know the password, contact your IT manager.

4. In the Save converted file as field, enter a path and file name for the converted certificate file.

   -Or

   Click ... to open a standard Windows® Save As window.

5. In the Password for output file field, enter a password for the converted certificate.

Note: It is recommended that you use the same password as the one used for the original certificate file.

6. In the Confirm password field, enter the password that you entered in the Password for output file field.

7. Click **Convert**. The file is converted.

Appendix E

Recording Mobile Applications

WebLOAD enables recording mobile applications in two ways:

- Native Mobile Recording – recording traffic from an actual mobile device. This works for both mobile web applications and native mobile applications.
- Simulate mobile in browser – records mobile web applications in the desktop by simulating a mobile browser. Does not require a mobile device, but only works for mobile web applications.

In addition, you can playback a recording as if it were a recording from a mobile application. This is performed in the WebLOAD Console using the Current Project Options window. In the **Browser Type** tab, specify the mobile application in the **Browser Parameters** section. The browser type you select overrides the settings that were defined during the recording, and plays back the recording according to the new settings.

Native Mobile Recording

To record traffic from an actual mobile device:

1. Start native mobile recording, as follows:
   a. In WebLOAD Recorder, click **Start** in the **Home** tab of the ribbon. The Recording dialog appears.

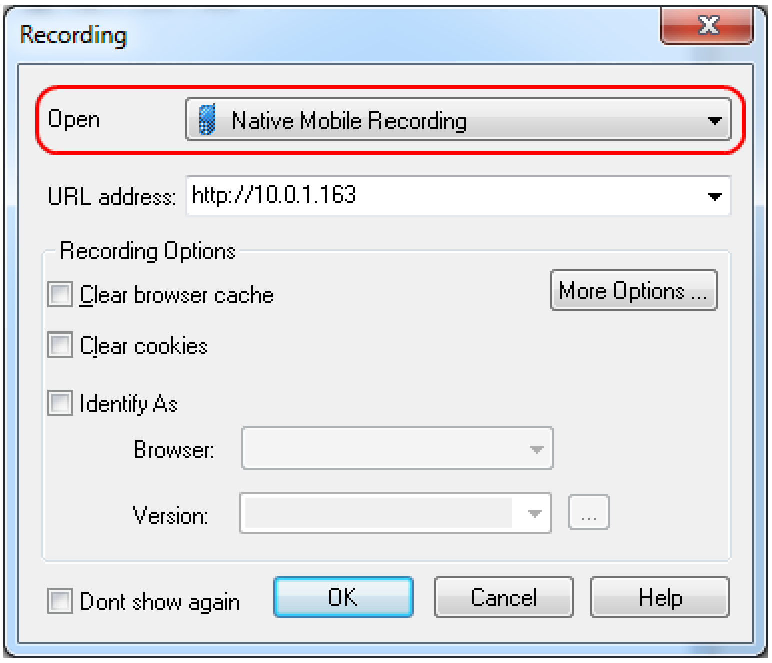


Figure 217: Selecting Native Mobile Recording

   b. In the **Open** drop-down list, select **Native Mobile Recording**.

   c. Click **OK**.

2. Set up the mobile device proxy, as follows:

   a. Connect the mobile device to a Wireless network that can access your WebLOAD machine.

   b. Configure the device's wireless proxy settings to go through the WebLOAD machine, on the noted port.
   This step depends on the device OS type and version. For example, see *Setting Proxy Settings in iPhone* (on page 377) and *Setting Proxy Settings in Android* (on page 380).

3. Perform actions in your mobile device application or web browser; the HTTP requests will be recorded in the test script.

4. Stop recording, as follows:

   a. In WebLOAD Recorder, click **Stop Recording**.

   b. In the mobile device, change the Proxy settings back to off.

Note: You may need to open you firewall to accept connections to proxynator.exe, on port 9884.

Note: To record secure traffic (HTTPS), the root certificate needs to be imported to the phone. For example, see *Recording HTTPS Traffic on iPhone* (on page 379) and *Recording HTTPS traffic on Android (4.0 and above)* (on page 380).

Note: When not recording, the mobile device will not be able to access the network (because the proxy is not available) – to use the network normally, revert the **HTTP Proxy** setting back to **Off**.

Setting Proxy Settings in iPhone

To set iPhone proxy settings:

1. Open **Settings**, and access the **Wi-Fi** network settings:

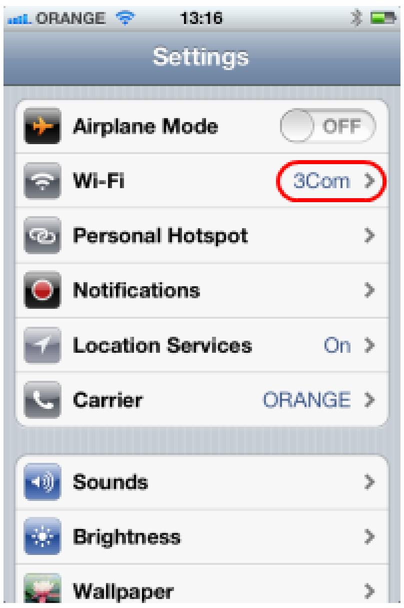


Figure 218: Accessing Wi-Fi iPhone Settings

2. Select edit current Wi-Fi Network settings:

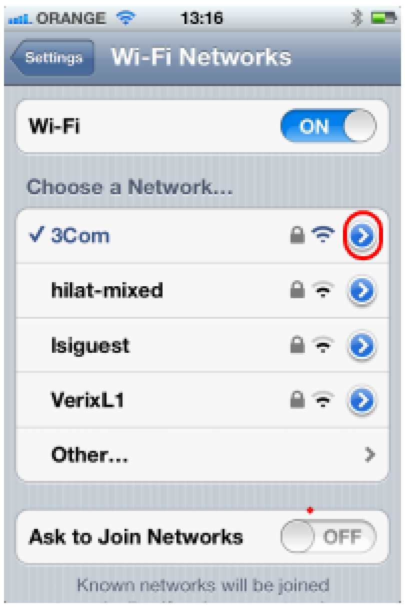
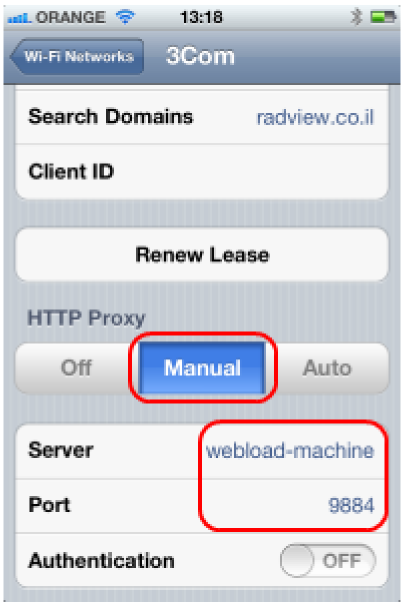


Figure 219: Edit Current Wi-Fi iPhone Settings

3. Scroll down the **HTTP Proxy** section. Change proxy to **Manual** and set the **Server** and **Port** to point to the WebLOAD machines.
   The default port for the proxy-recorder is 9884. You may need to use the machine's IP-address instead of name.



Search Domains radview.co.il

Client ID

HTTP Proxy: Off [✓ Manual] Auto

Server webload-machine

Port 9884

Authentication OFF

Figure 220: Setting Wi-Fi iPhone Settings

Recording HTTPS Traffic on iPhone

In order to record HTTPS traffic, the WebLOAD root certificate needs to be trusted by the phone. To import the root certificate:

1. Locate the root certificate, in:
   c:\Program Files\RadView\WebLOAD\bin\Certificates\root.pem
2. Open the root.pem file on the phone. This can be done by sending the file via e-mail, or accessing the file from a web-server.
3. Click **Install**.

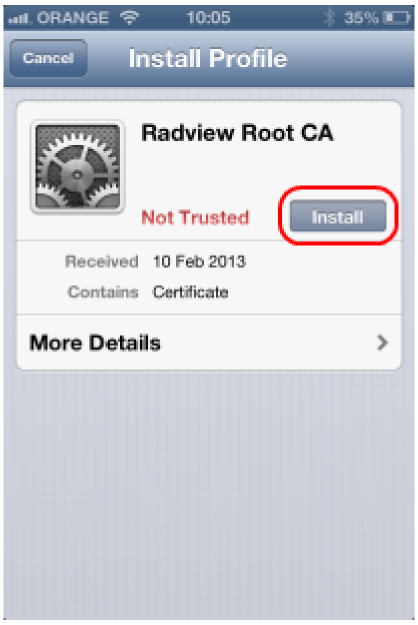
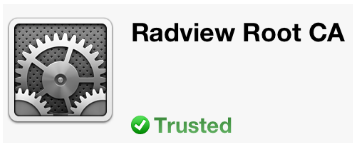


Figure 221: Installing the Root CA

4. A warning will appear; click **Install**.

   The certificate should now be trusted

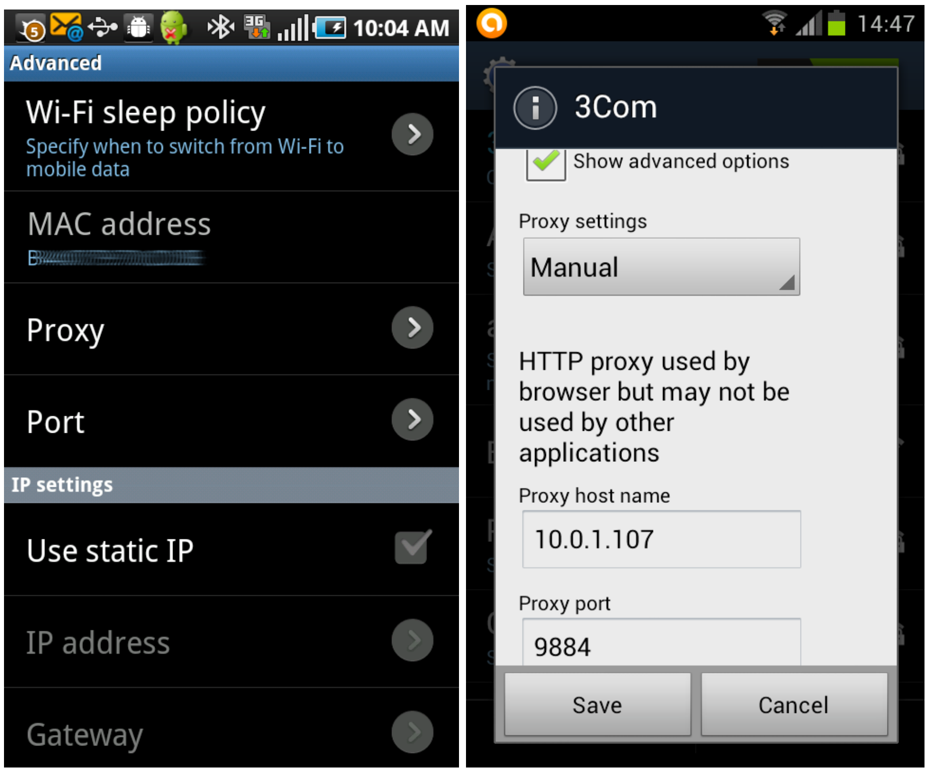


5. After recording is completed, the certificate may be removed. To remove the certificate, select the following:
   Settings > **General** > **Profile 'RadView Root CA'** > **Remove**.

Setting Proxy Settings in Android

To set Android proxy settings:

1. Using the menu button, select **Settings**.
2. Select **Wireless & networks**.
3. Select **Wi-Fi settings**.
4. Switch on and connect to your designated Wi-Fi network.
5. Once connected, press the Menu button again and select **Advanced**.
6. Set **Proxy** to the WebLOAD machine IP-address, and **Port** to 9884.


10:04 AM
Advanced
Wi-Fi sleep policy
Specify when to switch from Wi-Fi to mobile data
MAC address
[redacted]
Proxy
Port
IP settings
Use static IP
IP address
Gateway

14:47
3Com
Show advanced options
Proxy settings
Manual
HTTP proxy used by browser but may not be used by other applications
Proxy host name
10.0.1.107
Proxy port
9884
Save
Cancel

Figure 222: Proxy Settings in Android – Two Examples

Recording HTTPS traffic on Android (4.0 and above)

In order to record HTTPS traffic, the WebLOAD root certificate needs to be trusted by the phone. To import the root certificate:

1. Locate the root certificate, in:
   c:\Program Files\RadView\WebLOAD\bin\Certificates\root.pem
2. Copy locally, as root.crt.

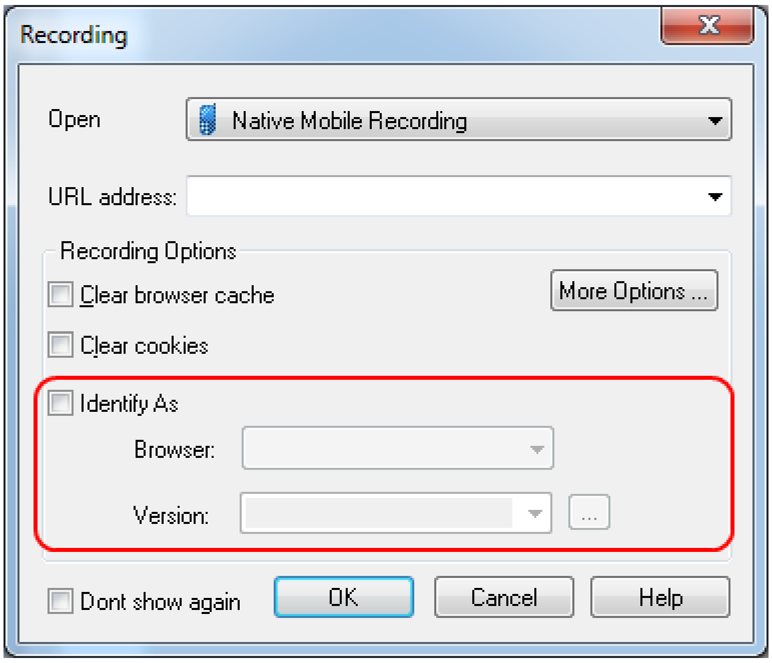
3. Copy root.crt from your computer to the root of your device's internal storage (that is, not in a folder).
4. From a Home or All Apps screen, tap the **Settings** icon.
5. Go to **Personal** > **Security** > **Credential storage** > **Install from storage**.

Simulating a Mobile in a Browser

In Simulate mode, the recording is done using the desktop browser, identified to the server as a mobile user agent.

To simulate a mobile:

1. In WebLOAD Recorder, click **Start Recording**.



Recording

Open: Native Mobile Recording

URL address:

Recording Options

- [ ] Clear browser cache
- [ ] Clear cookies
- [ ] Identify As

Browser:

Version:

- [ ] Dont show again

Figure 223: Simulating a Mobile

2. Check the **Identify As** checkbox.
3. Select the **Browser** family and **Version**.
4. Click **OK**.

Note: This approach is only applicable to some mobile web-sites, which rely on server-side mobile detection

Note: Some mobile sites may not render as expected in the desktop browser.

Note: For best results, use a Chrome browser.

Appendix F

Integrating with Perfecto Mobile

Perfecto Mobile is a cloud-based real-device mobile testing solution. It tests mobile apps on real devices, in real time, on live carrier networks.

WebLOAD enables creating and running scripts that run Perfecto Mobile scripts on mobile devices in the Perfecto Mobile cloud.

Note that you can only use this feature if you have an account in Perfecto Mobile.

Creating a Perfecto Mobile Script

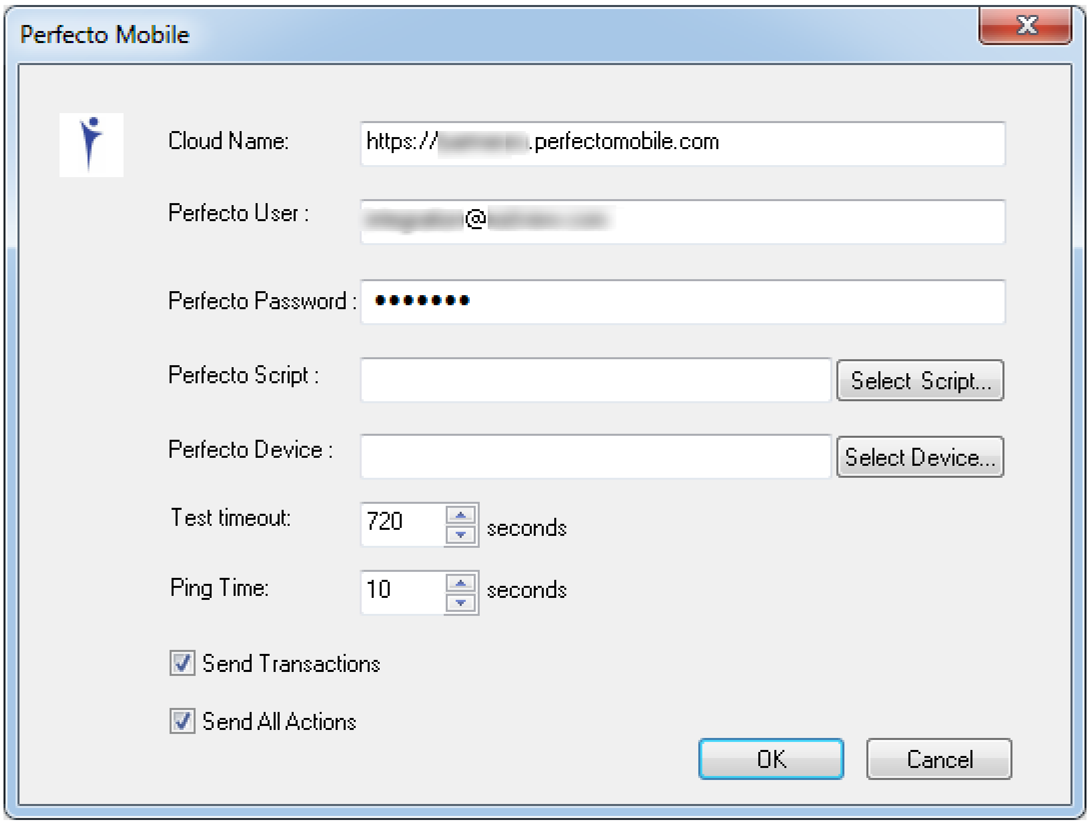
In order to create a Perfecto Mobile script, you must first create a Perfecto Mobile script in Perfecto Mobile. The WebLOAD script will execute this Perfecto Mobile script, and gather statistics from the s Perfecto Mobile cript and from the device.

For instructions on how to create a Perfecto Mobile script please refer to Perfecto Mobile tutorials.

To create a Perfecto Mobile WebLOAD script:

1. Drag the **Perfecto Mobile** icon from the Real Clients toolbox into the Script Tree at the desired location.

   The Perfecto Mobile dialog box opens.


Perfecto Mobile

Cloud Name: https://[redacted].perfectomobile.com

Perfecto User : [redacted]@[redacted]

Perfecto Password : •••••••

Perfecto Script :

Perfecto Device :

Test timeout: 720 seconds

Ping Time: 10 seconds

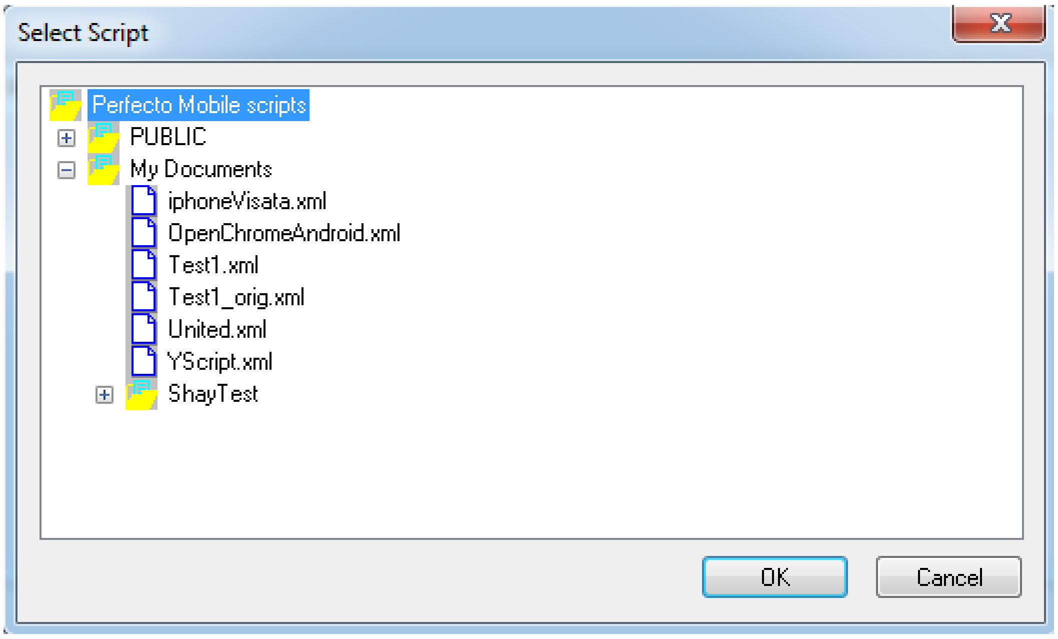
☑ Send Transactions

☑ Send All Actions

Figure 224: Perfecto Mobile Dialog Box

2. In the **Cloud Name** field, enter the URL for Perfecto Mobile.
3. Enter the Perfecto Mobile credentials:
   - In **Perfecto User**, enter your user name.
   - In **Perfecto Password**, enter your password.
4. Click **Select Script**. WebLOAD accesses the Perfecto Mobile scripts and displays them in the Select Script dialog box.

   Select a desired script and click **OK**.



Select Script

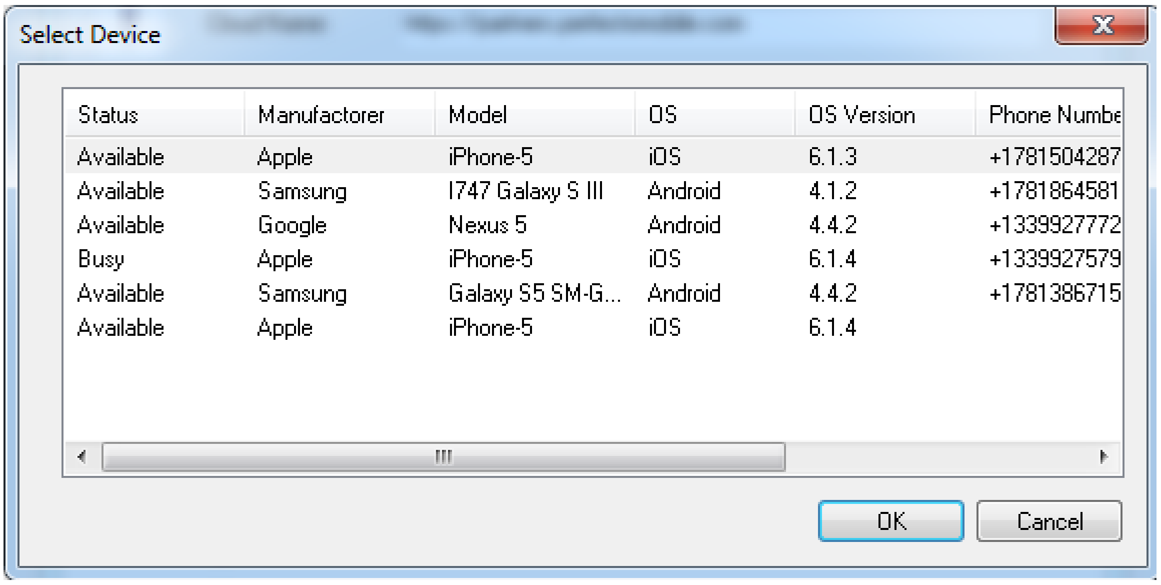
Perfecto Mobile scripts
- PUBLIC
- My Documents
  - iphoneVisata.xml
  - OpenChromeAndroid.xml
  - Test1.xml
  - Test1_orig.xml
  - United.xml
  - YScript.xml
  - ShayTest

OK Cancel

Figure 225: Select Script Dialog Box

5. Click **Select Device**. WebLOAD accesses Perfecto Mobile and displays the list of available devices in the Select Device dialog box.

   Select a desired device and click **OK**.



Select Device

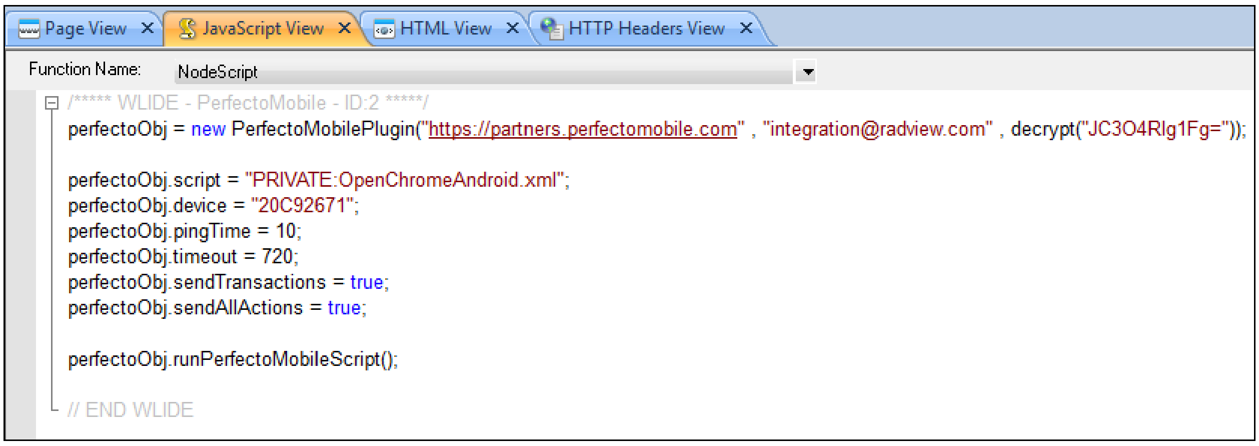
Status	Manufactorer	Model	OS	OS Version	Phone Numbe
Available	Apple	iPhone-5	iOS	6.1.3	+1781504287
Available	Samsung	I747 Galaxy S III	Android	4.1.2	+1781864581
Available	Google	Nexus 5	Android	4.4.2	+1339927772
Busy	Apple	iPhone-5	iOS	6.1.4	+1339927579
Available	Samsung	Galaxy S5 SM-G...	Android	4.4.2	+1781386715
Available	Apple	iPhone-5	iOS	6.1.4	

OK Cancel

Figure 226: Select Device Dialog Box

6. Set the **Test timeout**. This is the maximum number of seconds that the test is allowed to run.
7. Set the **Ping Time**. This is the frequency in seconds with which WebLOAD will check test status during test running.

8. Specify whether to **Send Transactions** to WebLOAD. That is, whether to send a list of all the transactions (Groups) that are written in the Perfecto Mobile script (such as: "Open Chrome", "Navigate to my application").
9. Specify whether to **Send All Actions** to WebLOAD. That is, whether to send a list of all the actions that the script performs (such as: "Press Key", "Touch Drag").
10. Click **OK**. The script is created, and is ready to be run.



Page View | JavaScript View | HTML View | HTTP Headers View

Function Name: NodeScript

```
/***** WLIDE - PerfectoMobile - ID:2 *****/
perfectoObj = new PerfectoMobilePlugin("https://partners.perfectomobile.com" , "integration@radview.com" , decrypt("JC3O4Rlg1Fg="));

perfectoObj.script = "PRIVATE:OpenChromeAndroid.xml";
perfectoObj.device = "20C92671";
perfectoObj.pingTime = 10;
perfectoObj.timeout = 720;
perfectoObj.sendTransactions = true;
perfectoObj.sendAllActions = true;

perfectoObj.runPerfectoMobileScript();

// END WLIDE
```

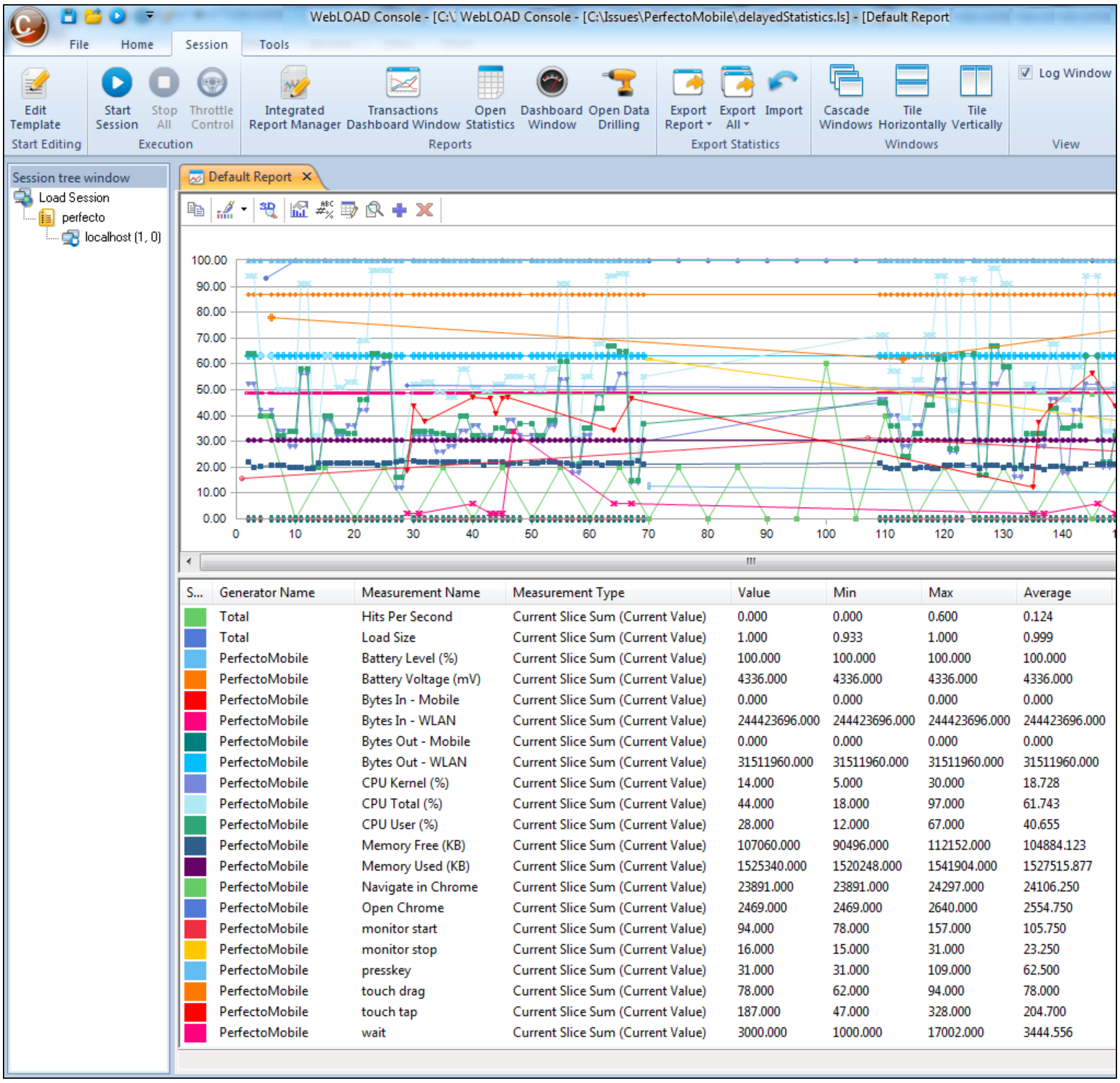
Figure 227: Script for Running a Script on a Device in the Perfecto Cloud

If you run the Perfecto Mobile script in WebLOAD Recorder, all you see are the periodic pings. In order to gather device, transaction and action statistics, you need to run the script in the WebLOAD Console, as described in the following section.

Viewing Session Results of a Perfecto Mobile Session

To gather statistics about the Perfecto Mobile script, create in the WebLOAD Console a Load template that runs the Perfecto Mobile script repetitively but for a single user (Probing Client).

The following figure shows an example results window. Note that the example displays also Actions and Transactions. That is, in addition to the various device statistics (such as Battery Level), statistics are also displayed for actions (such as Touch Drag) and transactions (such as Navigate in Chrome).



S...	Generator Name	Measurement Name	Measurement Type	Value	Min	Max	Average
🟩	Total	Hits Per Second	Current Slice Sum (Current Value)	0.000	0.000	0.600	0.124
🟦	Total	Load Size	Current Slice Sum (Current Value)	1.000	0.933	1.000	0.999
🟦	PerfectoMobile	Battery Level (%)	Current Slice Sum (Current Value)	100.000	100.000	100.000	100.000
🟧	PerfectoMobile	Battery Voltage (mV)	Current Slice Sum (Current Value)	4336.000	4336.000	4336.000	4336.000
🟥	PerfectoMobile	Bytes In - Mobile	Current Slice Sum (Current Value)	0.000	0.000	0.000	0.000
🟥	PerfectoMobile	Bytes In - WLAN	Current Slice Sum (Current Value)	244423696.000	244423696.000	244423696.000	244423696.000
🟩	PerfectoMobile	Bytes Out - Mobile	Current Slice Sum (Current Value)	0.000	0.000	0.000	0.000
🟦	PerfectoMobile	Bytes Out - WLAN	Current Slice Sum (Current Value)	31511960.000	31511960.000	31511960.000	31511960.000
🟦	PerfectoMobile	CPU Kernel (%)	Current Slice Sum (Current Value)	14.000	5.000	30.000	18.728
🟦	PerfectoMobile	CPU Total (%)	Current Slice Sum (Current Value)	44.000	18.000	97.000	61.743
🟩	PerfectoMobile	CPU User (%)	Current Slice Sum (Current Value)	28.000	12.000	67.000	40.655
🟦	PerfectoMobile	Memory Free (KB)	Current Slice Sum (Current Value)	107060.000	90496.000	112152.000	104884.123
🟪	PerfectoMobile	Memory Used (KB)	Current Slice Sum (Current Value)	1525340.000	1520248.000	1541904.000	1527515.877
🟩	PerfectoMobile	Navigate in Chrome	Current Slice Sum (Current Value)	23891.000	23891.000	24297.000	24106.250
🟦	PerfectoMobile	Open Chrome	Current Slice Sum (Current Value)	2469.000	2469.000	2640.000	2554.750
🟥	PerfectoMobile	monitor start	Current Slice Sum (Current Value)	94.000	78.000	157.000	105.750
🟨	PerfectoMobile	monitor stop	Current Slice Sum (Current Value)	16.000	15.000	31.000	23.250
🟦	PerfectoMobile	presskey	Current Slice Sum (Current Value)	31.000	31.000	109.000	62.500
🟧	PerfectoMobile	touch drag	Current Slice Sum (Current Value)	78.000	62.000	94.000	78.000
🟥	PerfectoMobile	touch tap	Current Slice Sum (Current Value)	187.000	47.000	328.000	204.700
🟥	PerfectoMobile	wait	Current Slice Sum (Current Value)	3000.000	1000.000	17002.000	3444.556

Figure 228: Results of Running a Perfecto Mobile Load Session in the WebLOAD Console

Appendix G

Selenium Integration

Selenium is a browser automation tool, enabling you to record and playback your interactions with a browser. Its development environment (IDE) is a Firefox add-on, and it can run various browser drivers, such as IE, Chrome, and Firefox. Selenium is open-source software, and can be used without charge.

WebLOAD enables integration of a Selenium script into WebLOAD, and the gathering of navigation timing statistics. This is useful for gathering information regarding user experience, such as how long it takes to login, or how long it takes a page to render. While WebLOAD simulates the load on a server, Selenium simulates the end-user experience. A hybrid WebLOAD-Selenium solution can thus give you the advantages of both simulation types. To implement such a solution, you can run a WebLOAD load session that does the following:

- Runs a regular WebLOAD script with many users.
- Run a Selenium script that simulates the same actions, for a single user.

The template will gather and display statistics from both scripts in the WebLOAD reports. In order to easily distinguish between the statistics from the regular WebLOAD script and those from the Selenium script, it is recommended to give the transactions in each of the two scripts a different distinctive name. For example:

- `Selenium-user-login` in the Selenium script
- `WL-user-login` in the WebLOAD script

Note: This appendix is intended for users creating Selenium code to be used in WebLOAD only. If you wish to use Selenium as usual and add WebLOAD navigation-timing functionality, refer to the *Working with Java Selenium Scripts* section in Chapter 3 *Advanced JavaScript Script Features* of the *WebLOAD Scripting Guide*.

Using Selenium Scripts

This section describes how to use Selenium scripts in WebLOAD.

Prerequisites

1. Download the Selenium required jars:
   a. In the selenium web-site (http://www.seleniumhq.org/), select the **Download** tab.
   b. In 'Selenium Client & WebDriver Language Bindings', download the Java version (selenium-java-x.x.zip).
2. Unzip the downloaded file in a temporary location.
3. Copy the selenium-java-x.x.jar and all jars in the 'libs' folder to the WebLOAD Java extensions folder: c:\ProgramData\RadView\WebLOAD\extensions\java
4. If there are duplicate jars (same base name with different number), delete the older versions.

Important: Selenium dependencies include jar files that are already shipped with WebLOAD, but not always the same version. Make sure there are no duplicates by deleting the older versions. For example, if you have both `commons-codec-1.9.jar` and `commons-codec-1.3.jar`, delete the `commons-codec-1.3.jar` file.

Note: Version numbers are not decimal numbers, so for example `commons-codec-1.10.jar` is newer than `commons-codec-1.3.jar`.

Step 1: Create a WebLOAD Selenium Script

Do one of the following:

- Record a Selenium script using the Selenium IDE, and export it to WebLOAD. Refer to *Recording and Exporting a Selenium Script* (on page 390). It is recommended to record the script both in Selenium IDE and in WebLOAD simultaneously.
- Create a Selenium script in WebLOAD using the Selenium building blocks. Refer to *Selenium Building Blocks* (on page 361).

Recording and Exporting a Selenium Script

Prerequisites for Selenium IDE Export

1. Install Firefox.
2. Install the Selenium IDE plug-in for Firefox.

3. Go to `https://addons.mozilla.org/en-US/firefox/addon/selenium-ide-webload-formatter/` and install the WebLOAD plug-in for Selenium that enables exporting Selenium scripts to WebLOAD.

Recording with Selenium IDE

1. Launch Selenium IDE by clicking the Selenium IDE button in the Firefox toolbar.

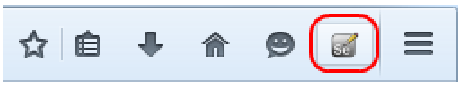
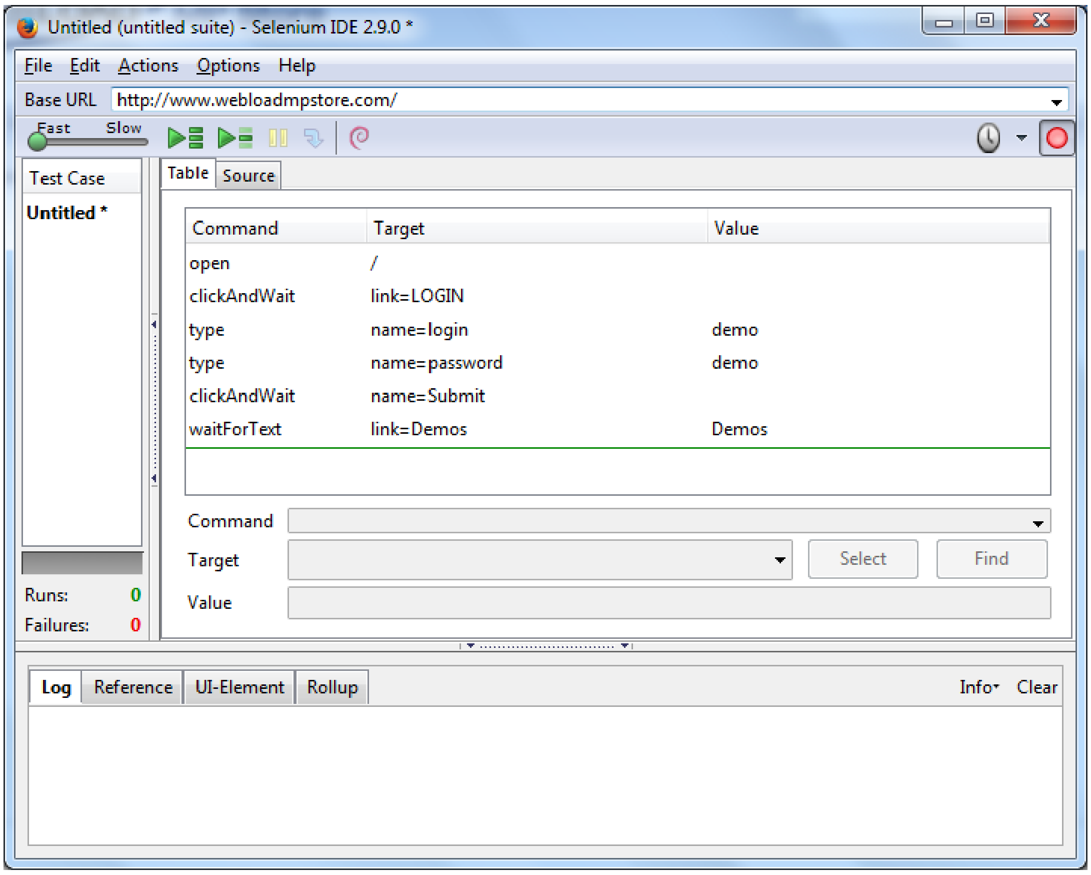


Figure 229: Launch Selenium IDE from Firefox

The Selenium IDE is launched.

2. Record a desired script in the Selenium IDE.



Base URL http://www.webloadmpstore.com/

Command	Target	Value
open	/	
clickAndWait	link=LOGIN	
type	name=login	demo
type	name=password	demo
clickAndWait	name=Submit	
waitForText	link=Demos	Demos

Runs: 0
Failures: 0

Figure 230: Recording a Script in Selenium IDE

Note: It is possible to record the same script both in Selenium and in WebLOAD simultaneously.

3. After you finish recording, export the Selenium script to WebLOAD format as follows:

   a. In the Selenium IDE, select **Export Test Case As** > **WebLOAD JavaScript**.

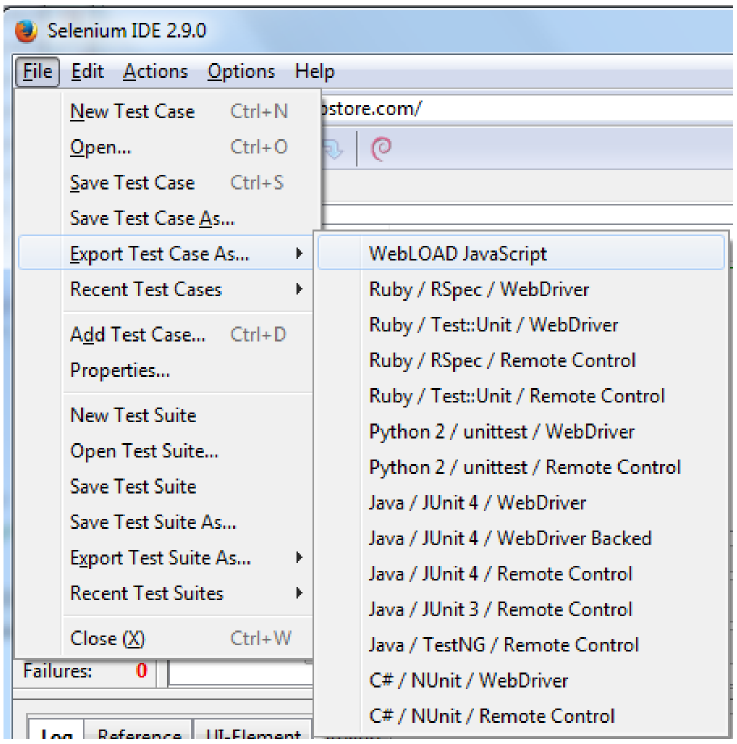


Figure 231: Exporting Selenium Script to WebLOAD JavaScript Format

   b. Enter a name for the script. The script is saved as a WebLOAD .js file.

   Note that the .js file contains a WebLOAD command, `reportStatistics`, which appears periodically throughout the script. The command instructs WebLOAD to collect navigation timing statistics from the browser regarding the current page.

Note: By default, WebLOAD reports navigation timing statistics as an average across all pages accessed by the Selenium Driver. If you wish to also display the statistics for a particular page separately, use a modified `reportStatistics` command, as described in *Selenium Report Statistics* (on page 363).

Step 2: Create a WebLOAD Recorder Project File from the Selenium Script

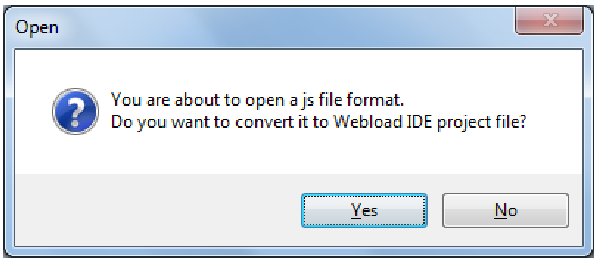
Perform one of the following:

- *Automatic Conversion to a Project File*
- Manual Conversion to a Project File

Automatic Conversion to a Project File

1. In the WebLOAD Recorder, click **Open** in the **File** tab of the ribbon and select the JavaScript .js file you exported from Selenium.

   The Open message appears.



You are about to open a js file format.
Do you want to convert it to Webload IDE project file?

Figure 232: Converting a JavaScript file to a WebLOAD Recorder project file

2. Click **Yes** to convert the JavaScript file to a WebLOAD Recorder project file.

Manual Conversion to a Project File

1. Open the JavaScript .js file you exported from Selenium, in a text browser such as Notepad.
2. Copy the contents of the file to the clipboard.
3. In the WebLOAD Recorder, select **New Project** in the **File** tab of the ribbon.
4. Click **Full Script** in the **Home** tab of the ribbon.
5. Paste the copied contents into the JavaScript View pane.
6. Save the project as a .wlp file.

Step 3: Edit Both Scripts

As mentioned above, it is possible to record the same actions both in Selenium and in WebLOAD simultaneously. Whether you do so or not, after creating a WebLOAD script that parallels the Selenium script, edit both scripts as desired.

At this point it is recommended to give the transactions in each of the two scripts different distinctive names so that you will be able to easily distinguish between the statistics of each when viewing the test results. For example, give the login action the following names:

- **`Selenium_user_login`** – in the Selenium script
- **`WL_user_login`** – in the WebLOAD script

Step 4: Run Both Scripts in a Test Session

1. In the WebLOAD Console, define a load session template that does the following:
   - Runs the Selenium script with a single user.
   - Runs the regular WebLOAD script with many users.

Note: For instructions on how to define two different scripts in a single template, refer to *Adding a New Script/Mix to the Template*, in the *WebLOAD™ Console User's Guide*.

2. Run the load session.

Step 5: View the Test Results

View the load session test results in the WebLOAD Console, WebLOAD Analytics, or the WebLOAD Web Dashboard.

The following table describes the navigation timing measurements. These measurements are gathered by WebLOAD's `reportStatistics` command. By default, these measurements report navigation timing statistics as an average across all pages for which the `reportStatistics` command was invoked.

If you added an optional `Page Name` parameter to a `reportStatistics` command (refer to *Selenium Report Statistics* on page 363), the resultant measurement names will be displayed as <Measurement Name>-<Page Name>. Thus for example, the measurement `Selenium Redirect - Home Page` displays the redirect time for the home page.

Note that in all navigation timing measurements, time is reported in units of seconds.

Table 104: Navigation Timing Statistics

Measurement Name	Type	Definition
Selenium Redirect Time	Timer	Redirection Time. The time from `redirectStart` to `redirectEnd`.
Selenium App Cache Time	Timer	Fetch from cache, if relevant. The time from `redirectEnd` to `fetchStart`.
Selenium DNS Time	Timer	DNS Lookup time. The time from `domainLookupStart` to `domainLookupEnd`.
Selenium Connect Time	Timer	TCP Connect time. The time from `connectStart` to `connectEnd`.
Selenium Request Time	Timer	Request until first-byte. The time from `requestStart` to `responseStart`.

Measurement Name	Type	Definition
Selenium First Byte Time	Timer	The time from `navigationStart` to `responseStart`. Note that the First Byte is calculated since navigation start, and not since request end as in WebLOAD.
Selenium Page Response Time	Timer	Response download time. The time from `responseStart` to `responseEnd`.
Selenium domInteractive Time	Timer	The time from `navigationStart` to `domInteractive`.
Selenium domContentLoaded Time	Timer	The time from `navigationStart` to `domContentLoadedEventStart`. Note that this measures the time until domContentLoadedEvent-Start but not event end, meaning until all content is loaded, but not including the javascript execution.
Selenium Processing Time	Timer	The time from load `responseEnd` to `loadEventStart`.
Selenium onLoad Time	Timer	The time from `loadEventStart` to `loadEventEnd`.
Selenium Full Page Time	Timer	The time from `navigationStart` to `loadEventEnd`.

The following figure explains the various timing attributes listed in Table 104.

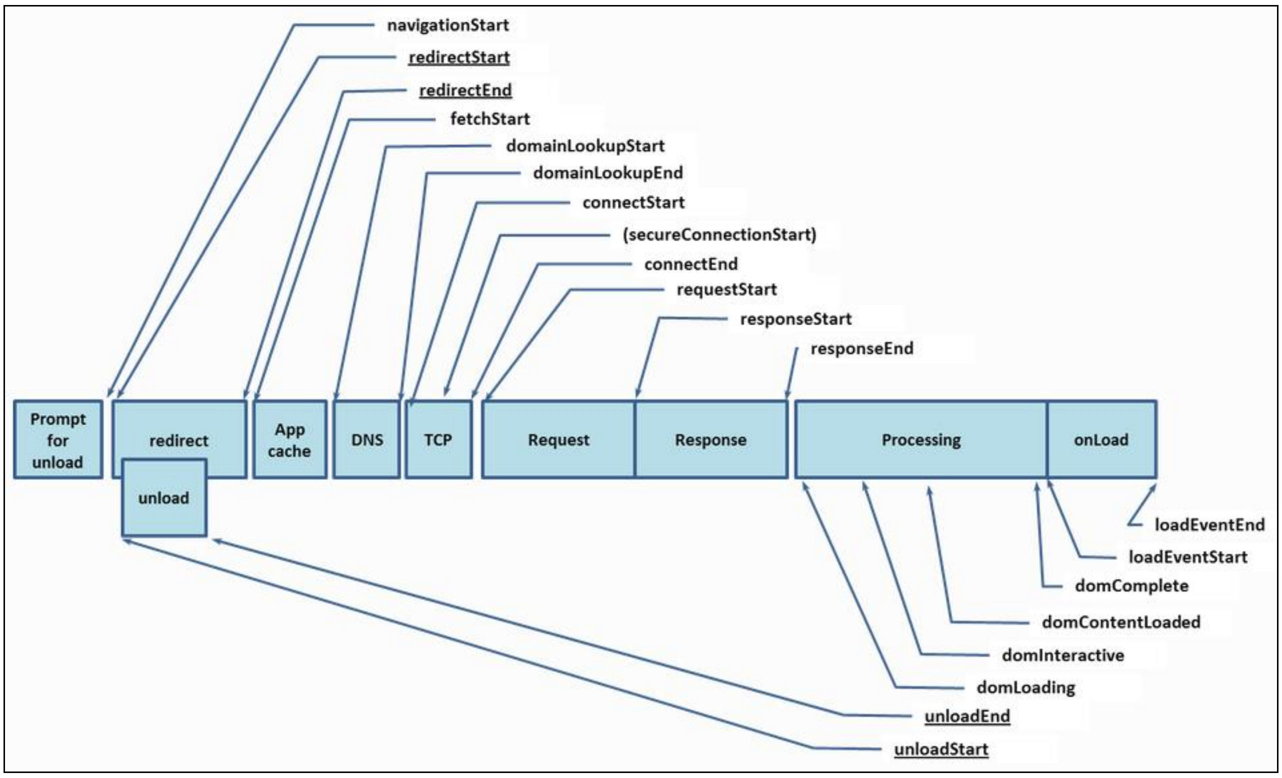


Figure 233: Timing Attributes used in Navigation Timing Measurements

Appendix H

Glossary

Glossary Term	Description
AAT	An older, obsolete WebLOAD utility that was used for recording web session activities as a JavaScript file. It is replaced by WebLOAD Recorder.
Aborted Rounds	The number of times the Virtual Clients started to run a script but did not complete the script, during the last reporting interval. This might be due to a session being stopped either automatically or manually by the user.
Script	Specification of the sequence of HTTP protocol calls sent by Virtual Clients to the SUT (System Under Test). scripts are written in JavaScript. You can either write scripts as a text file or generate them automatically using the WebLOAD Recorder.
Application Being Tested (ABT)	See *SUT*.
Attempted Connections	The total number of times the Virtual Clients attempted to connect to the SUT during the last reporting interval.
Automatic Transaction counters	If you have Automatic Transactions enabled, WebLOAD creates three counters for each GET and POST statement in the script: • The total number of times it occurred • The number of times it succeeded • The number of times it failed during the last reporting interval.
Average	For timers, average is the total amount of time counted by the timer (not the elapsed time) divided by the Count (that is, the total number of readings). For example, the average for Transaction Time is the amount of time it took to complete all the successful transactions, divided by the number of successful transactions (the Count).
Built-in Timer	A timer measures the time required to perform a given task. WebLOAD supports both programmed timers and built-in timers. ROUND TIME is a built-in timer. The ROUND TIME is the time needed for one complete execution of a script.

Glossary Term	Description
Connect Time	The time it takes for a Virtual Client to connect to the System Under Test (the SUT), in seconds. In other words, the time it takes from the beginning of the HTTP request to the TCP/IP connection. The value posted in the Current Value column is the average time it took a Virtual Client to connect to the SUT during the last reporting interval. If the Persistent Connection option is enabled, there may not be a value for Connect Time because the HTTP connection remains open between successive HTTP requests.
Connection Speed (Bits Per Second)	The number of bits transmitted back and forth between the Virtual Clients and the System Under Test (SUT), divided by the time it took to transmit those bits, in seconds. You can set the Virtual Clients to emulate a particular connection speed during the test, either by using the Variable Connection Speed settings, or by coding the connection speed in the script. If a connection speed is specified for the test, WebLOAD reports it in the Statistics Report. The value posted in the Current Value column is the number (sum) of bits passed per second during the last reporting interval. It should match, very closely, the connection speed you specified for the test.
Console	The WebLOAD component that manages the test session. The Console performs the following: • Configures Load Session hosts and scripts • Schedules Load Session scripts • Configures Goal–Oriented test sessions • Monitors the application's performance under the generated load • Manages the Load Session as it is running, allowing you to pause, stop, and continue Load Session components as needed • Displays the current performance of the SUT • Provides a final performance reports for Probing Clients and Virtual Clients • Manages exporting of performance reports

Glossary Term	Description
Count	(For timers only.) The total number of readings (the number of times the item being timed has occurred) for the timed statistic since the beginning of the test. For example, for Transaction Time, Count shows the number of transactions that have been completed.
Current Slice	The value posted for this reporting interval in the Statistics Report main window.
Current Slice Average	For per time unit statistics and counters, average is the total of all of the current values for the last reporting interval, divided by the number of readings. For timers, average is the total amount of time counted by the timer (not the elapsed time), divided by the Count (that is, the total number of readings for the last reporting interval). For example, the average for Transaction Time is the amount of time it took to complete all the successful transactions in the last reporting interval, divided by the number of successful transactions (the Current Slice Count).
Current Slice Count	(For timers only.) The total number of readings (the number of times the item being timed has occurred) for the timed statistic for the last reporting interval. For example, for Transaction Time, Current Slice Count shows the number of transactions that have been completed over the last reporting interval.
Current Slice Max	The highest value reported for this statistic over the last reporting interval.
Current Slice Min	The lowest value reported for this statistic over the last reporting interval.
Current Slice Standard Deviation	The average amount the measurement for this statistic varies from the average over the last reporting interval.
Current Slice Sum	The aggregate or total value for this statistic in this script over the last reporting interval.
DNS Lookup Time	The time it takes to resolve the host name and convert it to an IP address by calling the DNS server.
Failed Connections	The total number of times the Virtual Clients tried to connect to the SUT but were unsuccessful, during the last reporting interval. This number is always less than or equal to the number of failed hits because hits can fail for reasons other than a failed connection.

Glossary Term	Description
Failed Hits	The total number of times the Virtual Clients made an HTTP request but did not receive the correct HTTP response from the SUT during the last reporting interval. Note that each request for each `gif`, `jpeg`, `html` file, etc., is a single hit.
Failed Hits Per Second	The number of times the Virtual Clients did not obtain the correct HTTP response, divided by the elapsed time, in seconds. The value posted in the Current Value column is the number (sum) of unsuccessful HTTP requests per second during the last reporting interval.
Failed Pages Per Second	The number of times the Virtual Clients did not obtain the correct response to an upper level request, divided by the elapsed time, in seconds. The value posted in the Current Value column is the number (sum) of unsuccessful requests per second during the last reporting interval.
Failed Rounds	The total number of times the Virtual Clients started but did not complete the script during the last reporting interval.
Failed Rounds Per Second	The number of times the Virtual Clients started but did not complete an iteration of the script, divided by the elapsed time, in seconds. The value posted in the Current Value column is the number (sum) of failed iterations of the script per second during the last reporting interval.
First Byte	The time it takes a Virtual Client to receive the first byte of data.
Gallery	See *Templates Gallery*.
Goal–Oriented Test	A WebLOAD component enabling you to define the performance goals required, and view the status of your application when it is operating under this performance goal. WebLOAD provides a Goal–Oriented Test Wizard for configuring these performance goals. WebLOAD automatically accelerates the number of Virtual Clients accessing your website until you meet your performance goal. **Note:** The Goal-Oriented Test Wizard was previously called the Cruise Control Wizard.
Goal–Oriented Test Wizard	See *Goal–Oriented Test*.

Glossary Term	Description
Hit Time	The time it takes to complete a successful HTTP request, in seconds. Each request for each `gif`, `jpeg`, `html` file, etc., is a single hit. The time of a hit is the sum of the Connect Time, Send Time, Response Time, and Process Time. The value posted in the Current Value column is the average time it took to make an HTTP request and process its response during the last reporting interval.
Hits	The total number of times the Virtual Clients made HTTP requests to the System Under Test (SUT) during the last reporting interval. For example, a Get statement for a URL retrieves a page. The page can include any number of graphics and contents files. Each request for each `gif`, `jpeg`, `html` file, etc., is a single hit.
Hits Per Second	The number of times the Virtual Clients made an HTTP request, divided by the elapsed time, in seconds. Each request for each `gif`, `jpeg`, `html` file, etc. is a single hit. The value posted in the Current Value column is the number (sum) of HTTP requests per second during the last reporting interval.
Host	A computer connected via a network, participating in a test session. Each Host in a test session has assigned tasks. A host can act as either a Load Machine or a Probing Client Machine. All hosts participating in a test session must be accessible to the Console over a network. Therefore they must run TestTalk, the network agent.
HTTP Response Status	WebLOAD creates a row in the Statistics Report for each kind of HTTP status code it receives as an HTTP response from the SUT (redirection codes, success codes, server error codes, or client error codes). The value posted is the number of times the Virtual Clients received that status code during the last reporting interval.
Integrated Report	A single configurable report that can integrate both standard performance data, and data from the NT Performance Monitor. This report gives you a more complete picture of the performance of your application. The data to be monitored and the data to be displayed in the report are both configurable in the Console.
Internet Productivity Pack (IPP)	Provides a set of protocol implementations enabling you to load-test your application using these protocols.

Glossary Term	Description
Java and ActiveX counters	You can add function calls to your scripts that enable you to instantiate and call methods and properties in Java and ActiveX components (see the *WebLOAD Scripting Guide*). If there are ActiveX or Java function calls in the script that you are running, WebLOAD reports three counters for them in the Statistics Report: • The total number of times it occurred • The number of times it succeeded • The number of times it failed during the last reporting interval. The row heading in the Statistics Report is the name of the function call.
Java and ActiveX timers	You can add function calls to your scripts that enable you to instantiate and call methods and properties in Java and ActiveX components (see the *WebLOAD Scripting Guide*). If there are ActiveX or Java function calls in the script you are running, WebLOAD reports timers for them in the Statistics Report. The timer value is the average amount of time it took to complete the function call, in seconds, during the last reporting interval. The row heading in the Statistics Report is the name of the function call.
Load Generator	The component of the Load Machine that generates Virtual Clients. Load Generators have the task of bombarding the System Under Test with HTTP protocol call requests as defined in the script. WebLOAD assesses the application's performance by measuring the response time experienced by the Virtual Clients. The number of Virtual Clients at any given moment is determined by the user.
Load Generator Machine	See *Load Machine*.
Load Machine	A host that runs Load Generators. Load Generators bombard the application under test with a large load, to enable complete scalability and integrity testing.

Glossary Term	Description
Load Session	A Load Session includes both the complete Load Template and the results obtained while running that Load Session. A Load Template consists of information about the hosts and scripts participating in the current Load Session. The Load Template will also include scheduling information. The complete Load Template is illustrated in the Session Tree. Storing a Load Template saves you time when repeatedly running WebLOAD with the same, or even a similar network configuration, since you don't have to recreate your Load Template from scratch each time you want to start working. Storing Load Session results can be useful when you want to examine results from multiple test sessions or for analyzing test session results.
Load Size	The number of Virtual Clients running during the last reporting interval.
Load Template	A Load Template contains the complete Load Session definition, without the test results. A Load Template includes information about the participating hosts and the scripts used in the current Load Session. The definition also includes scheduling information and the configuration of the Server Monitor and Integrated Reports. The complete Load Template is illustrated in the Session Tree. Storing a Load Template saves you time when repeatedly running WebLOAD with the same, or even a similar network configuration, since you do not have to recreate your Load Template from scratch each time you rerun a test.
Page Time	The time it takes to complete a successful upper level request, in seconds. The Page Time is the sum of the Connection Time, Send Time, Response Time, and Process Time for all the hits on a page. The value posted in the Current Value column is the average time it took the Virtual Clients to make an upper level request and process its response during the last reporting interval.
Pages	The total number of times the Virtual Client made upper level requests, both successful and unsuccessful, during the last reporting interval.
Pages Per Second	The number of times the Virtual Clients made upper level requests divided by the elapsed time, in seconds. The value posted in the Current Value column is the number (sum) of requests per second during the last reporting interval.

Glossary Term	Description
Per Time Unit statistics	Ratios that calculate an average value for an action or process. For example: Transactions Per Second, Rounds Per Second.
Portfolio	A Portfolio of reports enables you to generate a single, inclusive report that contains all the charts generated by the templates included in the portfolio.
Probing Client	A software program which "bombards" the SUT as a single Virtual Client, to further measure the performance of the SUT. WebLOAD generates exact values for Probing Client performance.
Probing Client Machines	Hosts running Probing Client software simulating one Virtual Client, and running at the same time as Load Machines.
Probing Client software	See *Probing Client*.
Process Time	The time it takes WebLOAD to parse an HTTP response from the SUT and then populate the document-object model (DOM), in seconds. The value posted in the Current Value column is the average time it took WebLOAD to parse an HTTP response during the last reporting interval.
Receive Time	The elapsed time between receiving the first byte and the last byte.
Report Portfolio	See *Portfolio*.
Resource Manager	Distributes and circulates WebLOAD testing resources (Virtual Clients and Probing Clients) amongst users on a "need to use" basis. The Resource Manager is packaged with a maximum number of Virtual Clients, Probing Clients and Connected Workstation ports, as defined by the WebLOAD package. With the Resource Manager, every WebLOAD Console can operate in Standalone Workstation mode or Connected Workstation mode.
Response Data Size	The size, in bytes, of all the HTTP responses sent by the SUT during the last reporting interval. WebLOAD uses this value to calculate Throughput (bytes per second).

Glossary Term	Description
Response Time	The time it takes the SUT to send the object of an HTTP request back to a Virtual Client, in seconds. In other words, the time from the end of the HTTP request until the Virtual Client has received the complete item it requested. The value posted in the Current Value column is the average time it took the SUT to respond to an HTTP request during the last reporting interval.
Responses	The number of times the SUT responded to an HTTP request during the last reporting interval. This number should match the number of successful hits.
Round Time	The time it takes one Virtual Client to finish one complete iteration of a script, in seconds. The value posted in the Current Value column is the average time it took the Virtual Clients to finish one complete iteration of the script during the last reporting interval.
Rounds	The total number of times the Virtual Clients attempted to run the script during the last reporting interval.
Rounds Per Second	The number of times the Virtual Clients attempted to run the script, divided by the elapsed time, in seconds. The value posted in the Current Value column is the number (sum) of attempts (both successful and unsuccessful) per second during the last reporting interval.
Send Time	The time it takes the Virtual Client to write an HTTP request to the SUT, in seconds. The value posted in the Current Value column is the average time it took the Virtual Clients to write a request to the SUT during the last reporting interval.
Server Performance Measurements	If you selected Performance Monitor statistics for the report, WebLOAD creates a row for them and reports their values in the Statistics Report. For definitions of the statistics, see the Server Monitor Definition dialog box. Be selective when choosing server performance measurements , otherwise the system resources required to manage the data might affect the Console.

Glossary Term	Description
Session Tree	A graphic representation of a Load Template and status. It illustrates the different components of a test session, including Load Machines and Probing Clients, the scripts that they execute, and their status.
Single Client	See *Probing Client*.
Standard Deviation	The average amount the measurement varies from the average since the beginning of the test.
Successful Connections	The total number of times the Virtual Clients were able to successfully connect to the SUT during the last reporting interval. This number is always less than or equal to the number of successful hits because several hits might use the same HTTP connection if the Persistent Connection option is enabled.
Successful Hits	The total number of times the Virtual Clients made an HTTP request and received the correct HTTP response from the SUT during the last reporting interval. Each request for each `gif`, `jpeg`, `html` file, etc., is a single hit.
Successful Hits Per Second	The number of times the Virtual Clients obtained the correct HTTP response to their HTTP requests divided by the elapsed time, in seconds. The value posted in the Current Value column is the number (sum) of successful HTTP requests per second during the last reporting interval.
Successful Pages Per Second	The value posted in the Current Value column is the number (sum) of successful requests per second during the last reporting interval.
Successful Rounds	The total number of times the Virtual Clients completed one iteration of the script during the last reporting interval.
Successful Rounds Per Second	The number of times the Virtual Clients completed an entire iteration of the script, divided by the elapsed time, in seconds. The value posted in the Current Value column is the number (sum) of successful iterations of the script per second during the last reporting interval.
SUT	The system running the Web application currently under test. The SUT (System Under Test) is accessed by clients through its URL address. The SUT can reside on any machine or on multiple machines, anywhere on the global Internet or your local intranet.
Template	See *Load Template*.

Glossary Term	Description
Templates Gallery	The Templates Gallery is a single entity that contains predefined templates, user-defined templates, and portfolios.
Test Program	See *Test Script*.
Test Script	The script. This defines the test scenario used in your Load Session. Scripts are written in JavaScript.
Test Template	See *Load Template*.
TestTalk	The network agent. This program enables communication between the Console and the host computers participating in the test.
Throttle Control	A WebLOAD component that enables you to dynamically change the Load Size while a test session is in progress.
Throughput (Bytes Per Second)	The average number of bytes per second transmitted from the SUT to the Virtual Clients running the script during the last reporting interval. In other words, this is the amount of the Response Data Size, divided by the number of seconds in the reporting interval.
Time to First Byte	The time that elapsed since a request was sent until the Virtual Client received the first byte of data.
User-defined Automatic Data Collection	If you have Automatic Data Collection enabled, WebLOAD creates three counters for each GET and POST statement in the script: • The total number of times the Get and Post statements occurred • The number of times the statements succeeded • The number of times the statements failed during the last reporting interval.
User-defined counters	Your own counters that you can add to scripts using the `SendCounter()` and the `SendMeasurement()` functions (see the *WebLOAD Scripting Guide*). If there is a user-defined counter in the script that you are running, WebLOAD reports the counter's values in the Statistics Report. The row heading is the name (argument) of the counter. That is, the row heading is the string in parenthesis in the `SendCounter()` or `SendMeasurement()` function call. The value reported is the number of times the counter was incremented during the last reporting interval.

Glossary Term	Description
User-defined timer	Timers that you can add to scripts to keep track of the amount of time it takes to complete specific actions (see the *WebLOAD Scripting Guide*). If there are any timers in the scripts that you are running, WebLOAD reports their values in the Statistics Report. The row heading is the name (argument) of the timer. That is, the row heading is the string in parenthesis in the `SetTimer()` function call. The timer represents the time it takes to complete all the actions between the `SetTimer()` call and its corresponding `SendTimer()` call, in seconds. The value posted is the average time it took a Virtual Client to complete the actions between the pair of timer calls, in seconds, during the last reporting interval.
User-defined Transaction counters	Transaction functions that you can add to scripts for functional tests (see the *WebLOAD Scripting Guide*). If there is a user-defined transaction function in the script that you are running, WebLOAD reports three counters for it in the Statistics Report: • The total number of times the transaction occurred • The number of times a transaction succeeded • The number of times a transaction failed during the last reporting interval. The row heading is the name (argument) of the transaction. That is, the row heading is the string in parenthesis in the `BeginTransaction()` function call.
User-defined Transactions timers	A timer for user-defined transaction functions. If there is a user-defined transaction function in the script that you are running, WebLOAD reports a timer for it in the Statistics Report. The row heading is the name (argument) of the user-defined transaction. That is, the row heading is the string in parenthesis in the `BeginTransaction()` function call. The timer represents the average time it took to complete all the actions between the `BeginTransaction()` call and its corresponding `EndTransaction()` call, in seconds, during the last reporting interval.

Glossary Term	Description
Virtual Client	Artificial entities run by Load Generators. Each such entity is a perfect simulation of a real client accessing the System Under Test (SUT) through a Web browser. Virtual Clients generate HTTP calls that access the SUT. The Load Generators that run Virtual Clients can reside anywhere on the Internet or on your local intranet. scripts are executed by all the Virtual Clients in parallel, achieving simultaneous access to the SUT. The size of the load on your SUT is determined by the number of Virtual Clients being generated. You may define as many Virtual Clients as needed, up to the maximum supported by your WebLOAD "package."
WebLOAD Analytics	WebLOAD Analytics enables you to analyze data, and create custom, informative reports after running a WebLOAD test session.
WebLOAD Console	See *Console*.
WebLOAD Recorder	An easy-to-use tool for recording, creating, and authoring protocol scripts for the WebLOAD environment.
WebLOAD Load Template	See *Load Template*.
WebLoad Session	See *Load Session*.
WebLOAD Wizard	A WebLOAD Wizard that steps you through the configuration process. Each screen of the WebLOAD Wizard contains text explaining the configuration process. The WebLOAD Wizard enables you to create a basic Load Template. After using the demo, you can use the Console ribbon to add functionality not available through the WebLOAD Wizard.
WebRM	See *Resource Manager*.

Index