WebLOAD IDE User's Guide

Version 8.0

RadView Software

WebLOAD IDE User's Guide

January, 2007, RadView Publication Number

For product assistance or information, contact:

Toll free in the US:	1-888-RadView
Fax:	(781) 238-8875
World Wide Web:	www.RadView.com

North American Headquarters:

RadView Software Inc.
7 New England Executive Park
Burlington, MA 01803
Email: sales@RadView.com
Phone: (781) 238-1111

International Headquarters:

RadView Software Ltd.
14 Hamelacha Street, Park Afek
Rosh Haayin, Israel 48091
Email: sales@RadView.com
Phone: +972-3-915-7060
Fax: +972-3-915-7683

Table of Contents

C H A P T E R 1

Introduction

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

In This Chapter

WebLOAD Documentation

WebLOAD is supplied with the following documentation:

WebLOAD IDE ™ User's Guide

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

WebLOAD ™ User's Guide

A guide to using WebLOAD console, RadView's load/scalability testing tool to easily and efficiently test your Web-based applications.

WebLOAD REPORTER™ User's Guide

Instructions on how to use WebLOAD REPORTER 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.

TestView™ Suite Programmer's Guide

Complete information on programming and editing JavaScript Agendas for use in WebLOAD and WebLOAD IDE.

TestView™ Suite JavaScript Reference Manual

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

TestView™ Suite User's Guide

Instructions for working with the WebLOAD Testing Suite, including the TestView Manager and TestView Scheduler.

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 Web site (http://www.adobe.com http://www.adobe.com).

Typographical Conventions

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

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

Formatting convention	Type of Information
Triangular Bullet(➢)	Step-by-step procedures. You can follow these instructions to complete a specific task.
Special Bold	Items you must select, such as menu 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.

Formatting convention	Type of Information
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

TestView 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

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

- North American Headquarters

  e-mail: `support@RadView.com`

  Phone: 1-888-RadView (1-888-723-8439) (Toll Free)

  781-238-1111

  Fax: 781-238-8875

- International Headquarters

  e-mail: `support@RadView.com`

  Phone: +972-3-915-7060

  Fax: +972-3-915-7683

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.

Technical Support Web Site

The technical support pages on our Web site contain:

- FAQ (Frequently Asked / Answered Questions).
- Agenda Center
- Documentation
- RadView's Product Resource Center, where you can find prepared test scripts, product information, and industry related news.
- http://www.RadView.com/support/index.asp (http://www.radview.com/support/index.asp)

C H A P T E R 2

Overview of the WebLOAD Integrated Development Environment

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

In This Chapter

About WebLOAD IDE

WebLOAD IDE (Integrated Development Environment) is an easy-to-use tool for recording, creating, and authoring protocol test scripts for the WebLOAD environment.

Note: WebLOAD IDE is installed as part of the TestView Suite. The WebLOAD IDE license file is limited to the computer system (machine) on which WebLOAD IDE is initially installed. Before installing WebLOAD IDE ensure you are installing on the machine you intend on working with.

TestView supports another type of license. For more information, see the TestView User's Guide.

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

- Recording Agendas
- Editing Agendas
- Running and Debugging Agendas

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

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

The Agenda 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 Agenda in WebLOAD IDE.
3. Test the application in WebLOAD using the Agenda created in WebLOAD IDE.

   You do not need to modify the test Agenda as it can run from WebLOAD IDE 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 Agendas are executed during WebLOAD test sessions by multiple Virtual Clients in parallel, achieving simultaneous access to the (ABT) and generating the load burden necessary for effective testing. Each execution of the Agenda generates an independent instance running in parallel during your WebLOAD test session.

Note: Please see the WebLOAD documentation for more information about using WebLOAD.

Agenda Creation

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

An *Agenda* 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 Application Being Tested (ABT). For example, if you want to test the

performance of your Web application when clients access a certain page, your Agenda must contain the code for accessing the page.

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

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

Agenda creation is carried out in a number of steps:

1. Recording the Agenda.

2. Editing / enhancing the Agenda.

3. Running and debugging the Agenda.

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

WebLOAD IDE 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 IDE stores information about you actions in an Agenda file. Externally, your activities are represented in WebLOAD IDE by a set of icons arranged in a Visual Agenda Tree. Internally, WebLOAD IDE records these actions and automatically creates Agendas that act as scripts, recreating the actions and verifying the functionality of Web sites under realistic conditions.

The second step of creating an Agenda is editing the code of the recorded Agenda. This can be done in Visual Editing mode and/or JavaScript Editing mode. For example, if you want an Agenda 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 manual describes how to edit the code in your Agendas to add more complex functionality to your testing sessions.

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

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

Note: For examples of basic Agendas, see the *TestView Programmer's Guide*.

The Recording Tool

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

- Recording in the Microsoft Internet Explorer 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 Console. It runs under MS Windows 2000, XP, 2003.

For detailed instructions on using WebLOAD IDE to record Agendas, see Recording Agendas (on page 43).

The Editing Modes

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

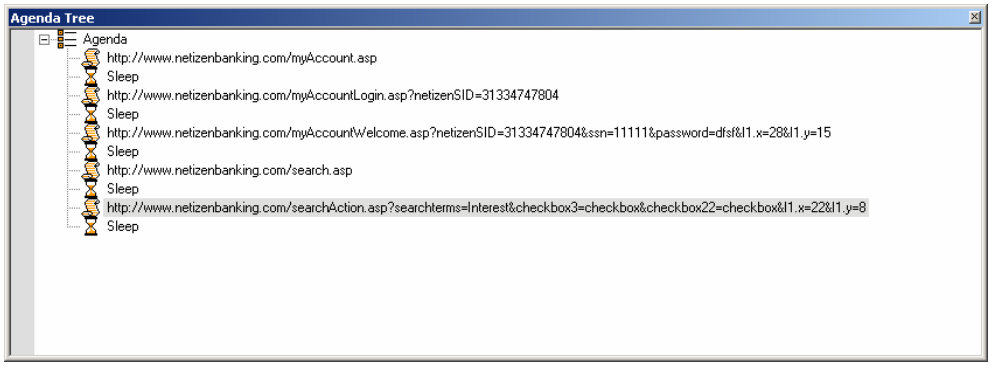
- Visual Editing mode
- JavaScript Editing mode

You can switch between modes while customizing Agendas.

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 Agenda Tree.

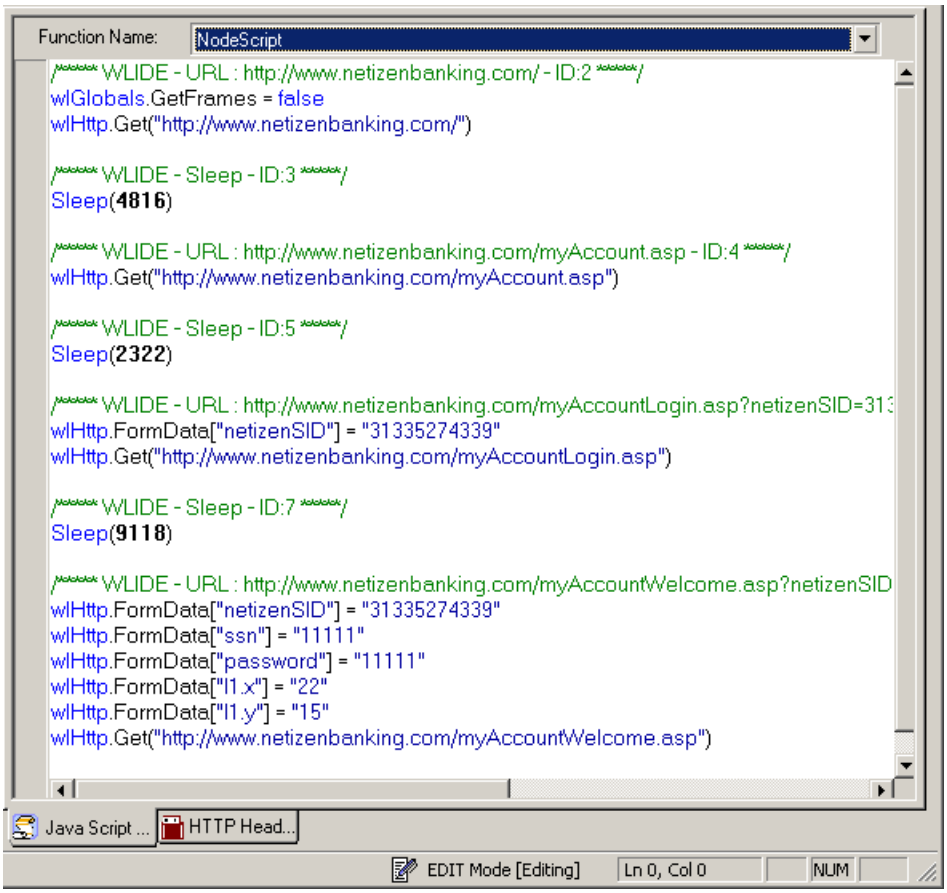
When editing an Agenda, you can also drag and drop items from the WebLOAD IDE toolboxes into the Agenda Tree. This makes programming easier by building the code behind an intuitive drag-and-drop interface.



Each node in the Agenda Tree is a graphical representation of the JavaScript code.

JavaScript Editing Mode

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



WebLOAD IDE provides the following programming assistance to manually edit an Agenda:

- 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 Agenda 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, please refer to the section entitled *The Core JavaScript Language*. This manual is supplied in Adobe Acrobat format with the TestView 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 TestView test environment.

The Run Mode

WebLOAD IDE enables you to run the Agenda and view the results. You can then debug the Agenda.

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

C H A P T E R 3

Before You Begin using WebLOAD IDE

This section provides information before you begin using WebLOAD IDE.

In This Chapter

Before You Begin

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

If you plan to record an Agenda that includes retrieving a page that you accessed previously during that Agenda, you should clear the cache in the browser. See Clearing the Cache in Your Browser (on page 20).

When you have completed these startup steps, you can either start working with WebLOAD IDE immediately, or you can configure the recording options first. For more information about configuring the recording options, see Configuring the Record Options (on page 101).

Clearing the Cache 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 an Agenda.

WebLOAD IDE records only HTTP protocol statements. 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 an Agenda.

To configure your browser to work without a cache file, see Clearing the Cache for Internet Explorer (on page 20), Clearing the Cache for Netscape Navigator (on page 20), and Clearing the Cache for Mozilla Firefox 2.0 (on page 21).

Clearing the Cache for Internet Explorer

To clear the cache for Internet Explorer:

1. Start Internet Explorer.
2. Select either View | Internet Options or Tools | Internet Options.
3. In the Internet Options dialog box, select the General tab.
4. Under Temporary Internet Files, click Delete Files.
5. Click OK.

   The cache is cleared.

Clearing the Cache for Netscape Navigator

To clear the cache for Netscape Navigator:

1. Start Netscape Navigator.
2. Select Edit | Preferences.
3. In the Preferences dialog box, under Category, expand Advanced and select Cache.

   The Cache screen appears.
4. Set the Memory Cache to zero (0) Kbytes and click Clear Memory Cache.
5. Set the Disk Cache to zero (0) Kbytes and click Clear Disk Cache.
6. Click OK.

   The cache is cleared and is configured so that it does not cache new files.

Clearing the Cache for Mozilla Firefox 2.0

To clear the cache for Mozilla Firefox 2.0:

1. Start Mozilla Firefox.
2. Select Tools | Clear Private Data.

   The Clear Private Data dialog box opens.
3. In the Clear Private Data dialog box, select the items that you want to clear, and click Clear Private Data Now.

   The cache is cleared.

In Mozilla Firefox 2.0, there is an alternate method for clearing the cache.

To clear the cache for Mozilla Firefox 2.0 (alternate method):

1. Start Mozilla Firefox.
2. Select Tools | Options.
3. Click Privacy.
4. Click Clear Now in the Private Data area.

   The Clear Private Data dialog box opens.
5. In the Clear Private Data dialog box, select the items that you want to clear, and click Clear Private Data Now.

   The Clear Private Data dialog box closes.
6. Click OK.

   The cache is cleared.

Configuring the Proxy Value for Your Browser

Before you begin recording Agendas using WebLOAD IDE, your browser must be configured to use a specific proxy setting. This is usually done automatically when opening WebLOAD IDE (only for Internet Explorer), but can also be done manually in the browser.

The procedures described here describe how to configure the proxy server for Internet Explorer, Netscape Navigator, and Mozilla Firefox 2.0. 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 IDE default port (8080) for another application, you may designate an alternate port number (see Setting the Proxy Options (on page 116)).

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 IDE (see Starting WebLOAD IDE (on page 44)).
2. Locate the Proxy Port number in the Record Options dialog box - Proxy Options tab. Usually this port number is 8080. (See Setting the Proxy Options (on page 116)).
3. Determine if your organization has a Proxy Server that must be used to access the Internet when you record Agendas.
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 Record Options dialog box - Proxy Options tab, go to step 6.
   c. If the proxy port number is the proxy port number found in the Record Options dialog box - Proxy Options tab, go to step 7.
5. If your organization does not use a Proxy Server, go to step 7.
6. Configure your organization proxy as the secondary proxy in WebLOAD IDE. To do so, complete the following steps:
   a. Open WebLOAD IDE.
   b. Select Tools | Record Options and then select the Proxy Options tab.
   c. Select the Use Secondary Proxy option.
   d. In the Secondary Proxy Name field, type the name of your organization's proxy.
   e. In the Secondary Proxy Port field, type the port number of your organization's proxy.
   f. 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 `loathsome`.
12. In the Port field, type the proxy port number found in the Record Options dialog box - Proxy Options tab.
13. Be sure 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 IDE (see Starting WebLOAD IDE (on page 44)).
2. Locate the Proxy Port number in the Record Options dialog box - Proxy Options tab. Usually this port number is 8080. (See Setting the Proxy Options (on page 116)).
3. Determine if your organization has a Proxy Server that must be used to access the Internet when you record Agendas.
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 Record Options dialog box - Proxy Options tab, go to step 6.
   c. If the proxy port number is the proxy port number found in the Record Options dialog box - Proxy Options tab, go to step 7.
5. If your organization does not use a Proxy Server, go to step 7.
6. Configure your organization proxy as the secondary proxy in the WebLOAD IDE. To do so, complete the following steps:
   a. Open WebLOAD IDE.
   b. Select Tools | Record Options and then select the Proxy Options tab.
   c. Select the Use Secondary Proxy option.
   d. In the Secondary Proxy Name field, type the name of your organization's proxy.
   e. In the Secondary Proxy Port field, type the port number of your organization's proxy.
   f. 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. Next, 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 Record Options dialog box - Proxy Options tab.
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 2.0

To configure the proxy value in Mozilla Firefox 2.0:

1. Open WebLOAD IDE (see Starting WebLOAD IDE (on page 44)).
2. Locate the Proxy Port number in the Record Options dialog box - Proxy Options tab. Usually this port number is 8080. (See Setting the Proxy Options (on page 116)).
3. Determine if your organization has a Proxy Server that must be used to access the Internet when you record Agendas.
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 Record Options dialog box - Proxy Options tab, go to step 6.
   c. If the proxy port number is the proxy port number found in the Record Options dialog box - Proxy Options tab, go to step 7.
5. If your organization does not use a Proxy Server, go to step 7.
6. Configure your organization proxy as the secondary proxy in the WebLOAD IDE. To do so, complete the following steps:
   a. Open WebLOAD IDE.

b. Select Tools | Record Options and then select the Proxy Options tab.

c. Select the Use Secondary Proxy option.

d. In the Secondary Proxy Name field, type the name of your organization's proxy.

e. In the Secondary Proxy Port field, type the port number of your organization's proxy.

f. 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 `proxy`.
13. In the HTTP Proxy Port field, type the proxy port number found in the Record Options dialog box - Proxy Options tab.
14. Select the Use this proxy for all protocols checkbox.
15. Click OK.

    You are finished configuring your proxy value.

Configuring the Proxy Value for Your Browser

C H A P T E R 4

WebLOAD IDE Quick Start

Welcome to WebLOAD IDE, the load testing tool that helps you quickly and easily test the functionality of your application under load, and serves as the recorder for RadView's test suite, TestView. Using an intuitive visual interface, WebLOAD IDE helps you create your own test scripts, or Agendas, to automatically test your Web based applications.

WebLOAD IDE'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 IDE.

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

In This Chapter

Getting Started

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

You will be working with an Agenda, or test script. The basic steps are:

1. Recording your Agenda - describes the steps in recording a basic Agenda (see Creating an Agenda (on page 28)).
2. Editing your Agenda - explains how to edit and modify your script, insert new items into your Agenda, and parameterize form data to create data driven tests (see Editing Your Agenda (on page 34)).
3. Running and debugging your Agenda - explains run and debug your Agenda (see Running and Debugging Your Agenda (on page 37)).

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

Creating an Agenda

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

1. Start WebLOAD IDE by selecting Start | All Programs | TestView | WebLOAD IDE.

   WebLOAD IDE opens.

2. Select Create a new project, and click OK.

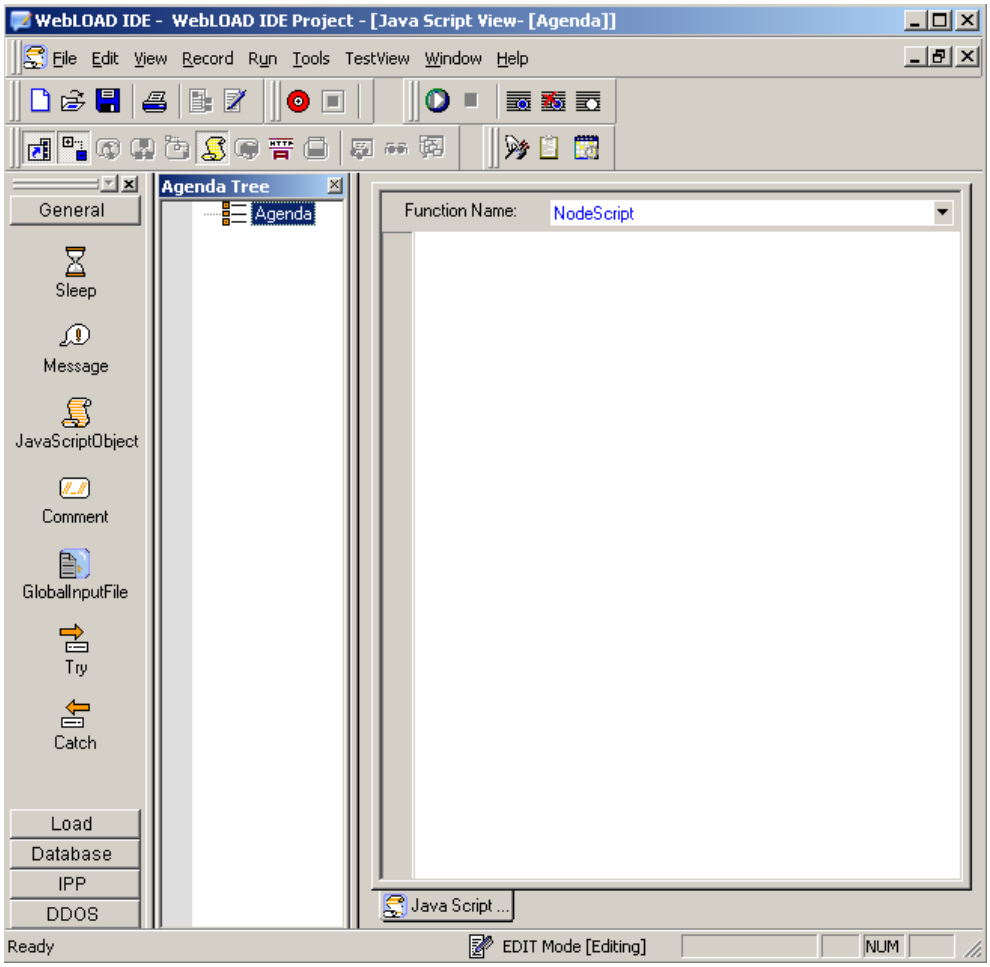
   The WebLOAD IDE main window opens in Editing Mode, for you to begin creating your Agenda.

   When the WebLOAD IDE main window first opens, it opens in Visual Editing Mode. In this mode, there are two active panes: the Agenda Tree and the JavaScript view pane.

   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 Agenda Tree.

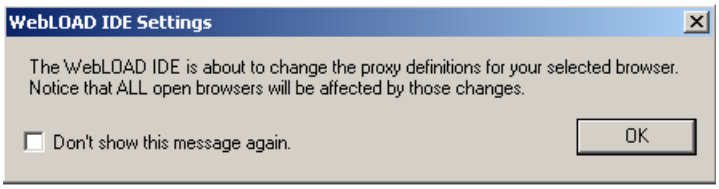
Each node in the Agenda Tree is actually a visual representation of JavaScript code. You can view the contents of the nodes in the JavaScript view pane.

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



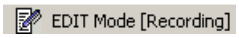
3. In the main window, in Editing Mode, click the Start Record toolbar button to begin recording.

   The following message appears.



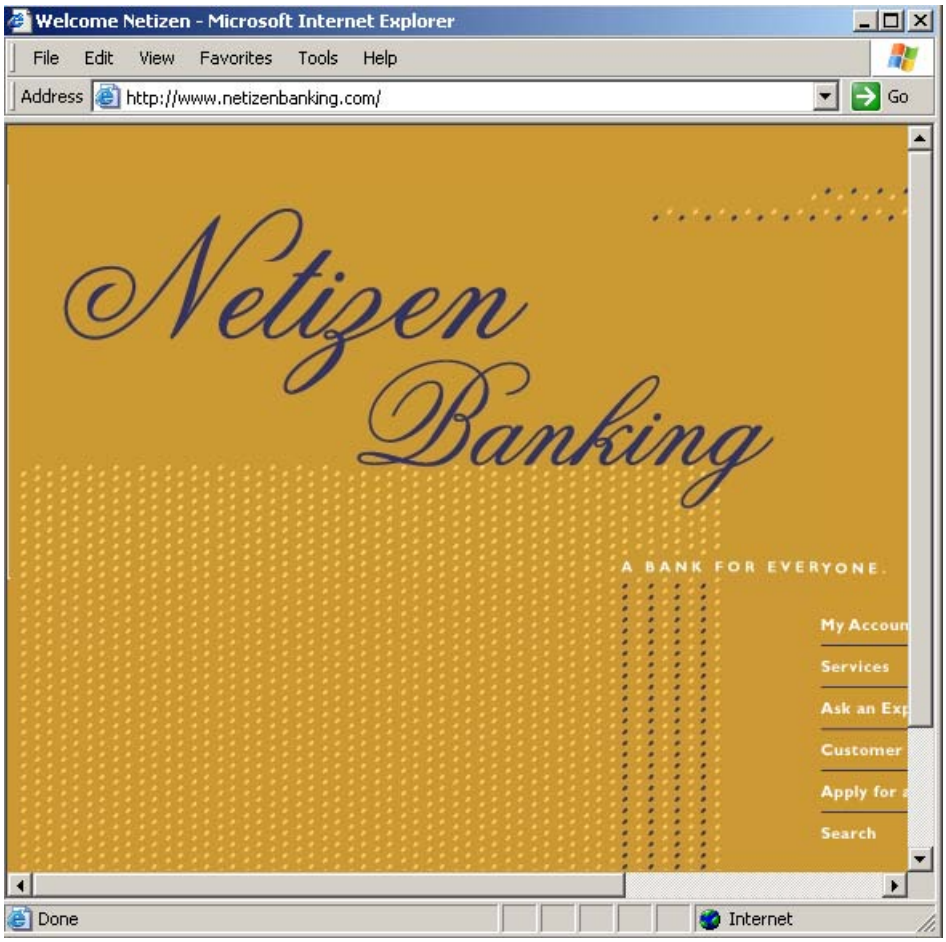
4. Click OK.

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



   A blank browser window opens.

5. In the address bar, enter the Web address www.netizenbanking.com to go to the RadView Software test site.



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

   For example:

   a. Click My Account on the home page.

   b. Click the Click Here to Login link.

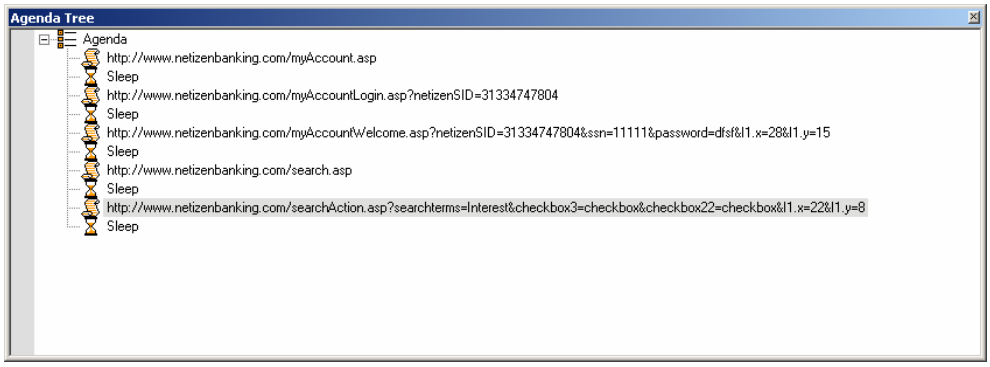
   c. Enter any set of numbers as a fictional social security number, enter a dummy password, and click Submit.

   d. Click the Search link to the right.

   e. Enter a keyword such as Interest to search for in the search field and select the checkboxes Home Mortgage and Business Banking.

   f. Click Search.

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



7. Click the Stop Record toolbar button in WebLOAD IDE to stop the recording.
8. Click the Save toolbar button or click File | Save As to save your Agenda.
9. Type in Netizen Banking for the name of the Agenda in the Save As dialog box and click Save.

   The Agenda is saved with the extension *.wlp.

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

Viewing Your Agenda

You can view the recorded Agenda in two views:

- JavaScript View

  When the WebLOAD IDE main window first opens, it opens in Visual Editing Mode. In this mode, there are two active panes: the Agenda Tree and the JavaScript view pane.

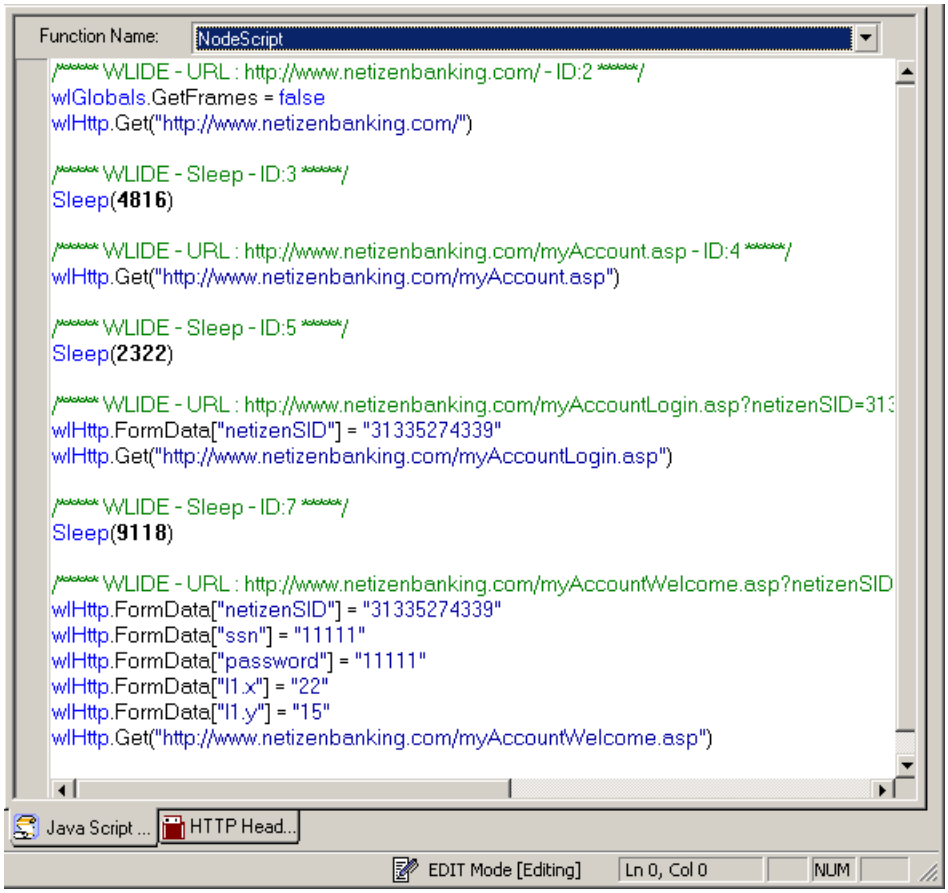
  When recording, your interactions with your Web application are captured, recorded, and presented graphically in the Agenda Tree.

  Each node in the Agenda Tree is actually a visual representation of JavaScript code. You can view the contents of the nodes in the 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 Agenda as a whole.

- View the code for different sections in the Agenda, by clicking the Agenda root node in the Agenda Tree and selecting a section from the Function Name list at the top of the JavaScript view pane.

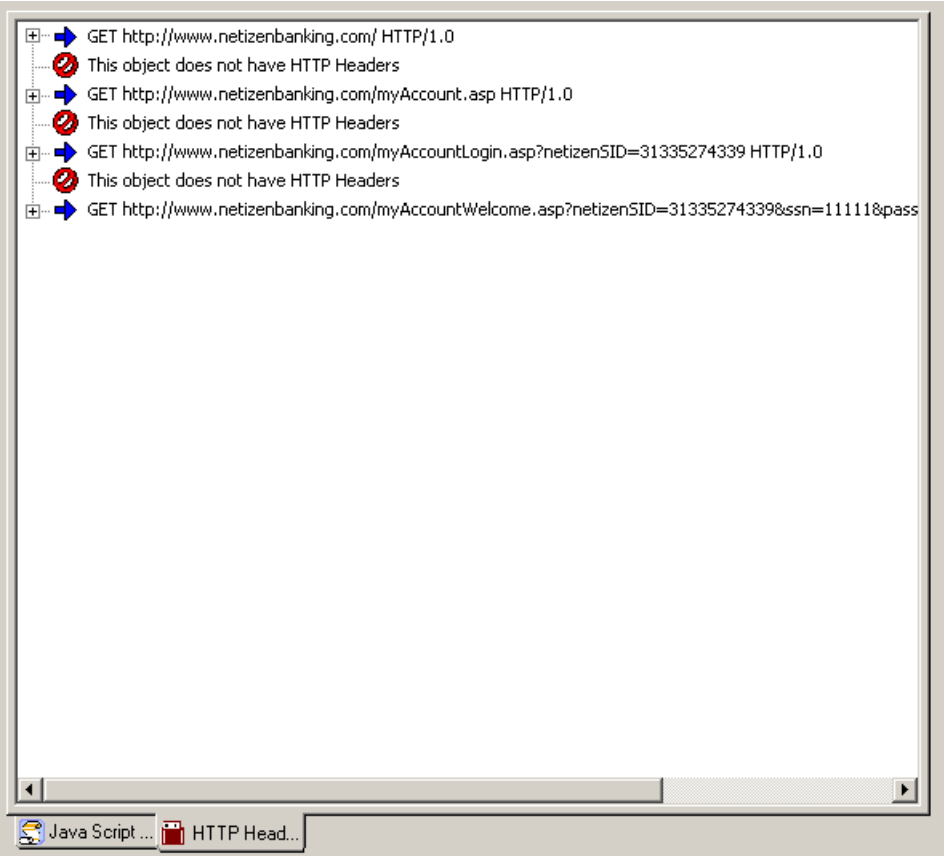


♦ HTTP Headers View

Each node in the Agenda Tree also has a visual representation of response headers. These response headers were received when the Agenda 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 Agenda as a whole.



Editing Your Agenda

To edit your Agenda, you can do the following:

- Edit the runtime settings using the Default and Current Project Options.
- Toggle between Visual Editing mode and JavaScript Editing mode.
- In Visual Editing mode, you can edit the Agenda Tree:
  - Drag-and-drop items from the WebLOAD IDE toolbox into the Agenda Tree.
  - Right-click to insert new items.
- In JavaScript Editing mode:
  - 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 IDE and is the connection between the Agenda 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.

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

Note: For complete reference information on all JavaScript objects, variables, and functions used in WebLOAD IDE Agendas, see the *TestView JavaScript Reference Manual*.

Toggling Between Edit Modes

You can toggle between Visual Editing mode and JavaScript Editing mode.

To toggle between Edit Modes:

- Select Edit | Start Visual Editing or click the Visual Editing  toolbar button

  -OR-

  Select Edit | Start Java Script Editing or click the Java Script Editing  toolbar button.

Basic Editing Techniques

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

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

- Drag and drop items into your Agenda Tree.
- Right-click for a menu of available options.
- Use the Insert menu.

Drag and Drop

WebLOAD IDE allows you to drag Agenda items from the WebLOAD IDE toolbox and drop them into your Agenda Tree.

To drag and drop items into your Agenda:

1. Place the mouse pointer over the item in the WebLOAD IDE toolbox that you want to add to your Agenda, such as a Message.
2. Press and hold the left mouse button.
3. Drag the item into the Agenda Tree, and place the mouse pointer at the step in the Agenda 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 Agenda Tree.
5. Click the Agenda item in the Agenda Tree to view and/or edit the JavaScript code in the JavaScript view pane.

Right-Click Menus

Throughout WebLOAD IDE, 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 Agenda item in the Agenda Tree to display a menu.

To insert a new item:

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

Adding Agenda Items

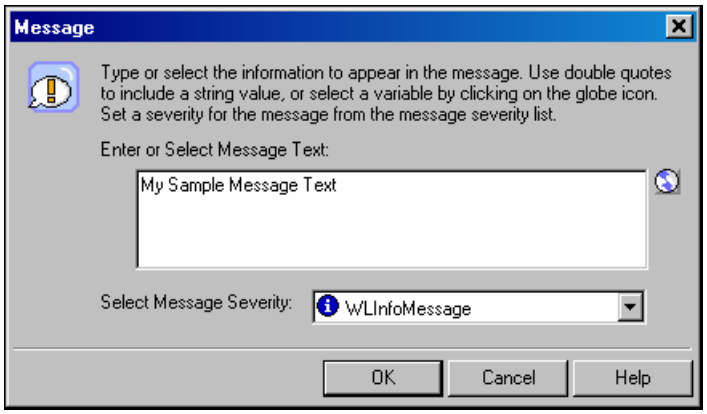
You can drag and drop an item, such as Message, from the WebLOAD IDE toolbox. For the list of toolboxes, see The WebLOAD IDE Toolbox Items (on page 149).

In the following instructions, adding a Message is used as an example.

To add a Message Agenda item:

1. Place the mouse pointer over the Message icon in the WebLOAD IDE toolbox.
2. Press and hold the left mouse button.
3. Drag the Message item into the Agenda, 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.



5. Enter the text you want to appear in the message or click the globe icon to add variable to the message text.

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

6. 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)
7. Click OK.

   The Message item appears in the Agenda Tree.

Running and Debugging Your Agenda

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

Running Your Agenda

To run your Agenda:

1. Select Run | Run or click the Run Test  toolbar button.

   As the Agenda is running:

   - A yellow arrow points to the node being executed in the Agenda Tree.
   - If the JavaScript View tab is open, you will also see the yellow arrow pointing to the script.
   - If the Browser 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: Save Changes to WebLOAD IDE Project, click Yes and enter a file name to save your Agenda 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 Agenda

WebLOAD IDE provides an integrated debugger with a variety of tools to help locate bugs in your Agenda. The debugger provides special menus, windows, dialog boxes, and grids of fields for debugging. You can pause the debugger and trigger WebLOAD IDE to wait for user input before proceeding with running the Agenda. In the Agenda, you can set breakpoints and step into / over / out.

To debug your Agenda:

1. Select Run | Step Into or click the Run  toolbar button

   -Or-

   Use the Step Into  toolbar button to run the Agenda step-by-step

   -Or-

Add breakpoints by clicking the Toggle Breakpoint  toolbar button and then clicking the Run  toolbar button to run the Agenda.

2. At the prompt: Save session file, click Yes and enter a file name to save your session.

Debugging Using the Watch Window

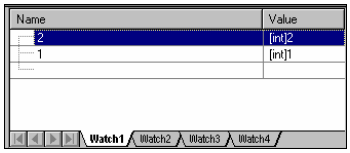
You can use the Watch window to specify variables and expressions that you want to watch while debugging your program.

To debug using the Watch window:

1. Start debugging.
2. In the main window, click View | Debug Windows | Watch

   -Or-

   Click the Watch Window  toolbar button.
3. To view a variable or expression in the Watch window, click View | Debug Windows | Watch, or click the Watch Window  toolbar button 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.

Debugging Using the Variables Window

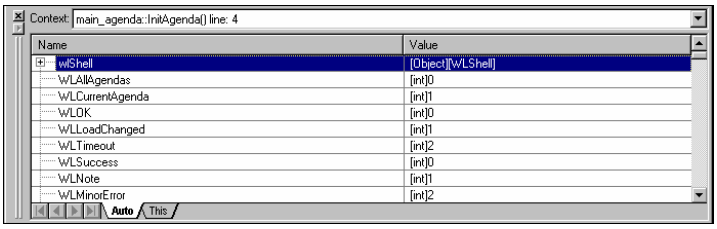
The Variables window provides quick access to variables that are important in the Agendas current context.

To debug using the Variables Window:

1. Start debugging.
2. In the main window, click View | Debug Windows | Variables

   -Or-

Click the Variables Window  toolbar button.



The window includes two tabs:

- Auto tab: 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.
- This tab: Displays the object pointed to by this.

Each tab 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.

In addition to the tabs, the Variables window has a Context box that displays the current scope of the variables displayed.

Debugging Using the Call Stack Window

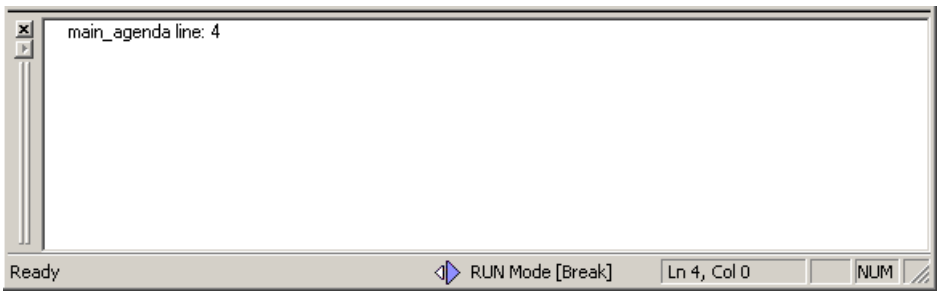
The Call Stack window lists the function calls that led to the current statement, with the current function on the top of the stack.

To debug using the Call Stack Window:

1. Start debugging.
2. In the main window, click View | Debug Windows | Call Stack

   -Or-

   Click the Call Stack  toolbar button.



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

C H A P T E R 5

Recording Agendas

This section provides instructions for recording Agendas with WebLOAD IDE.

In This Chapter

About Recording Agendas with WebLOAD IDE

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

As your actions are recorded, WebLOAD IDE displays them in the Agenda Tree, which is a tree hierarchy with visual indications of the information recorded. WebLOAD IDE records only HTTP protocol calls that place a load on the Application Being Tested (ABT). Activities that are not relevant to the Agenda, such as moving windows for a more comfortable display or opening another application, are not recorded.

This process creates the basic Agenda. You can then view the recorded Agenda as JavaScript code in the JavaScript view pane, revise the Agenda to test more objects in more detail, and run and debug the Agenda. For information on editing your Agenda, see Editing Agendas (on page 57). For information on running and debugging your Agenda, see Running and Debugging Agendas (on page 75).

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

Starting WebLOAD IDE

To start WebLOAD IDE:

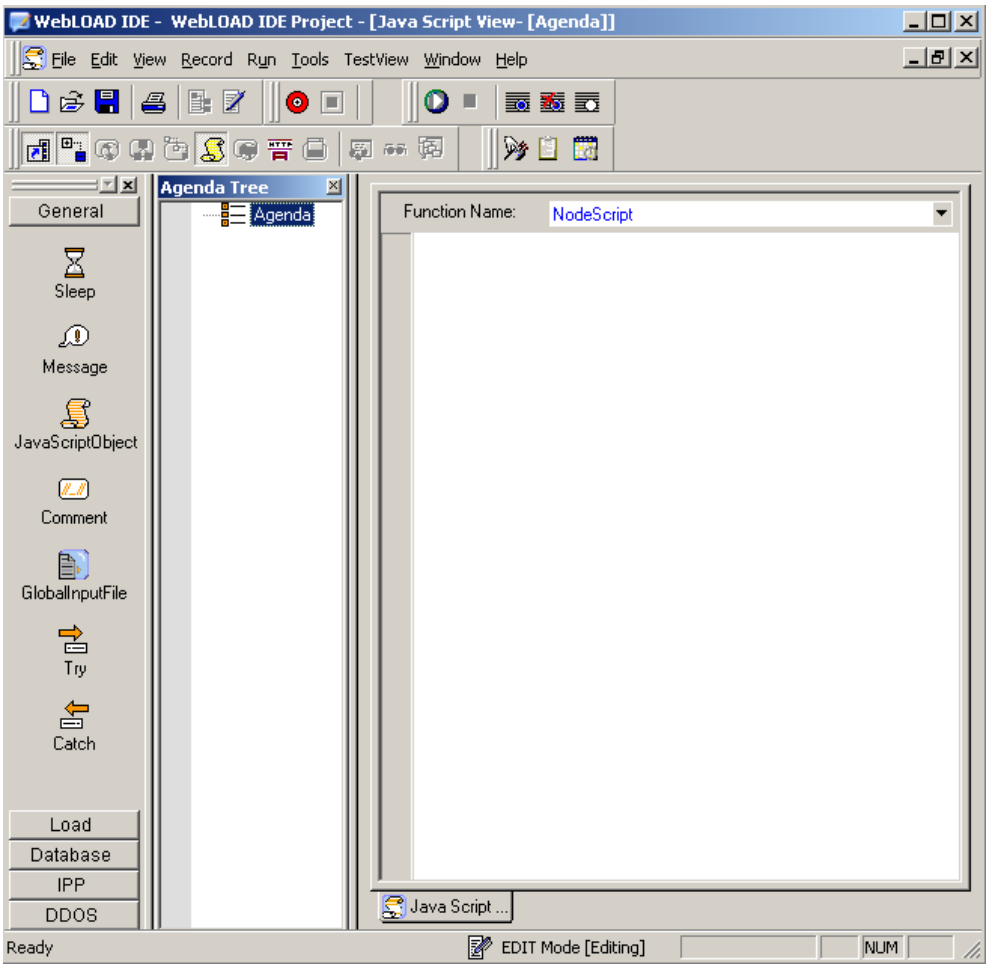
1. Select Start | All Programs | TestView | WebLOAD IDE.

   WebLOAD IDE opens.

2. Click an option.
   - Create a new project
   - Open an existing project and browse to the project.
   - Open a saved session and browse to the session.

3. Select or clear the Show this dialog every time WebLOAD IDE is started checkbox.
4. Click OK.

   The WebLOAD IDE main window opens in Editing Mode, for you to begin creating or editing your Agenda.



Recording an Agenda

You can either start working with WebLOAD IDE immediately, or you can configure the recording options first. For more information, see Configuring the Record Options (on page 101).

When you record an Agenda, WebLOAD IDE displays the Agenda being created in real time. You can watch WebLOAD IDE 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 Agenda. If you open an existing Agenda and start recording new Web activity, WebLOAD IDE also appends the new JavaScript commands to the end of the Agenda.

To record an Agenda:

1. Start WebLOAD IDE (see Starting WebLOAD IDE (on page 44))

   -Or-

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

   The WebLOAD IDE main window opens in Editing Mode, for you to begin recording your Agenda.

2. To create a new Agenda, click New

   -Or-

   Select File | New.

3. To open an existing Agenda:

   a. Click Open
      -Or-

      Select File | Open.

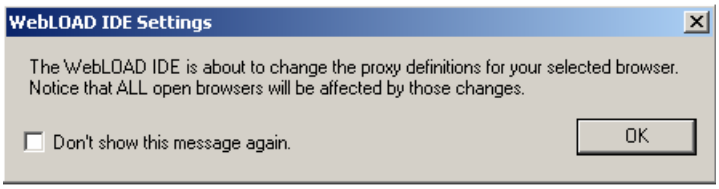
   b. Select a file.

4. Click Start Record

   -Or-

   Select Record | Start Record.

   The following message appears.

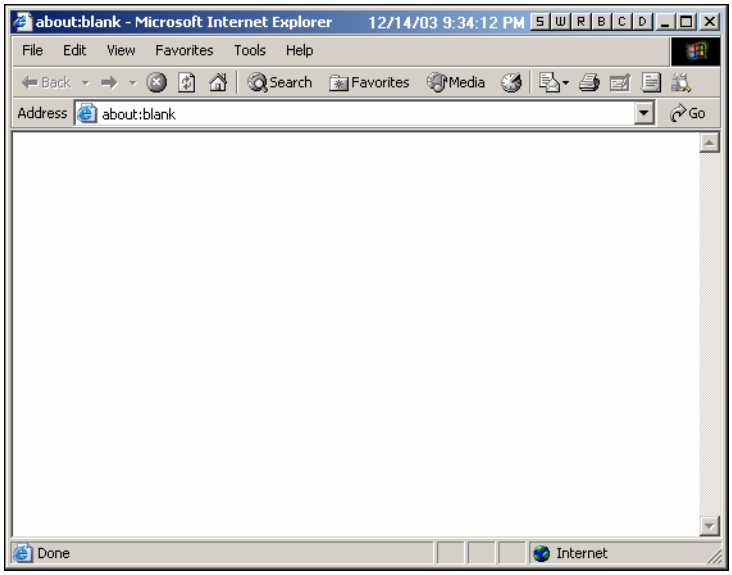


5. Click OK.

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

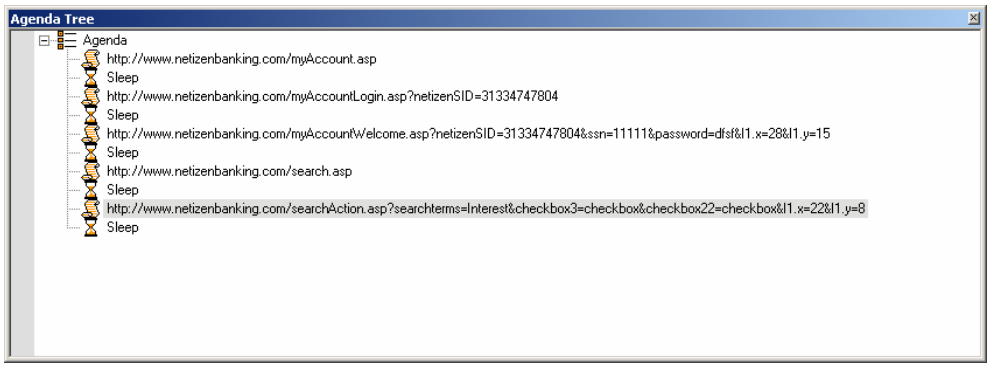
If this is the first time that you are recording after WebLOAD IDE was launched, the default browser opens automatically with its predefined home page. This enables you to start recording and then access a page. Thus the Get command will be the first command in the generated Agenda.

Note: WebLOAD IDE does not open a new browser window every time you start recording because WebLOAD IDE assumes that you will continue using the previously opened browser window and cannot know if you closed it or not.



6. In the Web browser window, access the Application Being Tested (ABT).
7. 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 IDE adds nodes to the Agenda as you work. Your actions are recorded and appear in the Agenda Tree as you navigate the site. (If you see more nodes in the Agenda Tree with different URLs, this may be traffic generated by browser plug-ins or extensions, for example, third-party toolbars).



   a. Click the JavaScript View toolbar button, 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 toolbar button to watch the response headers of the pages as they are being recorded.
8. When you are finished, select WebLOAD IDE.
9. Click Stop Record

   -Or-

   Select Record | Stop Record.

   WebLOAD IDE stops recording.
10. Click Save

    -Or-

    Select File | Save to save the Agenda.
11. In the File name field in the Save As dialog box, type a descriptive name for the Agenda, and then click Save.

    Your Agenda is saved with the file extension *.wlp.

12. Close the Browser window to work in WebLOAD IDE.

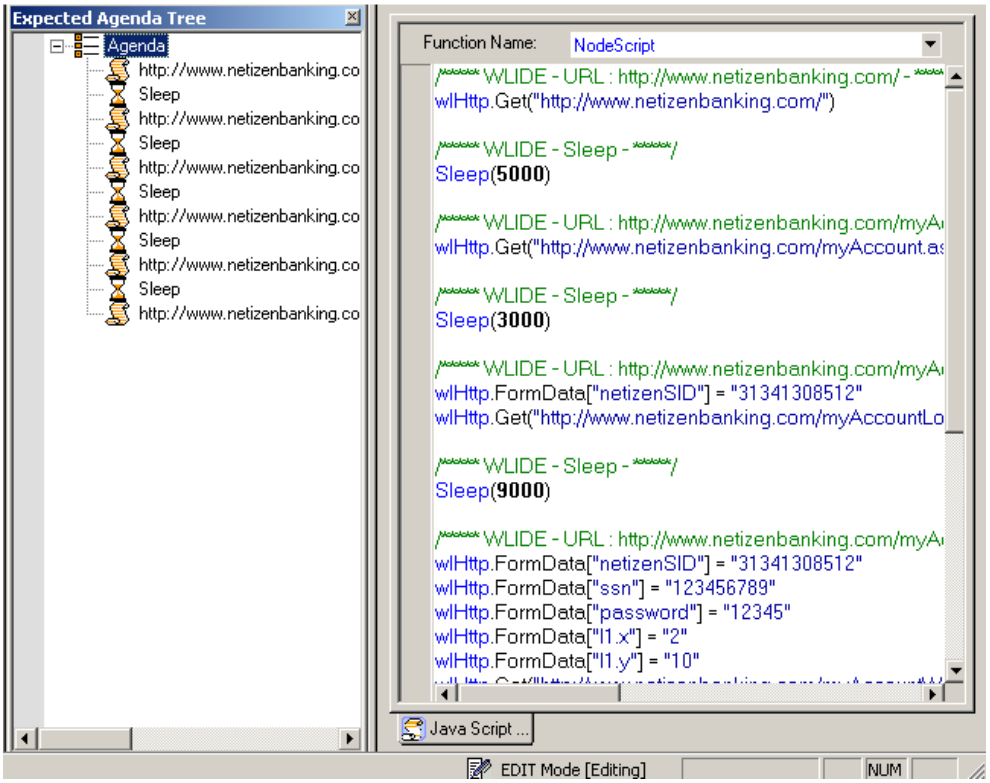
You can customize the Agenda in a variety of ways or you can run your Agenda as recorded. For information on editing your Agenda, see Editing Agendas (on page 57). For information on running your Agenda, see Running and Debugging Agendas (on page 75).

Note: If actions that you are interested in were not recorded, check the cache settings in your browser. WebLOAD IDE 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 in Your Browser (on page 20).

Viewing the Recorded Agenda

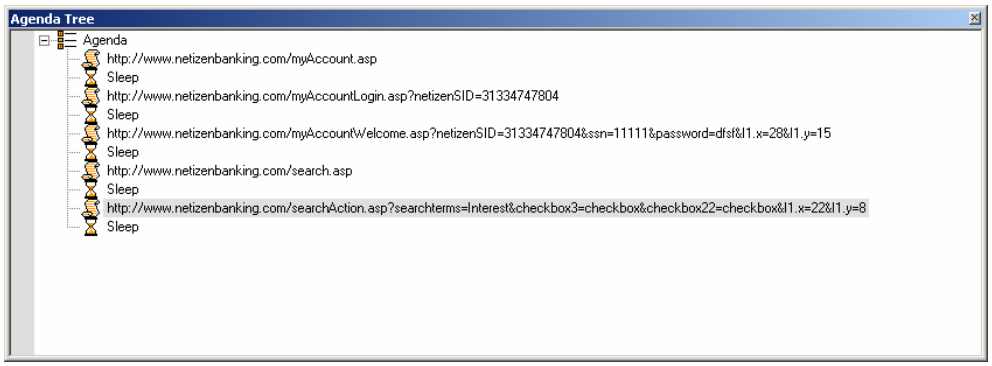
WebLOAD IDE creates Agendas by recording your actions as you interact with your Web application or Application Being Tested (ABT). Your recorded Agenda serves as a baseline, which is subsequently used in the WebLOAD environment to test the performance of your Web application.

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



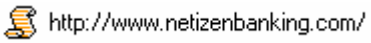
Viewing the Recorded Agenda in the Agenda Tree

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



WebLOAD IDE displays only the pages accessed and the Sleep time as nodes in the Agenda Tree.

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



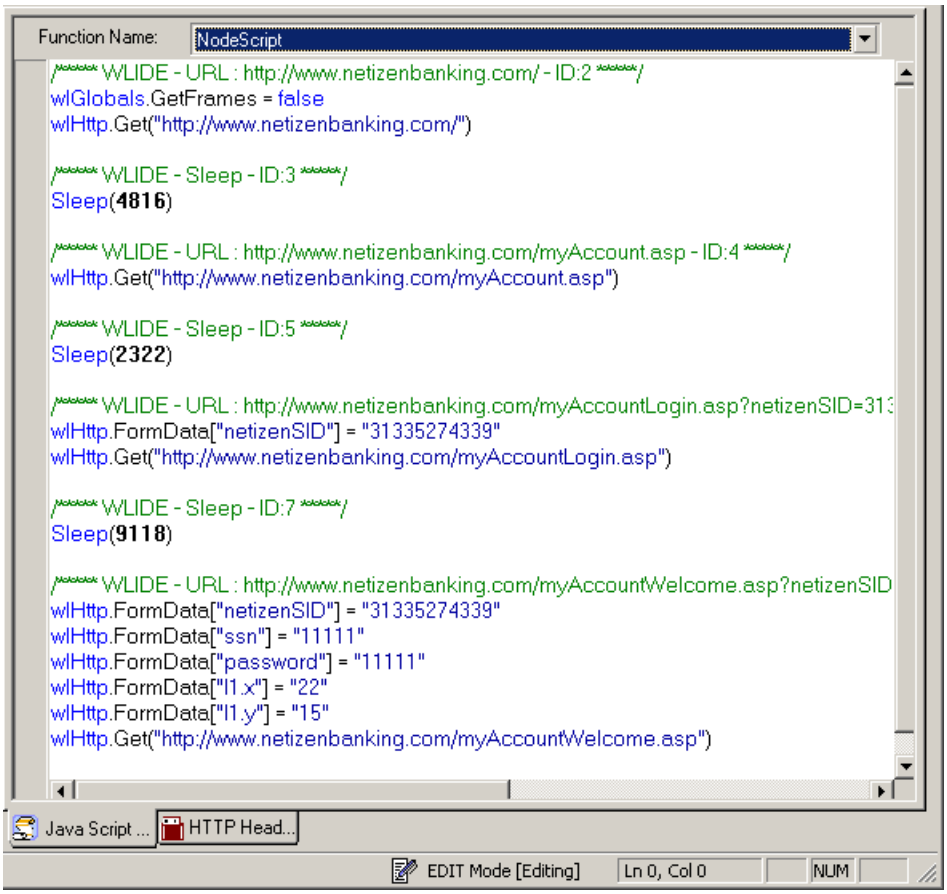
When you pause while navigating in the Web application, WebLOAD IDE inserts a Sleep node into the Agenda Tree. The Sleep node represents your thinking time.

After the Agenda is recorded, you can edit the Agenda (see Editing an Agenda in the Agenda Tree (on page 58)).

After your Agenda has been developed, you can run and debug it. While the Agenda is running, you can view it in the Agenda Tree (see Viewing the Execution Sequence in the Agenda Tree (on page 76)).

Viewing the Recorded Agenda in the JavaScript View Pane

Each node in the Agenda Tree is actually a visual representation of JavaScript code. You can view the contents of the nodes in the 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 Agenda as a whole.
- View the code for different sections in the Agenda, by clicking the Agenda root node in the Agenda 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. The ID number is automatically generated by WebLOAD IDE and is the connection between the Agenda node and the specific header.

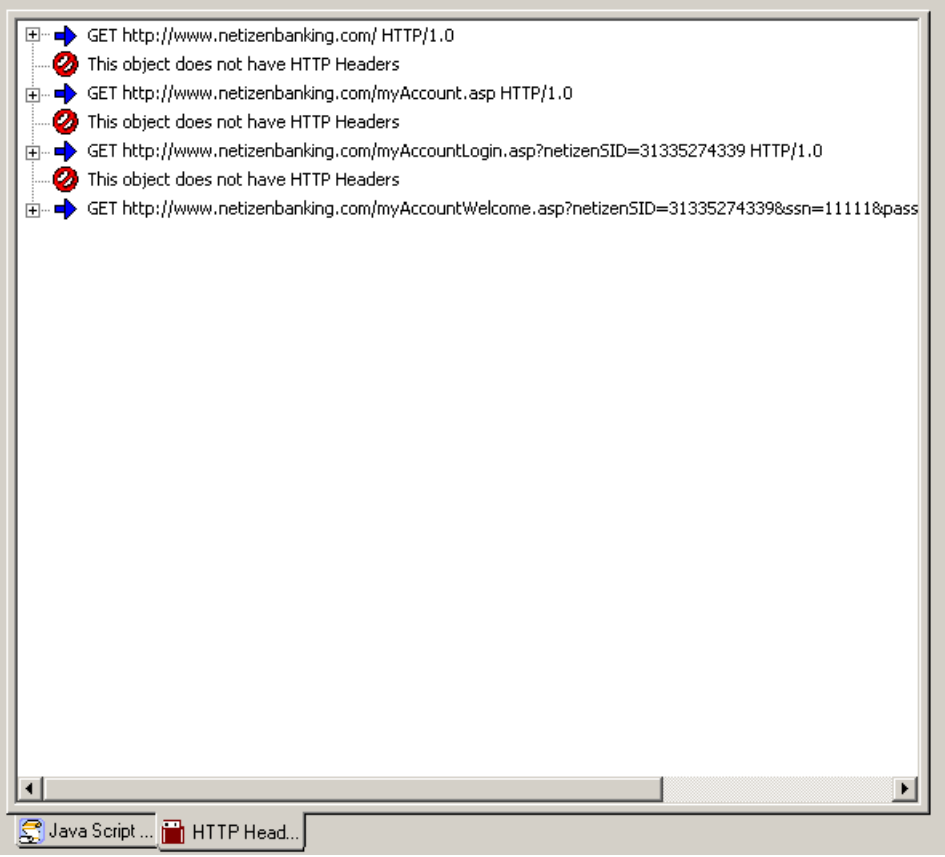
```
/***** WLIDE - URL : http://www.netizenbanking.com/myAccount.asp - ID:4 *****/
wlHttp.Get("http://www.netizenbanking.com/myAccount.asp")
```

After the Agenda is recorded, you can edit the Agenda (see Editing an Agenda in the JavaScript View Pane (on page 60)).

After your Agenda has been developed, you can run and debug it. While the Agenda is running, you can view it in the JavaScript View pane (see Viewing the Execution Sequence in the JavaScript View Pane (on page 77)).

Viewing the Recorded Agenda in the HTTP Headers View Pane

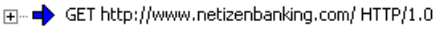
Each node in the Agenda Tree is also a visual representation of response headers. You can view the headers of the nodes in the HTTP Headers view pane.



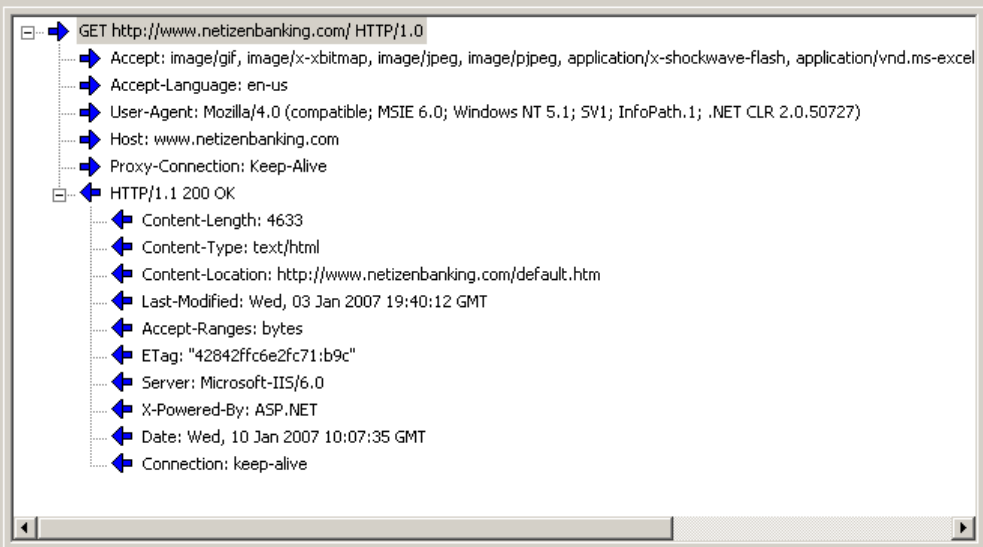
In the HTTP Headers view pane, you can do the following:

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

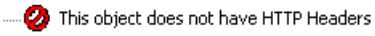
When you click on a node, you can view the header for that node.



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



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



After the Agenda is recorded, you can edit the Agenda (see Editing an Agenda in the JavaScript View Pane (on page 60)).

After your Agenda has been developed, you can run and debug it. While the Agenda is running, you can view it in the JavaScript View pane (see Viewing the Execution Sequence in the JavaScript View Pane (on page 77)).

Saving an Agenda

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

To save an Agenda:

1. Select File | Save or Save As

   -Or-

   Click Save  .

   The Save As dialog box appears.
2. Type the Agenda name in the File name field.
3. Click Save.

   Your Agenda is saved with the file extension *.wlp.

You may now run a test using the Agenda.

Saving Additional Project Information

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

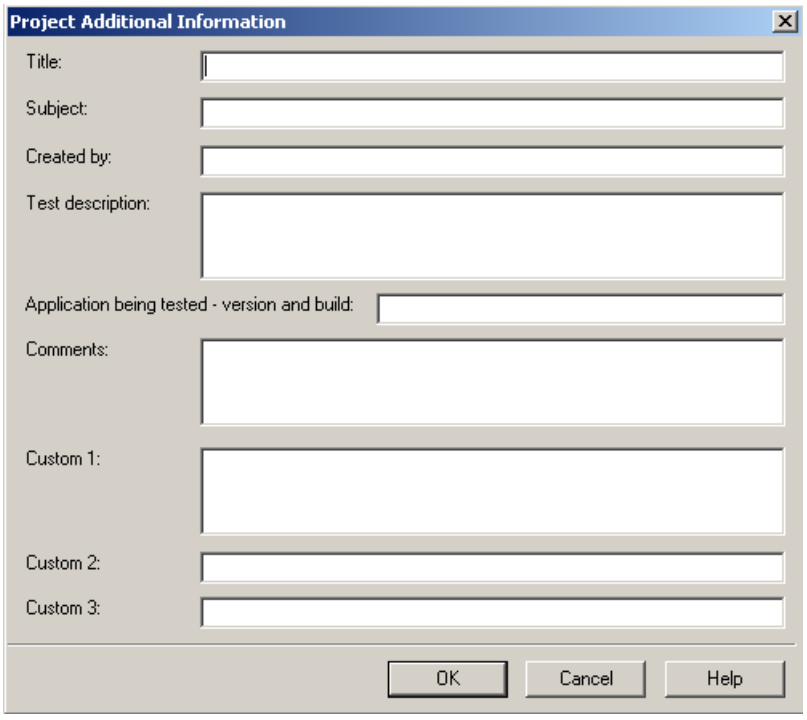
- A descriptive title
- The author name
- The subject of the test
- The application being tested
- Other important information about the project

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

To save additional information properties for the project:

1. Select File | Additional Information.

   The Project Additional Information dialog box opens.



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 Project Additional Information dialog box.

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.
Application being tested - version and build	Provides a space for you to type the name, version and build number of the application being tested.
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.

C H A P T E R 6

Editing Agendas

This section provides instructions for editing Agenda with WebLOAD IDE.

In This Chapter

About Editing Agendas with WebLOAD IDE

WebLOAD IDE is both flexible and extendable to fit all of your Agenda editing needs, from the most basic to the most advanced. On the simplest level, you use the WebLOAD IDE GUI to record your basic Agenda. You can then edit your Agenda to and 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 Agenda.

In the Agenda, each request and event is based on previous input, tying the entire Agenda into a whole, making many actions interdependent. Items such as JavaScript Objects, Comments, Messages and Sleeps can be added to the Agenda, but changing the sequence of items in effect means changing the sequence of activities, and may destroy the functionality of the Agenda. For more information on recording Agendas, see Recording Agendas (on page 43).

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

The following Agenda editing tools are discussed:

- Editing an Agenda in the Agenda Tree (on page 58)describes how to add Agenda items and JavaScript Objects, and edit an Agenda by right-clicking in the Agenda Tree.
- Editing an Agenda in the JavaScript View Pane (on page 60) describes how to use JavaScript objects to create Agenda scripts with the full functionality of JavaScript code programs. The WebLOAD IDE JavaScript Editor includes a set of context-sensitive prompts that help you code your Agenda more effectively.
- Editing your Agenda Using the WebLOAD IDE Toolbox Set (on page 71) describes how to use the WebLOAD IDE toolbox that contains drag-and-drop items to create a script with minimal coding.

Note: The technical support pages on our Web site contain the Agenda Center, which is a test script library. The library contains code fragments and samples that can help you work with WebLOAD IDE and WebLOAD. Each example contains a full description of the code fragment or file, with an explanation of how the code works. We also include tips and suggestions and downloadable copies of the files, to get you up and running faster.

To view the Agenda Center, navigate to http://www.RadView.com/support/index.asp (http://www.radview.com/support/index.asp), login as a registered user, and go to the Test Script Library.

Editing an Agenda in the Agenda Tree

This section describes how to edit an Agenda in the Agenda Tree.

Note: You must be in Visual Editing mode.

Adding Agenda Items and JavaScript Objects to an Agenda

WebLOAD IDE contains shortcuts to frequently performed actions. This section describes how to place Agenda items and JavaScript Objects from the Insert menu into an Agenda. For guidelines for replacing the placeholder variables with your own, see Guidelines for Editing JavaScript Code (on page 69).

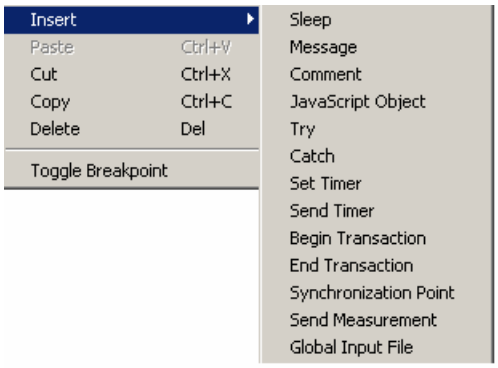
To add items and JavaScript Objects to an Agenda:

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

   A pop-up menu appears.

4. From the pop-up menu, click Insert, and then click Agenda Items.

   The following list of shortcuts appears.



5. Select an Agenda item or JavaScript Object.

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

   The Agenda Items and JavaScript Objects that you can insert are also available through the WebLOAD IDE toolbox, see Editing your Agenda Using the WebLOAD IDE Toolbox Set (on page 71).

Editing an Agenda by Right-Clicking in the Agenda Tree

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

To right-click menus in Edit mode:

1. In the main window, select File | Open and open the Agenda you want to edit.
2. Make sure that you are in Visual Editing mode.
3. In the Agenda Tree, right-click the Agenda root node or right-click in the tree.

   A pop-up menu appears.

   The following table describes the menu options:

Right-Click Menu Option	Purpose
Insert	Insert an Agenda Item or JavaScript Object into the Agenda (see Adding Agenda Items and JavaScript Objects to an Agenda (on page 58)). The Agenda items and JavaScript Objects that you can insert are also available through the WebLOAD IDE toolbox, described in Editing your Agenda Using the WebLOAD IDE Toolbox Set (on page 71).
Cut	Cut the Agenda item from the tree to paste elsewhere.
Copy	Copy the Agenda item from the tree to paste elsewhere.
Paste	Paste the Agenda item you cut or copied after the current Agenda item. Note: If you copied an Agenda item, you can paste more than once. Each time you paste, the node ID automatically changes. If you cut an Agenda item, you can paste only once, and the node ID does not change.
Delete	Delete the Agenda item from the tree.
Current Project Options	Display the Current Project Options dialog box. Only available at the Agenda level. For more information, see Configuring the Default and Current Project Options (on page 120).
Toggle Breakpoint	Add or remove a breakpoint at the selected Agenda item in the Agenda Tree. For more information, see Setting Breakpoints (on page 82).

Editing an Agenda in the JavaScript View Pane

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

Editing the JavaScript Code for an Agenda Item

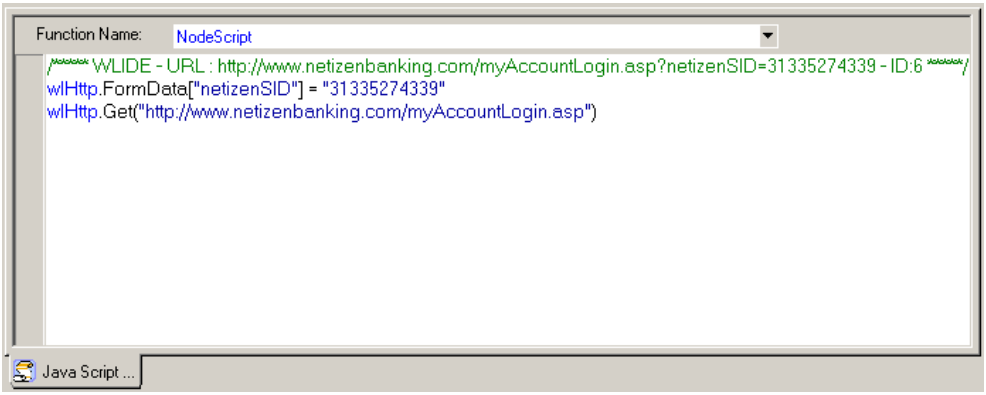
You can edit the JavaScript code generated by WebLOAD IDE for any item in the Agenda.

Note: When you select the Agenda root node, the entire Agenda appears in the JavaScript View pane as read only. To edit the entire Agenda, see Using the JavaScript Editor (on page 63).

To edit the JavaScript code for an Agenda item:

1. In the main window, select File | Open and open the Agenda you want to edit.
2. Make sure that you are in Visual Editing mode.
3. Click the JavaScript View  toolbar button to open the JavaScript View pane.
4. Select the item in the Agenda Tree.

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



5. Edit the Agenda (see Editing the JavaScript Code (on page 66)).

Important: The ID number is automatically generated by WebLOAD IDE and is the connection between the Agenda 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

An Agenda 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 Agenda as a whole file when in JavaScript Editing mode (see Using the JavaScript Editor (on page 63)).

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 Agenda. By default, WebLOAD IDE calls the `InitAgenda(), InitClient(), TerminateClient(), and TerminateAgenda()` functions for each Agenda.

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 IDE 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 IDE 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.

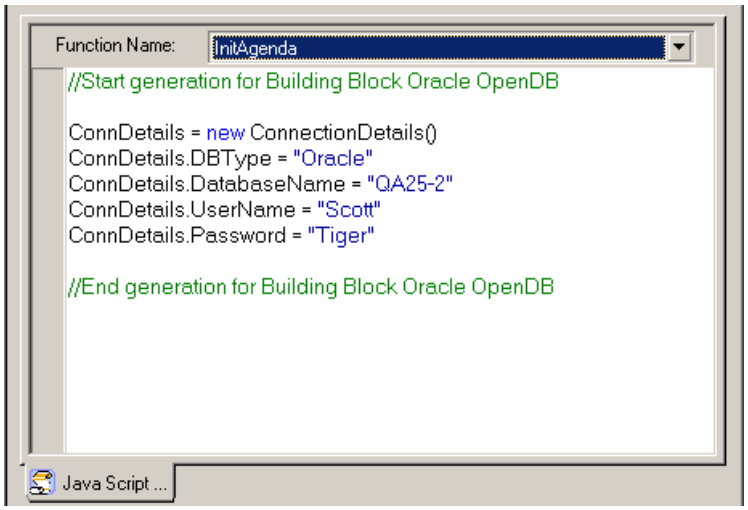
To edit the JavaScript code for functions:

1. In the main window, select File | Open and open the Agenda you want to edit.
2. Make sure that you are in Visual Editing mode.
3. Click the JavaScript View  toolbar button to open the JavaScript View pane.
4. Select the Agenda item in the Agenda Tree.

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

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.



6. Type the JavaScript code to include in the `InitClient, InitAgenda, TerminateClient,` or `TerminateAgenda` (see Editing the JavaScript Code (on page 66)).

   For guidelines for replacing the placeholder variables with your own, see Guidelines for Editing JavaScript Code (on page 69).

Note: You cannot add a WebLOAD IDE 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 Agendas are written in JavaScript. The JavaScript code within an Agenda is created from the actions you record and the verification tests you place in the Agenda. You can add JavaScript Objects to your recorded Agenda, allowing you to add additional written code directly to your Agenda. The JavaScript Editor is both a viewer and an editor for adding and editing JavaScript code in the Agenda.

WebLOAD IDE provides the following features for manually editing an Agenda:

- Import JavaScript Files

  WebLOAD IDE enables you to import JavaScript files into your Agenda.

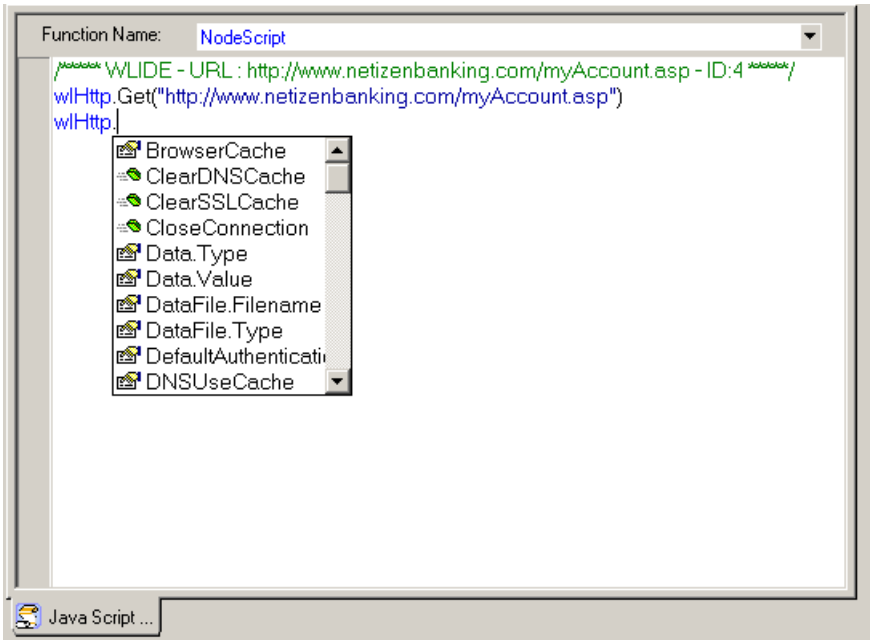
- WebLOAD IDE Protocol Block

  WebLOAD IDE enables you to add code to your Agenda which is then represented visually in the Agenda Tree.

- An IntelliSense Editor mode for the JavaScript View pane.

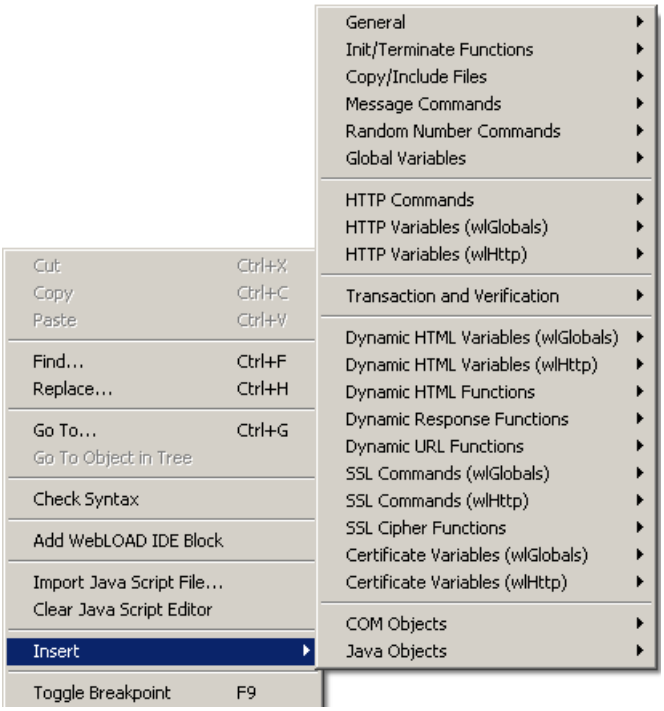
  Add new lines of code to your Agenda or edit existing JavaScript functions through the IntelliSense Editor mode of the JavaScript View pane. The IntelliSense Editor helps you write the JavaScript code for a new function by formatting new code and prompting with suggestions and descriptions of appropriate code choices and syntax as programs are being written. For example, in the following figure the IntelliSense Editor displays a drop-down list of available properties and objects for the `wlHttp` object being added to the program, with a pop-up box describing the highlighted method in the list.

  For more information, see the *TestView Programmer's Guide.*



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

  You can who choose to program your own JavaScript Object code within your Agenda and take advantage of the WebLOAD IDE GUI to simplify your programming efforts. Rather than manually type out the code for each command, with the risk of making a mistake, even a trivial typographical error, and adding invalid code to the Agenda 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 IDE 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.



- A Syntax Checker that checks the syntax of the code in your Agenda 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 IDE desktop, select Tools | Check Syntax to check the syntax of the code in your Agenda file.

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

Editing the JavaScript Code

To edit the JavaScript code for the Agenda:

1. In the main window, select File | Open and open the Agenda you want to edit.
2. Click the JavaScript Editing  toolbar button to open the Agenda in JavaScript Editing mode.

   The entire Agenda appears.
3. Position the cursor where you want to edit the JavaScript 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 an Agenda (on page 68)).
6. Import a JavaScript file.

   a. Right-click in the Agenda.

   b. Click Import JavaScript File from the pop-up menu.
   The JavaScript code is added to the Agenda.
7. Add a WebLOAD IDE protocol block from the pop-up menu (see Adding WebLOAD IDE Protocol Blocks (on page 66)).
8. Perform a syntax check:

   a. Right-click in the Agenda.

   b. Click Check Syntax from the pop-up menu.
   WebLOAD IDE performs a syntax check and displays the errors.
9. Toggle a breakpoint (for more information, see Setting Breakpoints (on page 82)).

Adding WebLOAD IDE Protocol Blocks

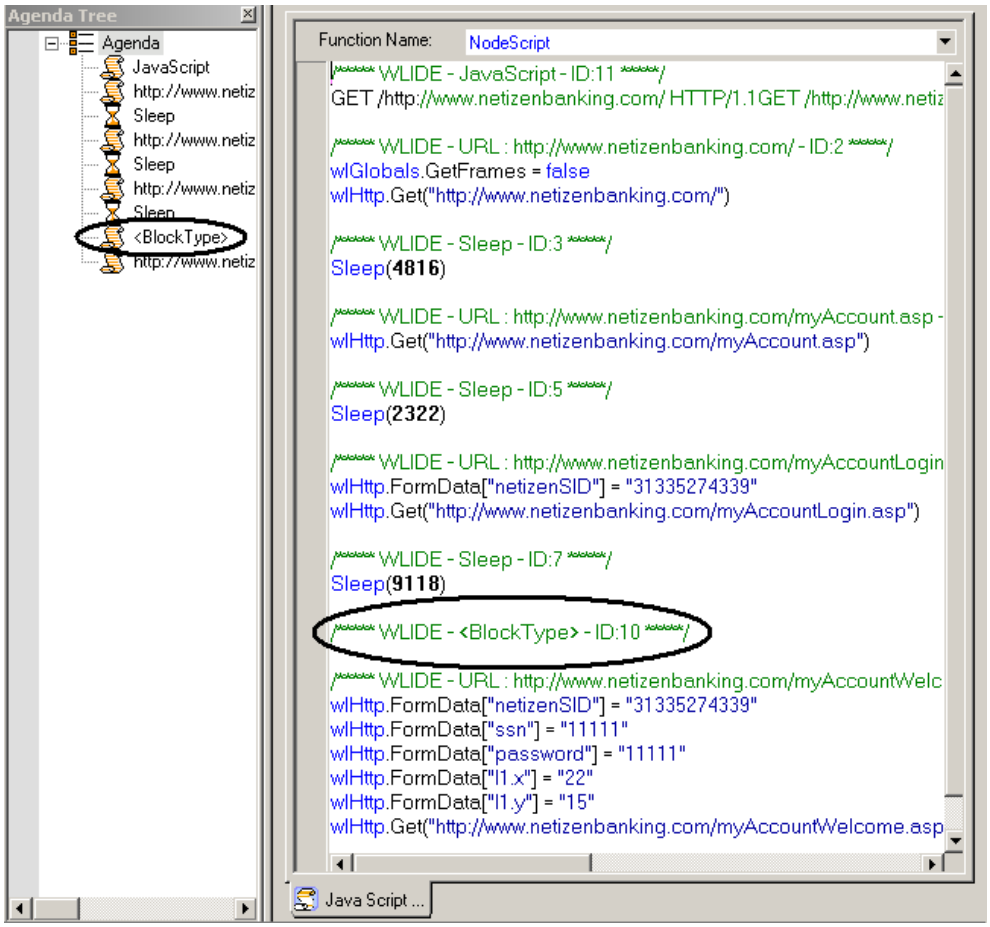
To add WebLOAD IDE Protocol Blocks to an Agenda:

1. In the main window, select File | Open and open the Agenda you want to edit.
2. In the JavaScript View pane, position the cursor where you want to place the WebLOAD IDE protocol block.
3. Select the Edit | Insert menu, and click WebLOAD IDE Block

   -Or-

   Right-click in the Agenda, and click Add WebLOAD IDE Block from the pop-up menu.

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

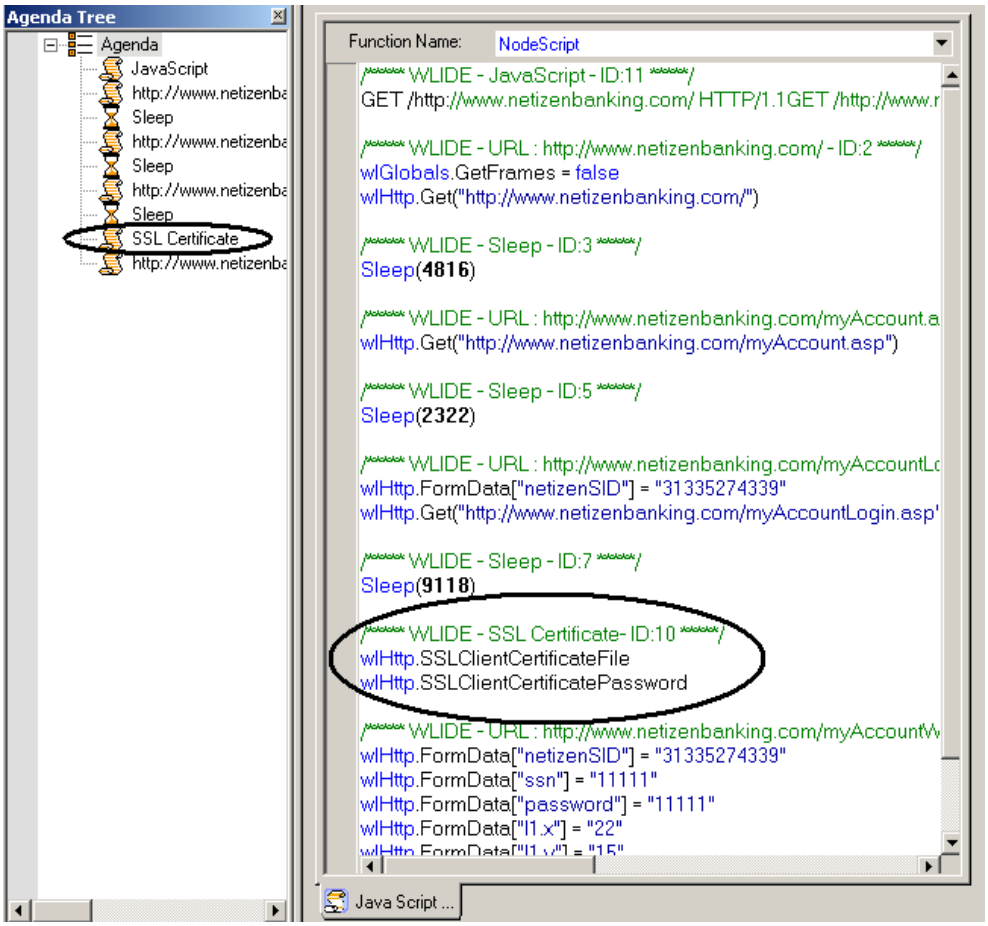


4. Replace the placeholder `<Block Type>` with a description.

   For example: Replace `<Block Type>` with `SSL Certificate`.

5. Add the JavaScript code after the WebLOAD IDE protocol block header.

   The code is added to the Agenda.



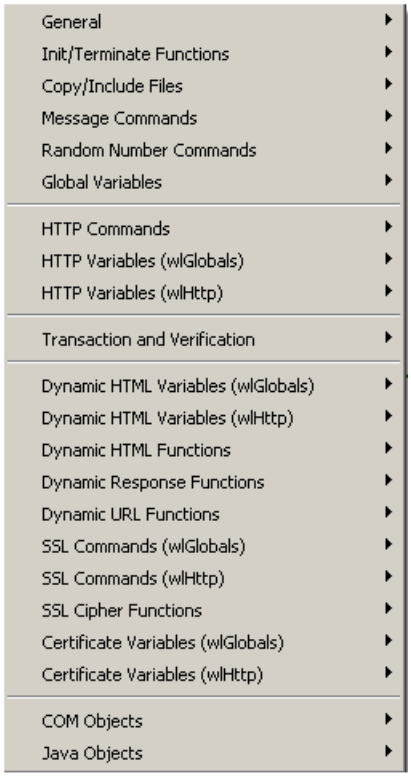
Adding Commands and Functions to an Agenda

WebLOAD IDE contains shortcuts to frequently performed actions. This section describes how to place Commands, and functions from the Insert menu in an Agenda. For guidelines for replacing the placeholder variables with your own, see Guidelines for Editing JavaScript Code (on page 69).

To add commands and functions to an Agenda:

1. In the main window, select File | Open and open the Agenda 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 Agenda and click Insert.

   The list of shortcuts appears.



4. Select a command or function.

   The command or function selected is inserted on a new line in the Agenda, 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 an Agenda 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 IDE protocol block, replace the placeholder `<Block Type>` with a description.

  For example:

  ```
  <Block Type>
  ```

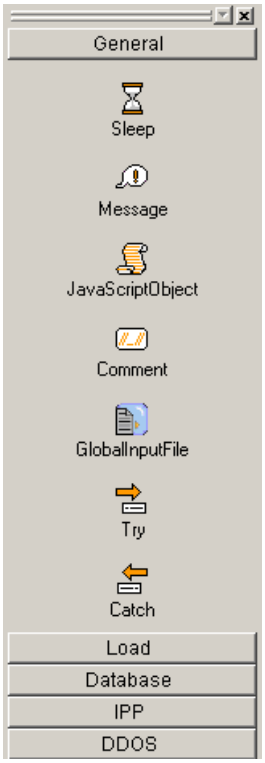
  can be replaced with:

  ```
  SSL Certificate
  ```

See the *TestView Programmer's Guide* for more information.

Editing your Agenda Using the WebLOAD IDE Toolbox Set

The WebLOAD IDE provides a set of objects, such as Sleep, that you can drag and drop to add Agenda items in the Agenda Tree. The WebLOAD IDE bar is referred to as the toolbox.



Use the WebLOAD IDE toolboxes to add the following items to your Agenda:

- General objects, such as Message or Sleep timers. These objects are used in all test Agendas, run in both WebLOAD IDE and WebLOAD. General toolbox tools are described in The WebLOAD IDE General Toolbox (on page 151).
- Load objects, such as transactions and synchronization points used in WebLOAD tests. Load toolbox tools are described in The WebLOAD IDE Load Toolbox (on page 157).
- Database actions, such as opening and getting data from a database for a WebLOAD IDE test. Database Utility building blocks are described in The WebLOAD IDE Database Toolbox (on page 166).
- IPP functionality, such as downloading data from an FTP site for a WebLOAD IDE test. IPP building blocks are described in The WebLOAD IDE IPP Toolbox (on page 188).
- DDoS toolbox set. This is an optional add-on tool, used to simulate DDoS attacks during test sessions, described in DDoS LOAD Testing (on page 243).

Adding Agenda Items from a WebLOAD IDE Toolbox

To drag and drop a WebLOAD IDE toolbox item into your Agenda:

1. Place the mouse over the item in the WebLOAD IDE toolbox that you want to add.
2. Press and hold the mouse button (just "clicking" has no effect).
3. Drag the item into the Agenda tree, highlighting the item after which you want to add the new item.
4. Release the Agenda 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 Agenda Tree at the point where you placed the item.
6. For JavaScript Objects, add JavaScript code to the Agenda (see Using the JavaScript Editor (on page 63)).

Working with JavaScript Files

WebLOAD IDE enables you to open a JavaScript file and convert it to a WebLOAD IDE 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 agenda) and not the main Agenda.

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

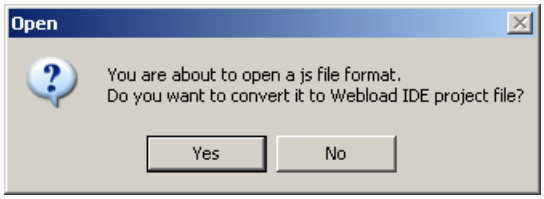
- The project file is better suited to the WebLOAD IDE visual environment.
- Allows you to save additional information to the script, such as the Current Project options.

Note: When you convert a JavaScript file to a WebLOAD IDE 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, select File | Open.
2. Select a JavaScript file.

   The Open message appears.

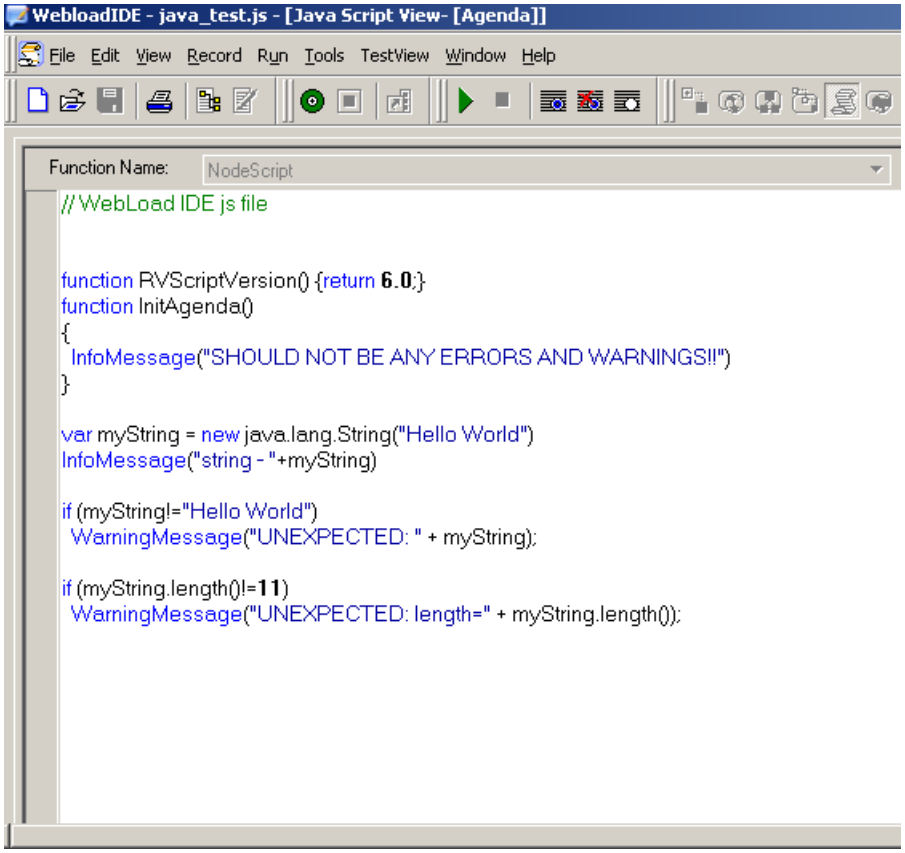


3. Click Yes to convert the JavaScript file to a WebLOAD IDE project file

   -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 IDE block shows that it is a JavaScript file.



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

C H A P T E R 7

Running and Debugging Agendas

This section provides instructions for editing Agenda with WebLOAD IDE.

In This Chapter

About Running and Debugging Agendas with WebLOAD IDE

When you run your Agenda, WebLOAD IDE interacts with your Web application just as a real user would. WebLOAD IDE runs your Agenda line by line. As your Agenda executes, execution arrows are displayed in the left margin of the Agenda 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 IDE to stop the entire test.

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

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

Running an Agenda

This section provides instructions for running an Agenda.

Before running an Agenda, you can do the following:

- Set the number of iterations to run, see Setting Playback Iteration (on page 140).
- Set the file locations for a test session, see Setting File Locations (on page 141).
- Set WebLOAD IDE to ignore the recorded sleep time, see Configuring Sleep Time Control Options (on page 124).

Starting the Execution of an Agenda

To execute the Agenda:

1. In the main window, select File | Open and open the Agenda you want to edit.
2. Select Run | Run

   -Or-

   Click the Run Test toolbar button

   -Or-

   Click the Step Into toolbar button to run the Agenda step-by-step.

   The Agenda runs and displays the following:

   - A sequence of the events generated by the Agenda in the Execution Tree.
   - The execution sequence in the JavaScript View pane and the Agenda Tree.
   - If the Browser View tab is open, the pages that return 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 Iteration (on page 140)).

Viewing the Execution Sequence in the Agenda Tree

When you run your Agenda, WebLOAD IDE interacts with your Web application just as a real user would. WebLOAD IDE runs your Agenda line by line. Execution arrows are displayed in the left margin of the Agenda Tree.

WebLOAD IDE enables you to do the following:

- Run through the entire Agenda, run it line by line, and add breakpoints (see Debugging an Agenda (on page 81)).
- Display the Current Project Options by right-clicking the Agenda root node, and clicking Current Project Options from the pop-up menu (see Configuring the Default and Current Project Options (on page 120)).

  All of the fields are read only

To view the Agenda Tree:

- In the main window, click View | Agenda Tree

  -Or-

  Click the Agenda Tree View  toolbar button.

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

Viewing the Execution Sequence in the JavaScript View Pane

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

WebLOAD IDE enables you to do the following:

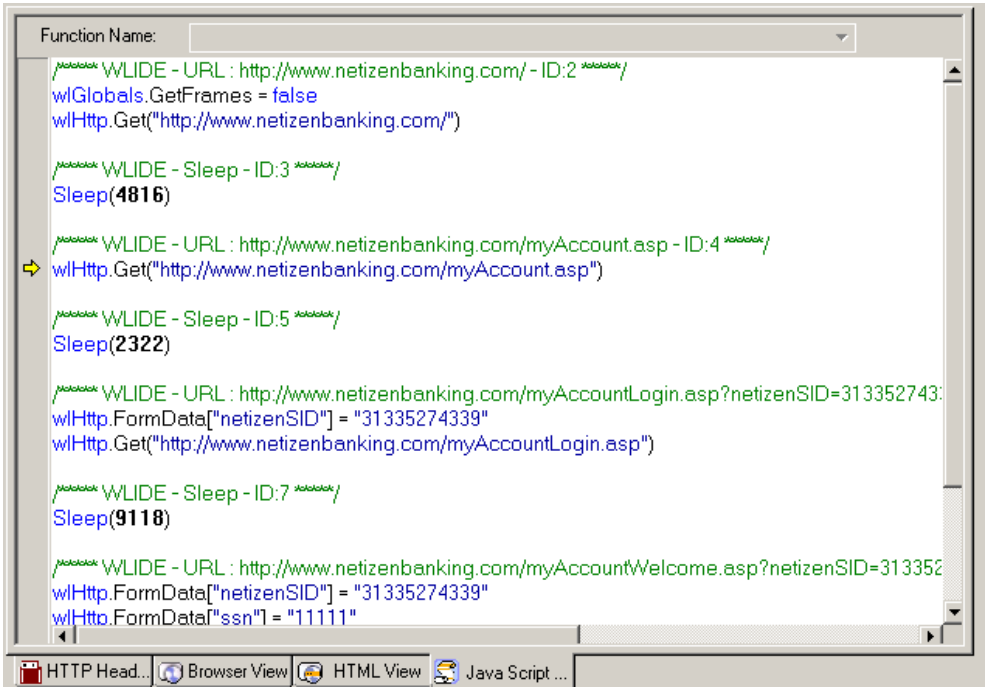
- Run through the entire Agenda, run it line by line, add breakpoints, and add Watch variables (see Debugging an Agenda (on page 81)).
- Check the syntax by right-clicking in the Agenda and clicking Check Syntax from the pop-up menu.

To view the JavaScript View:

- In the main window, click View | JavaScript View

  -Or-

  Click the JavaScript View  toolbar button.



Viewing the Response Data in the Execution Tree

As you execute an Agenda, WebLOAD IDE 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.

Stopping the Execution of an Agenda

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

To stop the execution of an Agenda:

- Select Run | Stop

  -Or-

  Click the Stop Execution  toolbar button.

  -Or

  Use the hotkeys Shift + F5.

  The playback session is stopped.

Debugging Agendas

WebLOAD IDE provides an integrated debugger with a variety of tools to help locate bugs in your Agenda. The debugger provides special menus, windows, dialog boxes, and grids of fields for debugging. You can pause the debugger and trigger WebLOAD IDE to wait for user input

before proceeding with running the Agenda. In the Agenda, you can set breakpoints and step into / over / out.

Run Menu Items

Commands for debugging can be found on the Run menu, and the View | Debug Windows menu.

The Run menu contains commands to start the debugging process.

The following options are available through the Run menu.

Menu Item		Description
Run		Starts playback of the Agenda script from the current statement until a breakpoint or the end of the Agenda is reached.
Stop		Stops the playback of the Agenda script.
Step Into		Starts the play back of the Agenda script, a step at a time, entering each function encountered.
Step Over		Starts the playback of the Agenda, on 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 Agenda 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		Stops the playback of the Agenda at that point.
Toggle Breakpoint		Defines a line in the Agenda where WebLOAD IDE suspends execution.
Remove all Breakpoints		Eliminates all breakpoints.
Disable/Enable all Breakpoints		Disable or enable all breakpoints.
Add Watch...		Adds a selected expression to the Watch window.

Debug Windows

WebLOAD IDE provides several specialized windows to display debugging information for your Agenda. While debugging your Agenda, you can access these windows using through the View | Debug Windows menu. By default, the Watch, Variables, and Call Stack Windows appear at the bottom of the main window.

The View | Debug Windows menu contains commands that display the various debugger windows.

Menu Item		Description
Watch		Opens the Watch window (available only during runtime in debug mode), and displays the names and values of variables and expressions.
Variables		Opens the Variables window (available only during runtime in debug mode), and displays information about variables used in the current and previous statements and functions, (in the Auto tab) and the object pointed to by this (in the This tab).
Call Stack		The Call Stack window lists the function calls that led to the current statement, with the current function on the top of the stack.

Debugging an Agenda

When debugging an Agenda, you can set the Agenda to run in the following ways:

- Step-by-step

  The execution starts at the first line of the Agenda and stops at each subsequent line.

- Breakpoints

  The execution starts at the first line of the Agenda and stops when it reaches a breakpoint.

- A combination of step-by-step and breakpoints

To debug an Agenda:

1. Select Run | Run

   -Or-

Click the Run on toolbar button

-Or-

Select Run | Step Into.

2. When you reach the end of the script you can:

   a. Click Step Into to return to the beginning of the script.

   b. View results (see Viewing and Analyzing the Test Results (on page 92)).

   c. Add breakpoints (see Setting Breakpoints (on page 82)).

3. Return to Edit mode and revise your Agenda.

Starting the Debugger

To start debugging:

- Click the Run toolbar button to run the Agenda continuously

  -Or-

  Use the Step Into toolbar button to run the Agenda step-by-step.

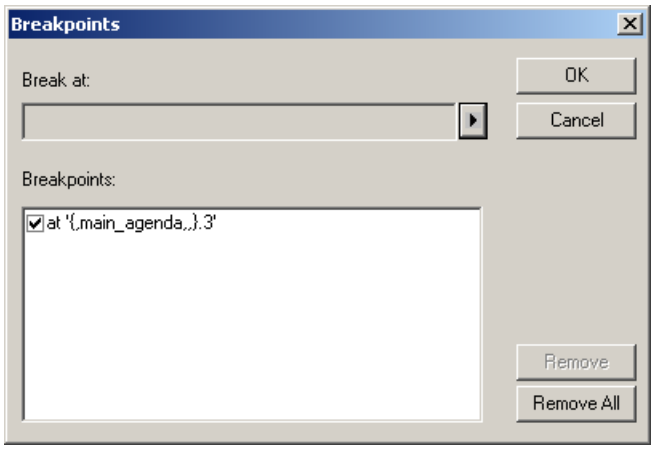
Setting Breakpoints

Use Breakpoints to define places in the Agenda 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 IDE project.

To set multiple breakpoints to an Agenda:

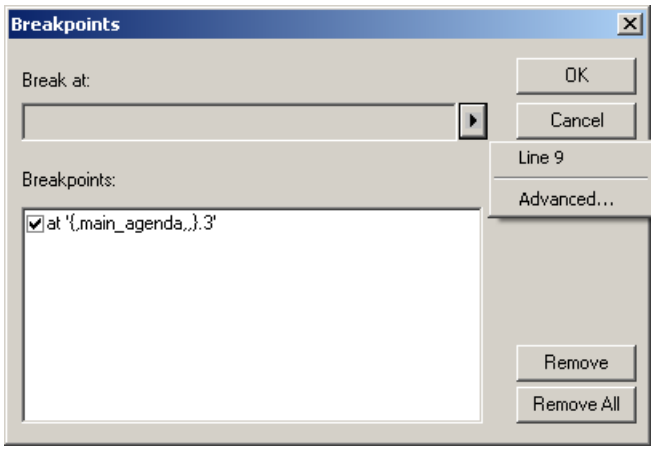
1. Display the entire Agenda.
2. Select the line of code.
3. Click Edit | Breakpoints.

   The Breakpoints dialog box opens.



4. Click the arrow.

   The options appear.

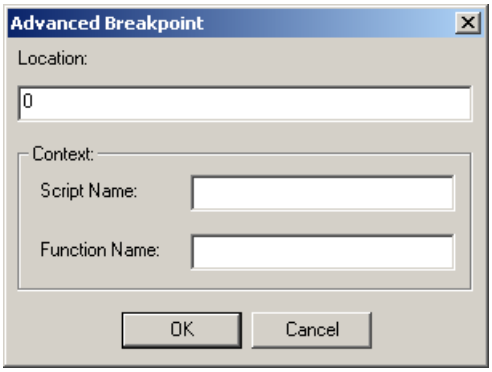


5. Click the Line number.

   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.



7. Fill in the fields, and click OK.

To set a breakpoint in the Agenda Tree:

1. Right-click an item in the Agenda 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 Agenda Tree adjacent to the visual Agenda element for which the breakpoint is defined, indicating that the breakpoint is set.

To set a breakpoint in the JavaScript View pane:

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

   -Or-

   Click the Toggle Breakpoint toolbar button.

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

To set a breakpoint in Debug mode:

1. Run the Agenda by clicking the Step Into  toolbar button.
2. Continue stepping through the Agenda 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 Run | Toggle Breakpoint

   -Or-

   Click the Toggle Breakpoint  toolbar button.

   While in debug mode a red dot appears in the left margin of your Agenda 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 82)).
2. Click Run | Run

   -Or-

   Click the Run  toolbar button

   -Or-

   Click the Step Into  toolbar button to run the Agenda step-by-step.

Removing Breakpoints

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

To remove a breakpoint:

1. Click the JavaScript View  toolbar button to open the JavaScript View pane.
2. In the Agenda Tree, click the Agenda root node to display the entire Agenda in the JavaScript View pane.
3. In the JavaScript View pane, select the line containing the breakpoint you want to remove.
4. Click Run | Toggle Breakpoint

   -Or-

Click the Toggle Breakpoint  toolbar button.

The red dot in the left margin disappears.

To remove all breakpoints:

1. Click the JavaScript View  toolbar button to open the JavaScript View pane.
2. In the Agenda Tree, click the Agenda root node to display the entire Agenda in the JavaScript View pane.
3. Click Run | Remove all Breakpoints

   -Or-

   Click the Remove Breakpoint  toolbar button.

   The red dot in the left margin disappears.

Disabling and Enabling All Breakpoints

You can disable or enable all breakpoints in the Agenda.

To disable or enable all breakpoints:

1. Click the JavaScript View  toolbar button to open the JavaScript View pane.
2. In the Agenda Tree, click the Agenda root node to display the entire Agenda in the JavaScript View pane.
3. Click Run | Disable/Enable All Breakpoints

   -Or-

   Click the Disable/Enable Breakpoints toolbar button.

   - 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 Agenda

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

1. Click Run | Step Into

   -Or-

Click the Step Into  toolbar button.

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 Agenda 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 Agenda execution to that point.

   For information on setting breakpoints see Setting Breakpoints (on page 82).

2. Click Run | Step Into

   -Or-

   Click the Step Into  toolbar button.

Stepping Out or Over a Function

To step over a function:

1. Click Run  or Step Into  .
2. Execute the Agenda to the function call.
3. Click Run | Step Over

   -Or-

   Click the Step Over  toolbar button.

   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  and execute the program to some point inside the function.
2. Click Run | Step Out

   -Or-

   Click the Step Out  toolbar button.

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

Stopping the Playback of the Agenda

You can stop the playback of the Agenda at a specific point.

To stop the playback of the Agenda:

1. Start debugging. Click Run or Step Into.
2. Click Run | Break or click the Break toolbar button.

   The Agenda stops running.

Using the Watch Window

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

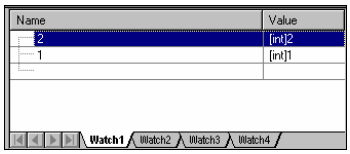
Use the Watch window to specify variables and expressions that you want to watch while debugging your program. You can also modify the value of a variable using the Watch window. To add a watch variable, see Adding a Watch Variable (on page 89).

To open the Watch window:

- In the main window, click View | Debug Windows | Watch

  -Or-

  Click the Watch Window toolbar button.



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.
2. Click View | Debug Windows | Watch, or click the Watch Window toolbar button 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

To add a Watch variable in the JavaScript View pane:

1. Start debugging. Click Run or Step Into.
2. Click the JavaScript View toolbar button to open the JavaScript View pane.
3. In the Agenda Tree, click the Agenda root node to display the entire Agenda 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. Click Run | Start Debug | Add Watch

   -Or-

Right-click the variable in the JavaScript View pane, and click Add Watch from the pop-up menu.

The Add Watch dialog box opens.

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.

To add a Watch variable in the Watch window:

1. Start debugging. Click Run or Step Into.
2. Click View | Debug Windows | Watch, or click the Watch Window toolbar button to open the Watch window.
3. In the Watch window, select a tab for the variable or expression.
4. Press F2, and type the variable name in the Name column on the tab.
5. Press ENTER (if typing).

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

   If you add 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.

Viewing the Variables Window

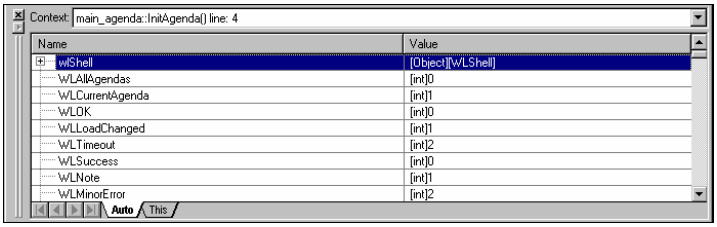
The Variables window provides quick access to variables that are important in the Agendas current context.

To open the Variables Window:

1. Start debugging. Click Run or Step Into.
2. In the main window, click View | Debug Windows | Variables

   -Or-

   Click the Variables Window toolbar button.



The window includes two tabs:

- Auto tab: 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.
- This tab: Displays the object pointed to by this.

Each tab 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 cannot add variables or expressions to the Variables window (you must use the Watch window (see Adding a Watch Variable (on page 89))), 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.

In addition to the tabs, the Variables window has a Context box 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.
2. Click View | Debug Windows | Variables, or click the Variables window toolbar button to open the Variables window.
3. Click the Auto or This tab, according to the type of variables you want to see.

Viewing the Call Stack Window

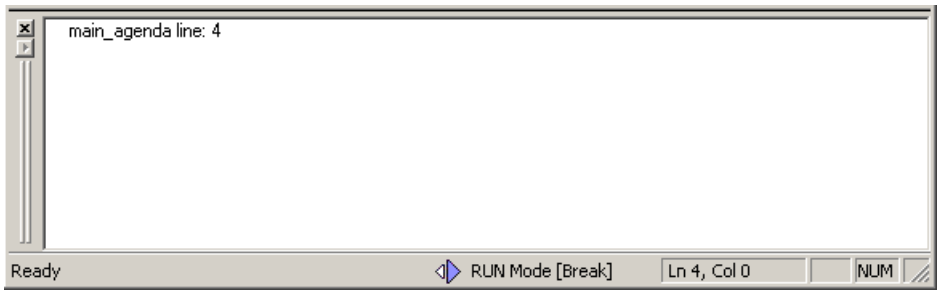
The Call Stack window lists the function calls that led to the current statement, with the current function on the top of the stack.

To open the Call Stack Window:

1. Start debugging. Click Run or Step Into.
2. In the main window, click View | Debug Windows | Call Stack

   -Or-

   Click the Call Stack toolbar button.



Viewing and Analyzing the Test Results

After running your Agenda, WebLOAD IDE 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 an Agenda, WebLOAD IDE displays the Web pages accessed in the Web application in the Execution Tree.

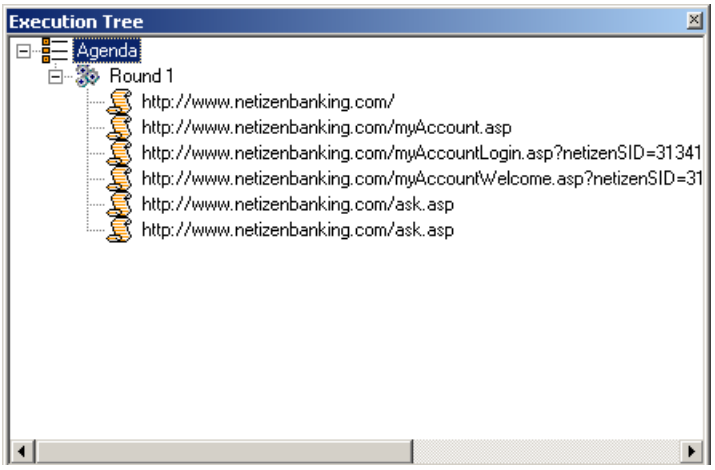
When working with a file JavaScript file that has not been converted to a WebLOAD IDE project file, WebLOAD IDE displays a playback node for each HTTP request of the JavaScript.

To open the Execution Tree:

- In the main window, click View | Execution Tree

  -Or-

Click the Execution Tree View  toolbar button.



Using the Browser View to View Results

The Browser View displays a visual representation of the baseline set of Web pages in your Agenda.

To open the Browser View:

- In the main window, click View | Browser Preview

  -Or-

Click the Browser View  toolbar button.

Using the DOM View to View Results

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

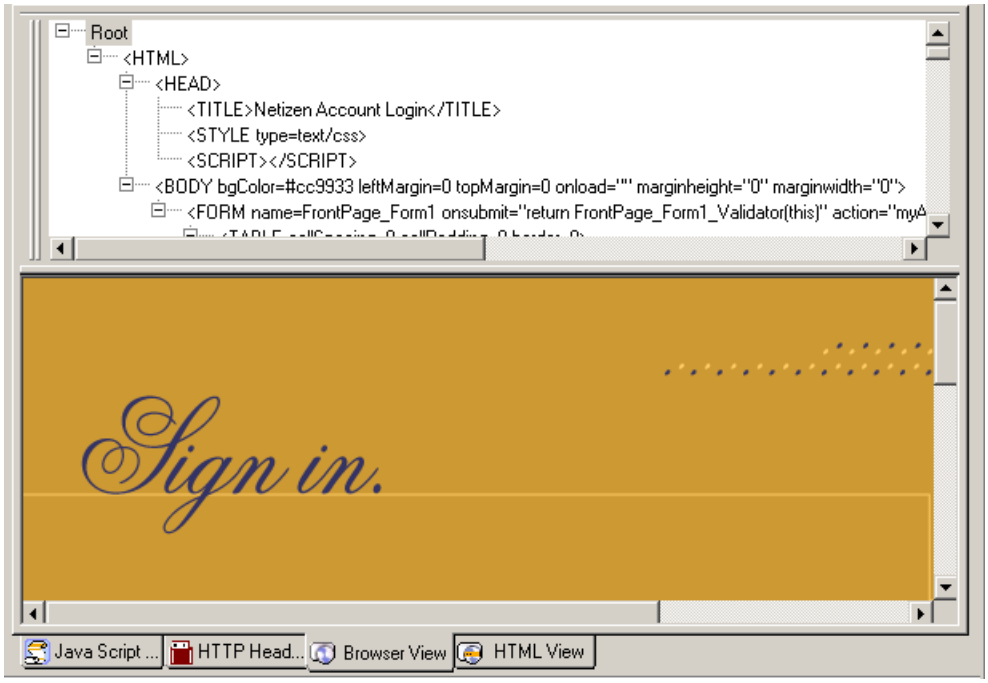
DOM View is available when Browser View is open. When an element is selected in the DOM View, the object is highlighted in the Browser View.

To open the DOM View:

- In the main window, click View | DOM View

  -Or-

Click the DOM View  toolbar button.



Using the HTML View to View Results

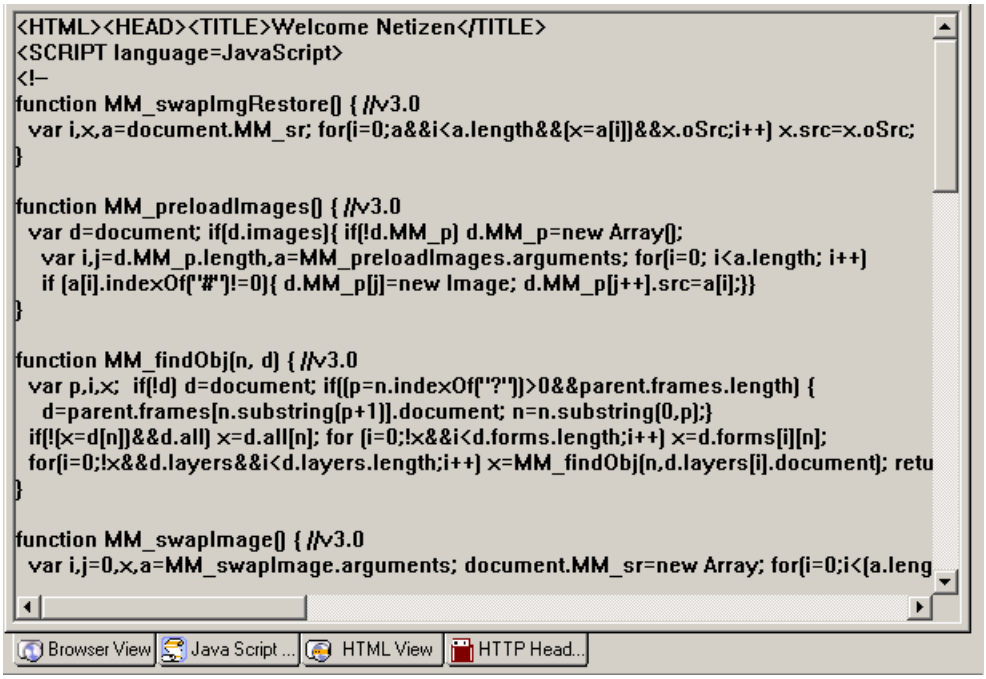
HTML view displays an HTML preview of each page and frame requested in the Agenda.

To open the HTML View:

1. In the main window, click View | HTML View

   -Or-

Click the HTML View toolbar button.



2. To search for text, do the following:

   a. Right-click and click Find... from-the pop-up menu.

   b. Type the text you want to find, and click Find Next.

3. To copy text, do the following:

   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

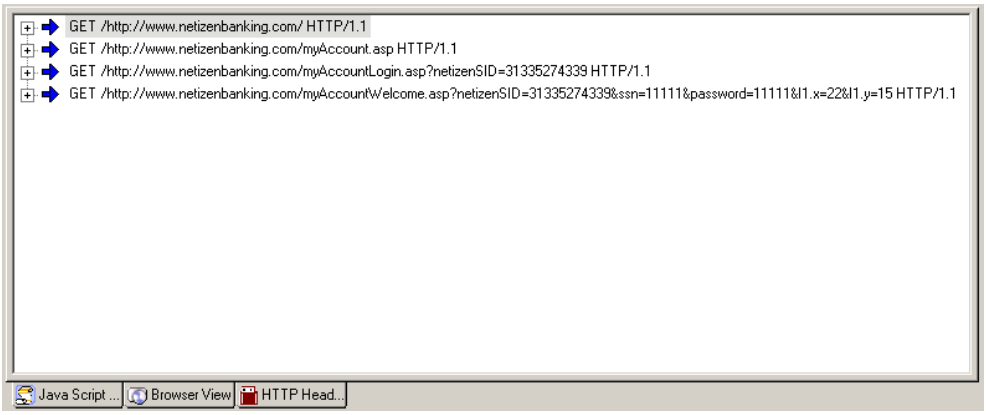
The HTTP Headers View displays the GET and POST HTTP protocol commands. Other commands can also be displayed, such as CONNECT.

To open the HTTP Headers View:

1. In the main window, click View | HTTP Headers View

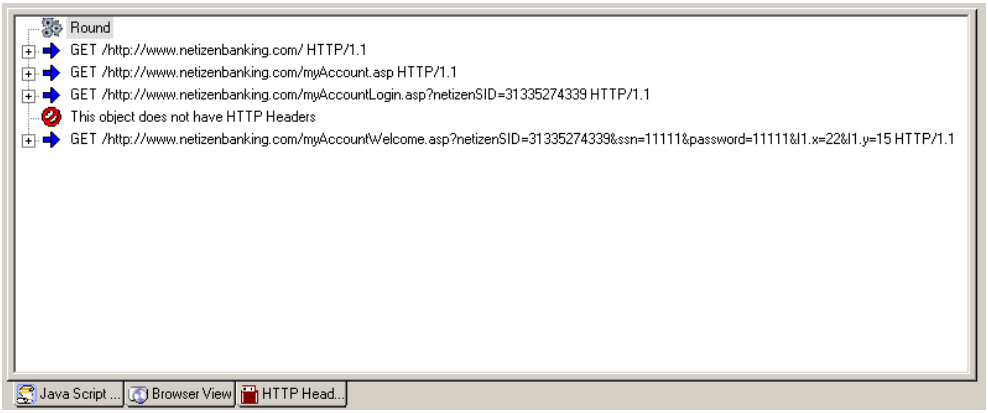
   -Or-

Click the HTTP Headers View  toolbar button.



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.



2. To view all of the headers on the Agenda, click the Agenda root node.
3. To view headers of a specific round, click the Round node in the Execution tree.
4. To search for text, do the following:
   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.
5. To copy text, do the following:
   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 Agenda 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. A higher level of severity (Error or Severe Error) ends the playback upon completion of the WebLOAD IDE protocol block.

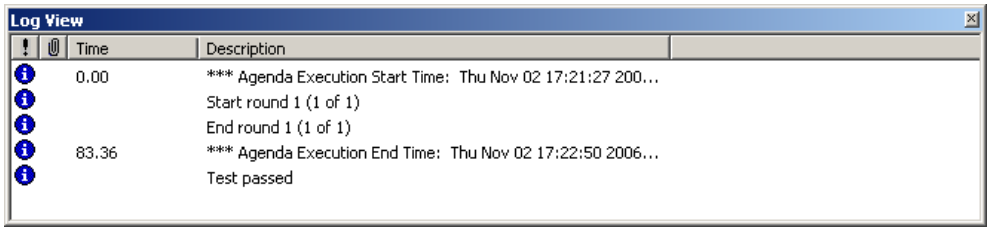
To open the Log View Window:

- In the main window, click View | Log Window

  -Or-

  Click the Log View  toolbar button.

  By default, the Log View pane appears at the bottom of the main window.



The following information is displayed:

- Message Status - 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.

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. Press Save.

   The file is saved with the extension *.log.

   You can view the saved log file with any text editor.

C H A P T E R 8

Configuring the WebLOAD IDE Options

The WebLOAD IDE configuration options are set through the Tools menu.

- Record Options: Settings that define the behavior of the WebLOAD IDE during the recording of a Web session.
- Default Project Options: Settings for WebLOAD IDE that will be in effect for each Agenda you create. These options are for the playback.
- Current Project Options: Settings that will override the Default Project Options settings.
- Settings: Settings for the WebLOAD IDE.
- Customize: Settings for the toolbar.

In This Chapter

Configuring the Record Options

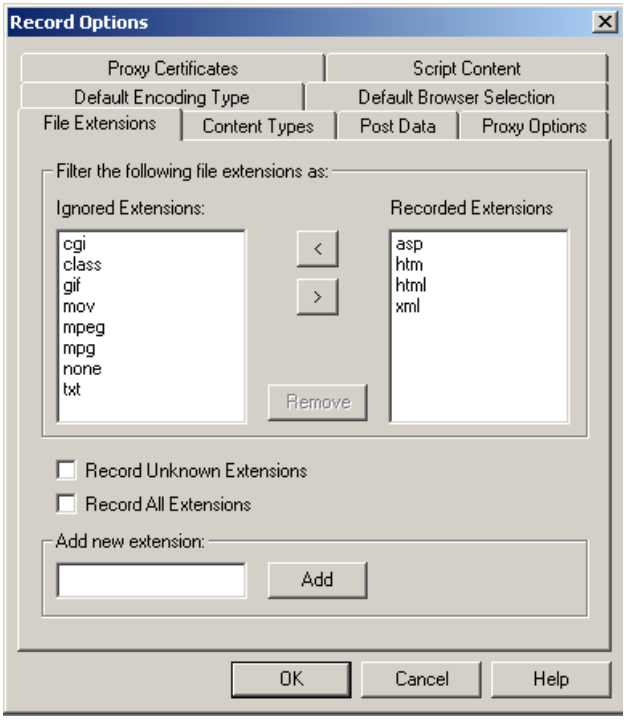
The Record Options enable you to define the behavior of the WebLOAD IDE during the recording of a Web session.

Opening the Record Options

To open the Record Options dialog box:

- Select Tools | Record Options.

  The Record Options dialog box opens with the File Extensions tab displayed.



The following table describes the tabs in the Record Options dialog box.

Tab	Description
Proxy Certificates	Configure the Server Side and Client Side certificates.
Script Content	Define how the WebLOAD IDE should handle various HTTP elements.
Post Data	Define how the WebLOAD IDE should handle Post Data.
Default Encoding Type	Select the default encoding type.

Tab	Description
Default Browser Selection	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, refer to Configuring the Proxy Value for Your Browser (on page 21).
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.
Proxy Options	Configure the proxy values for WebLOAD IDE and a secondary proxy (if necessary).

Specifying the HTTP Objects to be Recorded

Use the Script Content tab in the Record Options dialog box to specify what the WebLOAD IDE should record in your Agenda. The Script Content tab lists all the HTTP objects that can be automatically identified by the WebLOAD IDE and recorded in the Agenda so you do not have to enter them manually. For example, you can instruct WebLOAD IDE whether or not to record and display the headers.

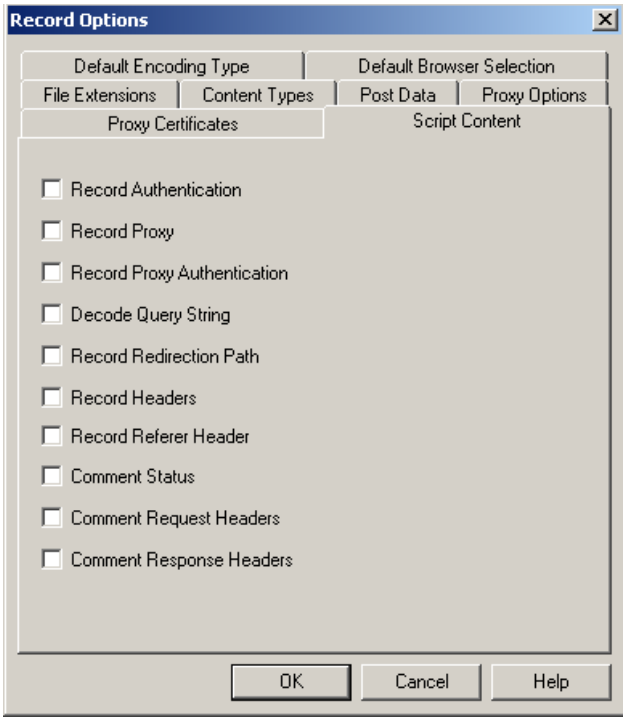
To specify the HTTP objects to be recorded:

1. Select Tools | Record Options.

   The Record Options dialog box appears.

2. Select the Script Content tab.

   The Script Content tab moves to the front of the dialog box.



3. Select or clear the options to specify what the WebLOAD IDE should record in your Agenda.
4. Click OK.

The following table describes the options in the Script Content dialog box.

HTTP Object	Description	Example
Record Authentication	Records the name and password that appear in the header of a request. This option is selected by default.	`wlHttp.UserName= "John"` `wlHttp.Password= "Blue"`
Record Proxy	Records the proxy setting. This option is selected by default.	`wlHttp.Proxy=` *<ProxyName>.<ProxyPort>*

Record Proxy Authentication	Records 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.

Record Redirection Path	Keeps a log of all the URLs visited when redirected. WebLOAD IDE records (in the Agenda) only the first GET statement; it does not record each GET statement for each redirection and it does not revisit all the URLs during playback.	`wlHttp.Get ("http://www.abcdef.com" )`
		`// Redirections:`
		`http://www.ghijkl.com, etc.`

Record Headers	Records all HTTP headers.
	The headers `If-Modified-Since` and `If-None-Matched` 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.
		`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"`

Record Referer Header	Records the referer header only. This server 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.	`wlHttp.Header["Referer"] = "http://www.easycar.com/"`
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`

Setting the WebLOAD IDE to Record Data Types

Use the Post Data tab in the Record Options dialog box to instruct the WebLOAD IDE how to treat different data types when it is recording. Data can be written in the Agenda as part of the command, as a data block, or in a data file.

For complete details on Post Data recording, see the *TestView Programmer's Guide*.

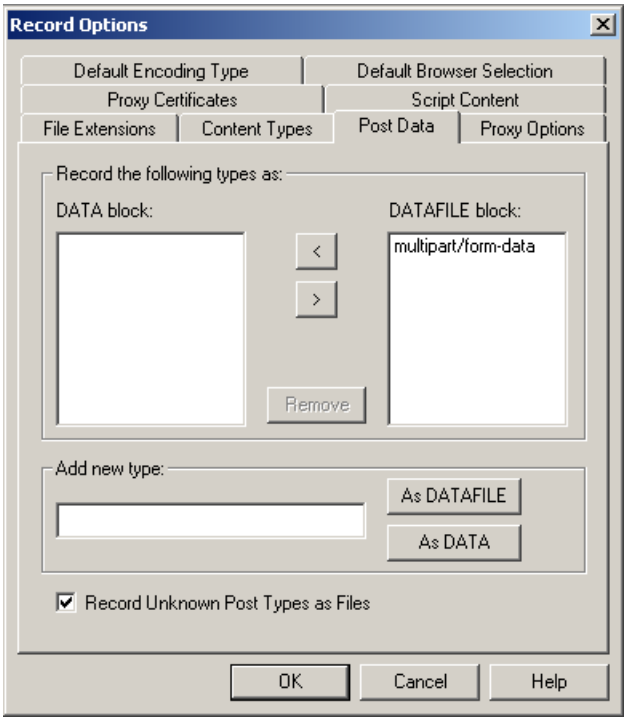
To set the WebLOAD IDE to record data types:

1. Select Tools | Record Options.

   The Record Options dialog box appears.

2. Select the Post Data tab.

   The Post Data tab moves to the front of the dialog box.



3. Fill in the fields, as described in the table below.
4. Click OK.

   The following table defines all the fields and options in the Post Data dialog box.

Field	Description
DATA block	Lists the types of data that the WebLOAD IDE records as DATA blocks (binary files).

Field	Description
DATAFILE block	Lists the types of data that the WebLOAD IDE records as DATAFILE blocks (files with a name and a path).
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 Files	Select to instruct the WebLOAD IDE to record any data type not defined on this tab as a DATAFILE.

Configuring the Default Encoding Type

Use the Default Encoding Type tab in the Record Options dialog box to set up the default encoding type.

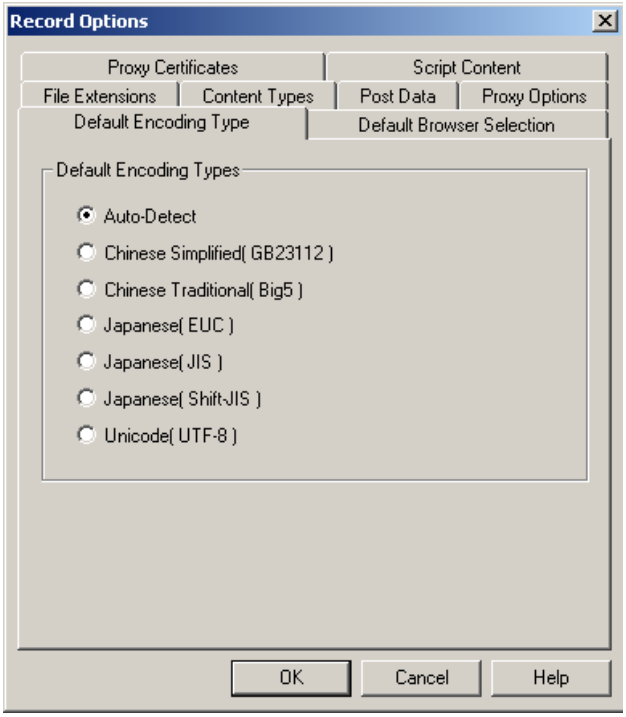
To configure the default encoding type:

1. Select Tools | Record Options.

   The Record Options dialog box appears.

2. Select the Default Encoding Type tab.

   The Default Encoding Type tab moves to the front of the dialog box.



3. Select an option as the default encoding type.
4. Click OK.

Configuring the Default Browser

Use the Default Browser Selection tab in the Record Options dialog box to set up the default browser.

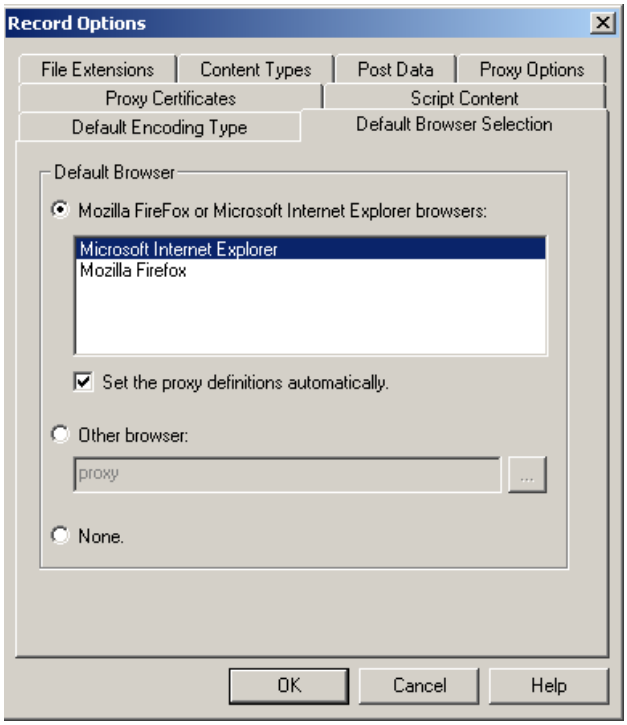
To configure the default browser:

1. Select Tools | Record Options.

   The Record Options dialog box appears.

2. Select the Default Browser Selection tab.

   The Default Browser Selection tab moves to the front of the dialog box.



3. Fill in the fields, as described in the table below.
4. Click OK.

   A message appears stating that in order for the WebLOAD IDE 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 IDE screen appears.

The following table defines all the fields and options in the Default Browser Selection dialog box.

Field	Description
Mozilla FireFox or Microsoft Internet Explorer browsers	Select this option to define Mozilla Firefox or Microsoft Internet Explorer as the browser of your choice. If you selected Mozilla Firefox as your browser, and Mozilla Firefox was installed on the machine after WebLOAD IDE 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 (refer to Configuring the Proxy Value for Your Browser (on page 21)). The next time you select check the Set the proxy definitions automatically checkbox, WebLOAD IDE will show the installation message again.
Set the proxy definitions automatically	If you selected either Mozilla Firefox or Microsoft Internet Explorer, you can also set WebLOAD IDE to configure the proxy value automatically (default). If you want to configure the proxy value manually, refer to Configuring the Proxy Value for Your Browser (on page 21). If you selected Mozilla Firefox as your browser, and Mozilla Firefox was installed on the machine after WebLOAD IDE 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 and the Set the proxy definitions automatically checkbox remains selected. If you do not accept, the Set the proxy definitions automatically checkbox is automatically cleared, and you should configure the proxy value manually (refer to Configuring the Proxy Value for Your Browser (on page 21)). The next time you select check the Set the proxy definitions automatically checkbox, WebLOAD IDE will show the installation message again.
Other Browser	Select this option and browse to define a browser other than Mozilla Firefox or Microsoft Internet Explorer as the browser of your choice.
None	Select this option to define that there is no default browser.

Configuring the File Extensions

Use the File Extension tab in the Record Options dialog box to set up which types of files the WebLOAD IDE records.

On the File Extensions tab, you specify which objects should be recorded by their file extension, such as "`.gif`", "`.wav`", or "`.txt`".

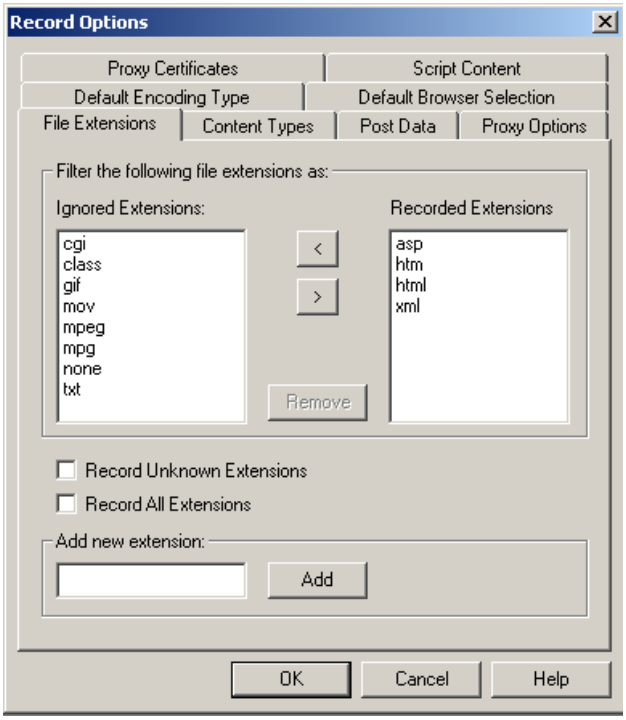
To configure the file extensions:

1. Select Tools | Record Options.

   The Record Options dialog box appears.

2. Select the File Extensions tab.

   The File Extensions tab moves to the front of the dialog box.



3. Fill in the fields, as described in the table below.
4. Click OK.

The following table describes the fields in the File Extensions dialog box.

Field	Description
Ignored Extensions	Lists the file extensions that WebLOAD IDE does not record. WebLOAD IDE 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 IDE records. WebLOAD IDE 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.
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 lists. The WebLOAD IDE 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 Record Options dialog box to set up which types of Web content the WebLOAD IDE records.

On the Content Types tab you define which objects should be recorded by type, such as "`image/gif`", "`image/jpeg`", or "`text/html`".

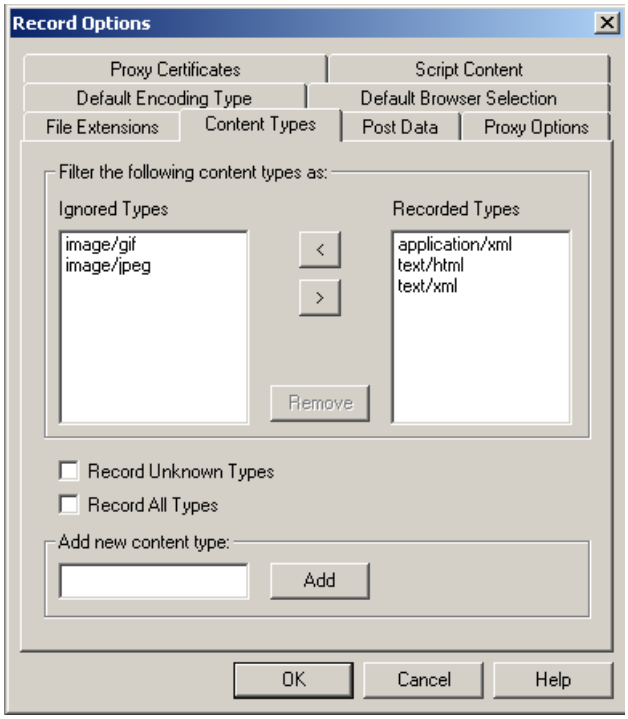
To configure the content types to record:

1. Select Tools | Record Options.

   The Record Options dialog box appears.

2. Select the Content Types tab.

   The Content Types tab moves to the front of the dialog box.



3. Fill in the fields, as described in the table below.
4. Click OK.

The following table describes the fields in the Content Types dialog box.

Field	Description
Ignored Types	Lists the content types that WebLOAD IDE does not record. WebLOAD IDE 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 IDE records. WebLOAD IDE 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.

Record All Types	Select this option to disregard the settings in the Ignored / Recorded lists. The WebLOAD IDE then records all actions involving all content types encountered during a Web session, including unknown content types.
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 window are treated as if they were included in the Recorded Types window.
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 Record Options dialog box to designate the proxy server at your organization as the secondary proxy during recording sessions or to change the proxy port number for WebLOAD IDE.

When you record Agendas with the WebLOAD IDE, your browser must be configured to use proxy port 8080 (which is the default proxy port). In other words, you must record Agendas through proxy port 8080.

If your company has a proxy server and its port number is 8080, you do not need to make any changes on the Proxy Options tab since your system is already set to record through port 8080. If your company has a proxy server and its port number is not 8080, you use the Proxy Options tab to define the proxy and to set it as the secondary proxy. Then, when you record Agendas, you record through port number 8080 and you access the Internet through the secondary proxy port.

WebLOAD IDE enables you to configure a double proxy configuration, which instructs the recorder to use two secondary proxies. To configure the double proxy, see Configuring the Double Proxy (on page 118).

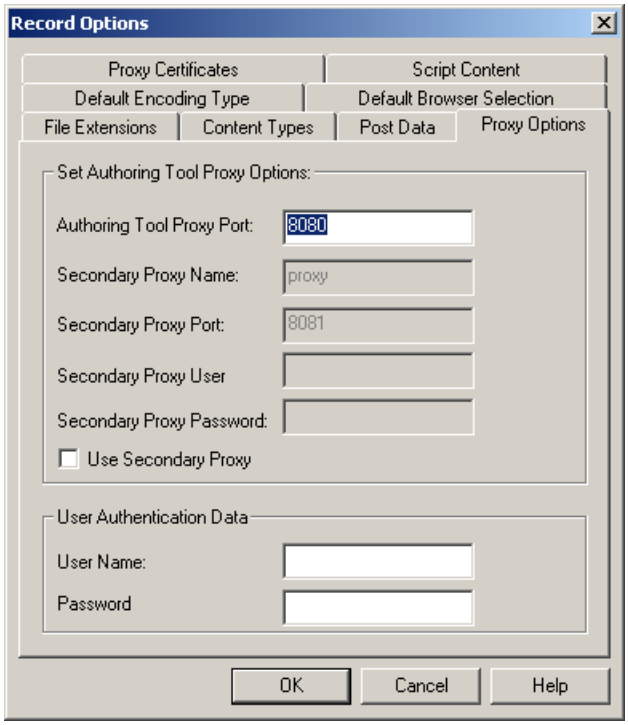
To set the proxy options:

1. Select Tools | Record Options.

   The Record Options dialog box appears.

2. Select the Proxy Options tab.

   The Proxy Options tab moves to the front of the dialog box.



3. Fill in the fields, as described in the table below.
4. Click OK.

The following table describes the fields and options on the Proxy Options dialog box.

HTTP Object	Description
Authoring Tool Proxy Port	The default port number for the WebLOAD IDE, 8080. When you are recording Agendas, your browser must use port number 8080.
Secondary Proxy Name	The name of your organization's proxy if you have one (perhaps to access the Internet beyond a company firewall).
Secondary Proxy Port	The port number of your organization's proxy if you have one.
Secondary Proxy User Name	The user who can access your organization's proxy if you have one.
Secondary Proxy Password	The password to access your organization's proxy if you have one.

Use Secondary Proxy	Select to enable the Secondary Proxy fields and to instruct WebLOAD IDE to account for a second proxy when recording an Agenda.

Configuring the Double Proxy

The double proxy configuration is a way to instruct the recorder to use two secondary proxies: one for non-secure HTTP traffic and one for SSL traffic. When you define a secondary proxy in the Proxy Options tab in the Record Options dialog box, the recorder will use the same definition for both traffic.

In order to instruct the recorder to use the double proxy definition, edit the file `wlproxyinclude.js` (which can be found in the TestView include directory) and add the following lines as in the example below:

```
ProxyObject.RUseSameSecondaryProxyForSSL = true;
ProxyObject.RSecondarySSLProxyName = "10.0.1.61"
ProxyObject.RSecondarySSLProxyPort = 3128
```

In this case, the recorder will use the definitions set in the Proxy Options tab for the proxy for non-secure traffic.

WebLOAD IDE also allows you to set authentication information for accessing the proxy. In the double proxy configuration, in order to set the authentication information for the SSL proxy, add the following lines to `wlproxyinclude.js`:

```
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 Record Options dialog box to configure the Server Side and Client Side certificates.

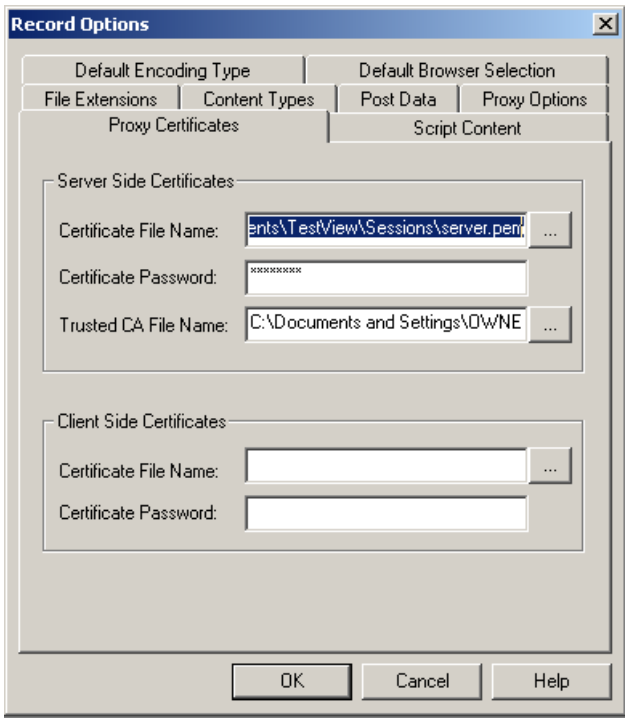
To set the proxy certificates:

1. Select Tools | Record Options.

   The Record Options dialog box appears.

2. Select the Proxy Certificates tab.

   The Proxy Certificates tab moves to the front of the dialog box.



3. Fill in the fields, as described in the table below.

4. Click OK.

The following table describes the fields and options on the Proxy Certificates dialog box.

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 TestView installation.
Certificate Password	Type the password for the supplied certificate file. Default: password of the supplied by RadView certificate.

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 TestView 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.

Configuring the Default and Current Project Options

The Project Options are settings for WebLOAD IDE that will be in effect for each Agenda you create.

- Default Project Options are settings that will be in effect for each Agenda you create. Each Agenda created or edited in WebLOAD IDE is automatically assigned these defaults. You can modify these settings to your specifications.
- Current Project Options are settings that to will override the Default Project Options settings and affect only the currently open Agenda. 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.

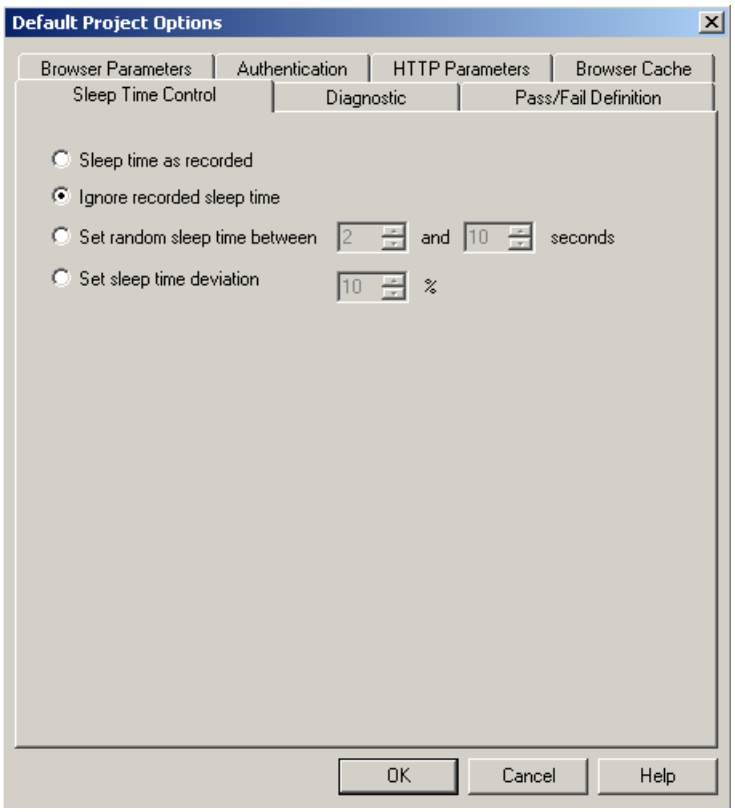
You must be in Edit mode to modify the options.

Opening the Default and Current Project Options

To open the Default Project Options dialog box:

- Select Tools | Default Project Options....

  The Default Project Options dialog box opens with the Sleep Time Control tab displayed.



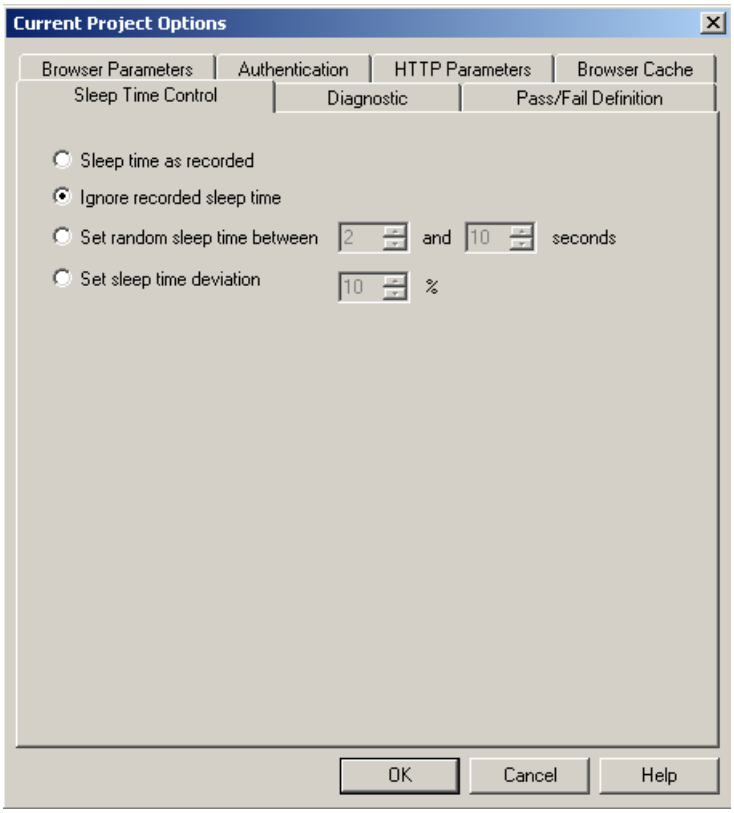
To open the Current Project Options dialog box:

- Select Tools | Current Project Options....

  -Or-

  Right-click the Agenda root node in the Agenda Tree and select Current Project Options.

The Current Project Options dialog box opens with the Playback Sleep Time tab displayed.



The following table describes the tabs in the **Default and Current Project Options** dialog box.

Tab	Description
Sleep Time Control (default)	Define the behavior of Sleep time during Agenda playback and debug. Sleep time is a pause to simulate the intermittent activity of real users.
Pass/Fail Definition	Define the test failure criteria in WebLOAD IDE.
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 an Agenda or for tracking problems in existing Agendas.

Setting Pass/Fail Definitions

Use the Pass/Fail Definition options to define test failure criteria in WebLOAD IDE.

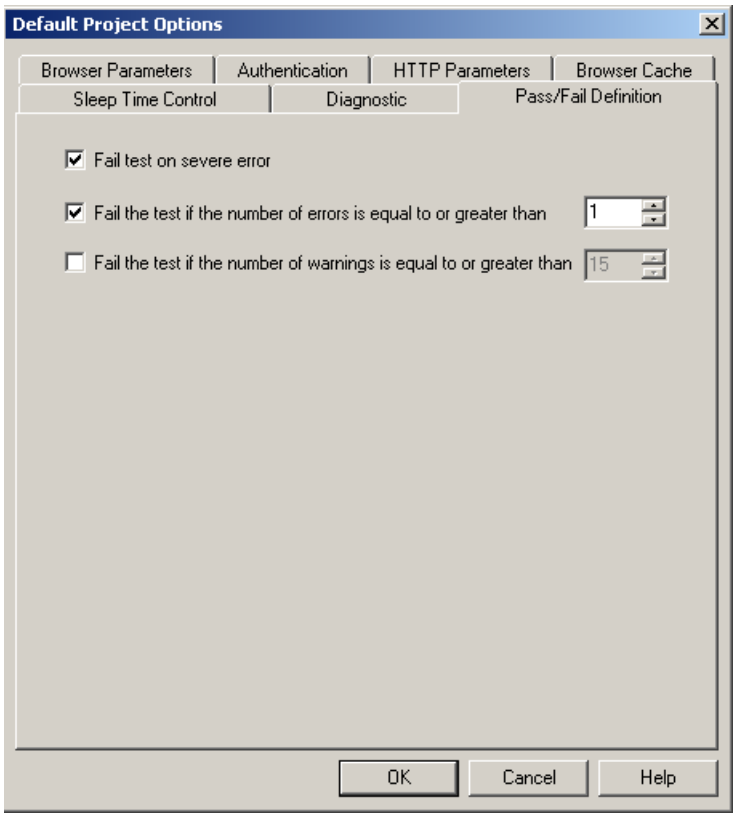
To set the Pass/Fail Definition options:

1. Select Tools | Default/Current Project Options.

   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.



3. Set test failure criteria. By default, WebLOAD IDE will fail a test if a severe error occurs during the test run. You can also set WebLOAD IDE 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 IDE enables you to control the sleep time during run-time and set an Agenda to execute with the sleep times recorded in the Agenda, random sleep times, or no sleep times.

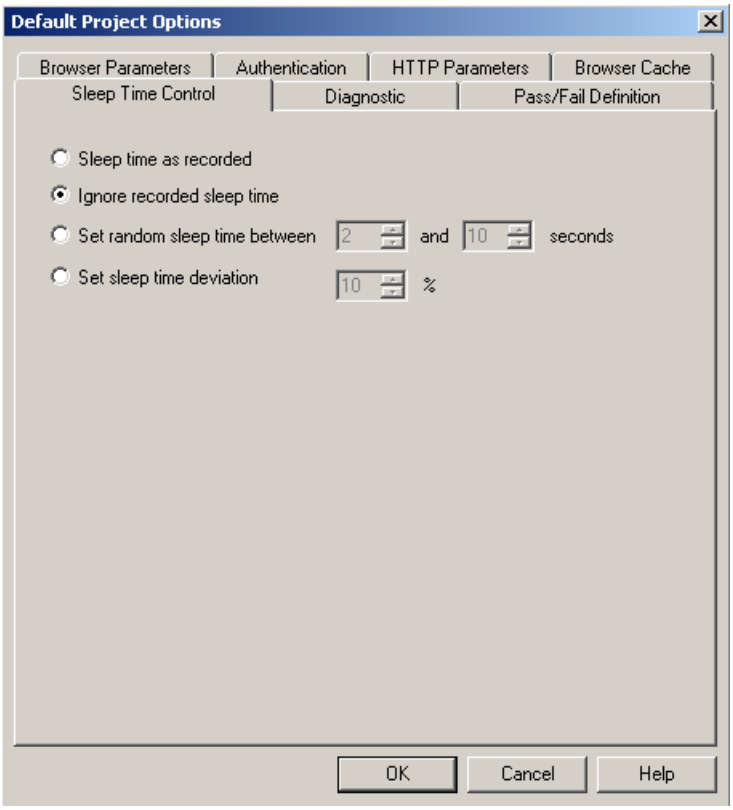
To configure Sleep Time Control options:

1. Select Tools | Default/Current Project Options.

   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.



3. Specify the type of sleep to use when playing the Agenda.

   There are four options:

   - Select Sleep time as recorded to run the Agenda with the delays corresponding to the natural pauses that occurred when recording the Agenda.

- Select Ignore recorded sleep time (default) to eliminate any pauses when running the Agenda 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.

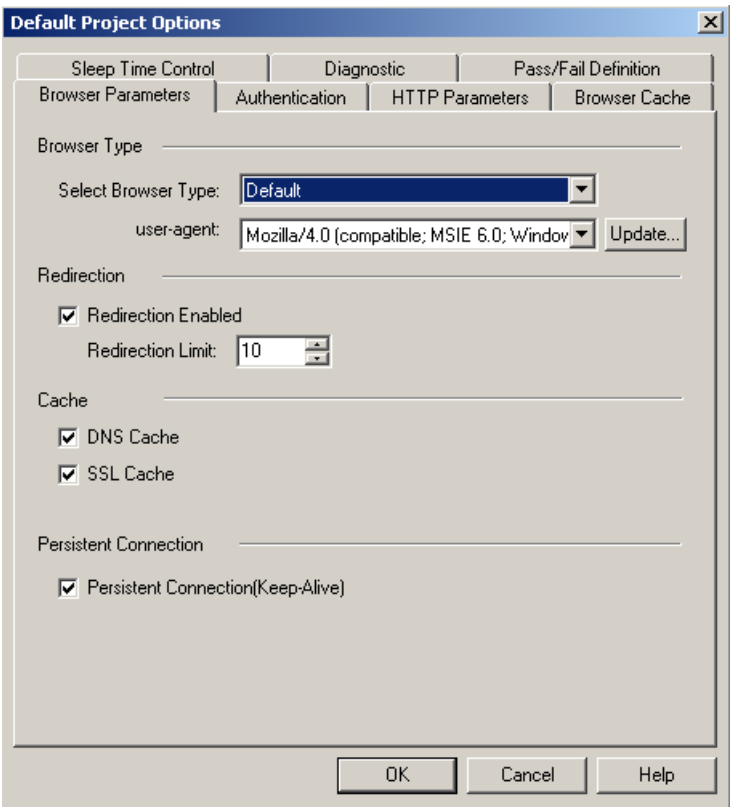
To set the Browser Parameters options:

1. Select Tools | Default/Current Project Options.

   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.



3. To set the Browser Type and User-Agent:

   a. Select a browser type from the drop-down list. An appropriate user-agent automatically appears in the user-agent field.

   b. You can select an alternative user-agent from the drop-down list, or click Update to add or delete additional user-agents (see Adding a User-Agent (on page 127) and Deleting a User-Agent (on page 128)).

      If you select WAP as the browser type, the user-agent field changes to WAP-emulation. When WAP-emulation is selected, WebLOAD IDE automatically sends all necessary headers including the screen size.

4. To set Redirection Limits:

   a. Select the Redirection Enabled checkbox.

   b. In the Redirection Limit field, type or select the desired redirection limit. The default limit is 10.

5. To simulate specific cache behaviors, select the DNS Cache checkbox and SSL Cache checkbox.

6. To enable a persistent connection to the server, select the Persistent Connection checkbox.

7. Click OK.

The following table describes the fields and buttons in the Browser Parameters tab.

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 IDE default browser agent.
Select Browser Type	You can set WebLOAD IDE to emulate any of the standard browsers.
user-agent	You can specify any specific application by supplying a custom user-agent that is included in all HTTP headers.
Redirection	
Redirection Enabled	Enables the redirection.
Redirection Limit	Sets the redirection limit.
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.

Field	Description
SSL Cache	Caches the SSL decoding keys received from an SSL server, and WebLOAD IDE only receives the key on the first SSL connection in each round. In subsequent connections, WebLOAD IDE 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.
Persistent Connection	
Persistent Connection (Keep-Alive)	When enabled, WebLOAD IDE 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.

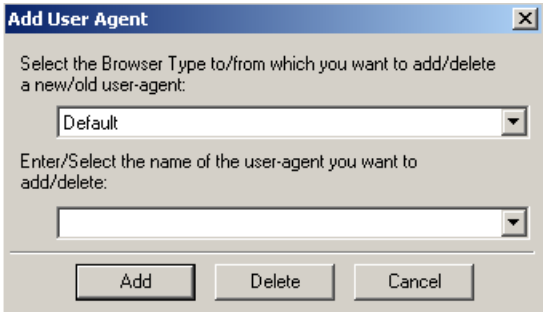
Adding a User-Agent

The available user-agent list is appropriate for the browser type you select. You can add to the user-agent list.

To add a User-Agent:

1. Click the Update button beside the user-agent field on the Browser Emulation tab.

   The Add User Agent dialog box opens.



2. Type or select the browser type to which you want to add a user-agent.
3. Type or select the new user-agent.
4. Click Add.

Deleting a User-Agent

The available user-agent list is appropriate for the browser type you select. You can delete from the user-agent list.

To delete a User-Agent:

1. Click the Update button beside the user-agent field on the Browser Emulation tab.

   The Add User Agent dialog box opens.
2. Type or select the browser type from which you want to delete a user-agent.
3. Type or select the user-agent to be deleted.
4. Click Delete.

Setting the HTTP Parameters

The HTTP Parameters option enables you to define HTTP client behavior on the HTTP protocol level.

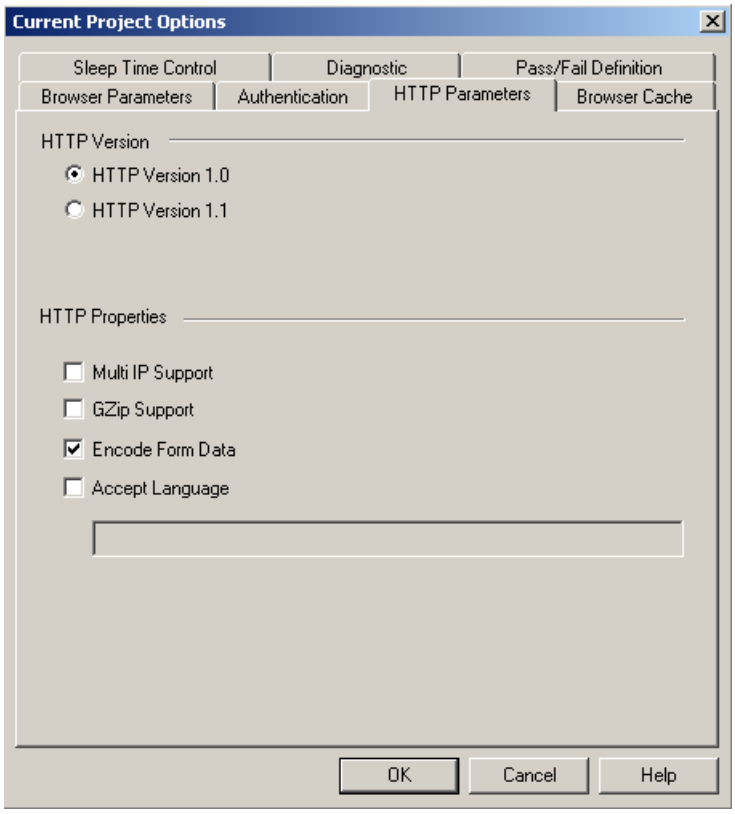
To set the HTTP Parameters options:

1. Select Tools | Default/Current Project Options.

   The Default/Current Project Options dialog box opens.

2. Select the HTTP Parameters tab.

   The HTTP Parameters tab moves to the front of the 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.

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.
HTTP Properties	
Multi IP Support	Sets the wlGlobals.MultiIPSupport flag.

Field	Description
GZip Support	Sets the wlGlobals.AcceptEncodingGzip flag. WebLOAD IDE will behave as follows: ▪ For each request, sends the header "Accept-Encoding: gzip, deflate". This tells the server that the client can accept zipped content. ▪ 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 IDE will unzip/inflate the content. Notes: ◆ 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 agenda 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 IDE will not unzip it.
Encode Form data	Sets the wlGlobals.EncodeFormdata flag. In general, when an HTTP client (Microsoft Internet Explorer/Firefox or WebLOAD IDE) 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 IDE allows 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.

Field	Description
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 IDE does not enforce any checks on the values. It is similar to the wlGlobals.UserAgent property in the sense that it is a wlGlobals/wlHttp setting that affects the value of request headers.

Setting the Browser Cache

The Browser Cache option enables you to define the type and behavior of the cache that WebLOAD IDE uses in order to simulate the behavior of a browser's cache.

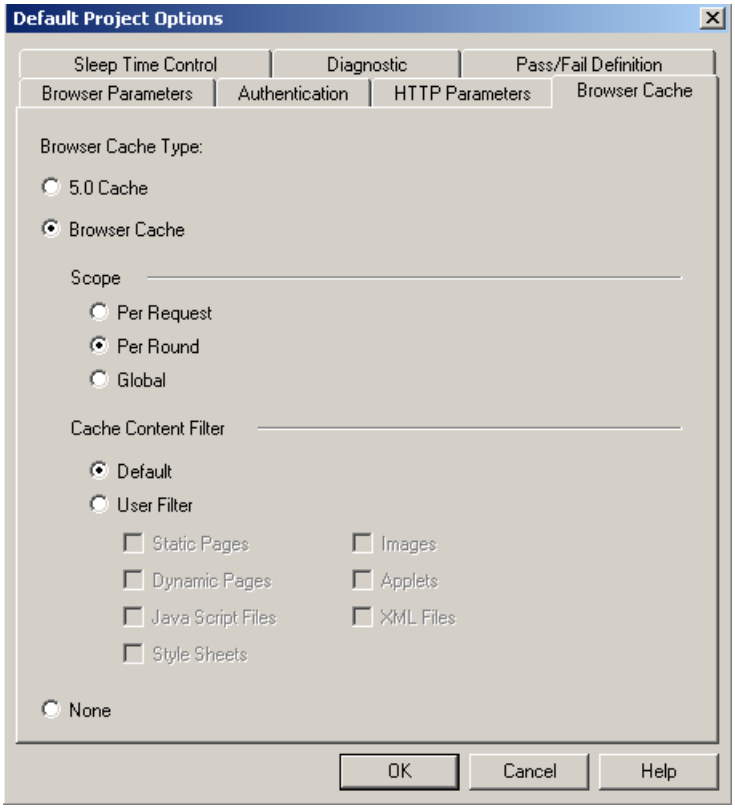
To set the Browser Cache options:

1. Select Tools | Default/Current Project Options.

   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.



3. Select the type of cache by clicking one of the following: **5.0 Cache**, **Browser Cache**, or **None**.
4. If you selected **Browser Cache**, do the following:
   a. Select a scope by clicking one of the following: **Per Request**, **Per Round**, or **Global**.
   b. Select a filter by clicking one of the following: **Default** or **User Filter**.
      If you selected **User Filter**, the filter checkboxes are enabled.
   c. If you selected **User Filter**, select one or more **User Filter** checkboxes.
5. Click **OK**.

The following table describes the fields and buttons in the **Browser Cache** dialog box.

Field	Description
5.0 Cache	All Virtual Clients simulate the browser cache the same way that WebLOAD 5.0 used to: 1. Only images were cached. 2. The content in the cache was not stored, just an indication that the image was read. 3. Whenever the engine had a request for a cached image, it simply sent nothing to the server. 4. If the same image appeared more than one time in the same page, it would always be cached only once, even if the cache is turned off. Of course, it would be cached in further pages if the cache was turned on.
Browser Cache	Stores the most recent downloaded document in a virtual browser cache. This speeds up document accesses, just as real browser cache speeds up access time. The browser cache is automatically cleared at the end of each round. WebLOAD IDE behaves as follows: 1. WebLOAD IDE can cache anything. 2. WebLOAD IDE stores actual HTML pages except for other resources, which is only an indication. 3. Whenever the engine has a request for a cached resource, it will send the request with an "if-modified-since" header. If the server responds with a 200 status, the engine will refresh the cache. 4. If the same resource appears more than one time in the same page, it will be requested with an "if-modified-since" header.
Scope	Defines when the cache will be cleared.
Per Request	Defines that cache be cleared after each request. Note: Cache will be cleared after each request on the Agenda level. It will not be cleared after internal requests.
Per Round	Defines that cache be cleared after each round.
Global	Defines that the cache will never be cleared. Each client maintains its own cache by the way and this is true for the old cache as well.
Cache Content Filter	The type of content to filter.

Field	Description
Default	Select to use the default, which is that WebLOAD IDE will cache only images and static pages.
User Filter	You can specify which content types to filter. Note: If you selected User Filter and do not select any of the content types, then WebLOAD IDE will cache all content types.
None	All Virtual Clients simulate a browser with no available cache.

Configuring Authentication Settings

The Authentication option enables you to define the Global and Proxy authentication settings.

WebLOAD IDE enables you to configure a double proxy configuration, which instructs the recorder to use two secondary proxies. To configure the double proxy, see Configuring the Double Proxy (on page 118).

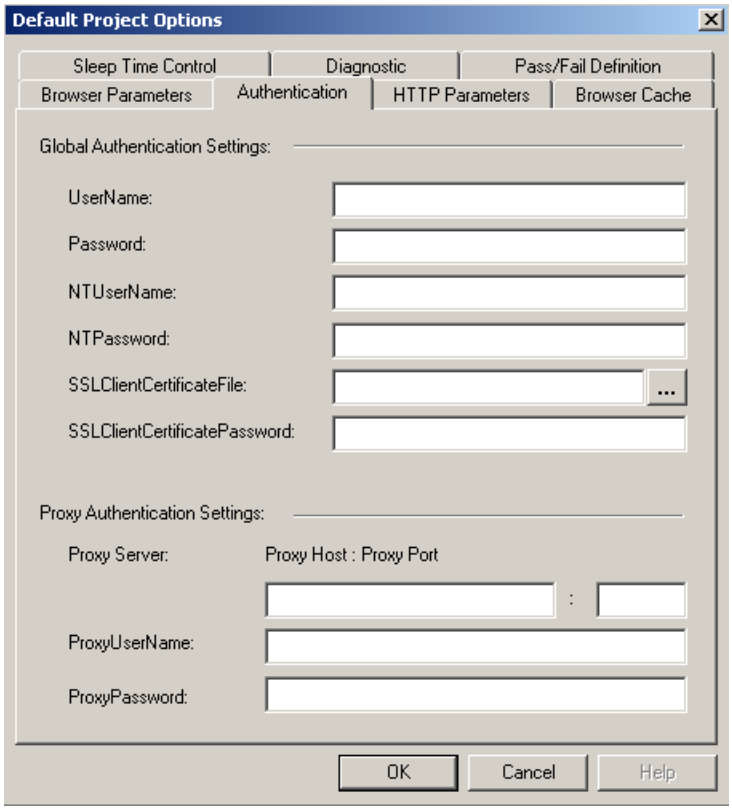
To configure Authentication settings:

1. Select Tools | Default/Current Project Options.

   The Default/Current Project Options dialog box appears.

2. Select the Authentication tab.

   The Authentication tab moves to the front of the dialog box.



3. Fill in the fields, as described in the table below.
4. Click OK.

The following table defines all the fields and options in the Authentication dialog box.

Field	Description
Global Authentication Settings	
UserName and Password	Type user name and password the Agenda should use to log onto restricted HTTP sites.
NTUserName and NTPassword	Type user name the Agenda should use for Windows NT Challenge response authentication.
SSLClientCertificateFile and SSLClientCertificatePassword	Type or browse to a file name (optionally including a directory path) of the certificate WebLOAD makes available to SSL servers and type the password.
Proxy Authentication Settings	

Field	Description
Proxy Server: Proxy Host and Proxy Port	Type host name and port number for the proxy server.
ProxyUserName and ProxyPassword	Type user name and password for the proxy server.

Setting Diagnostic Options

Diagnostic options can be enabled while developing an Agenda or for tracking problems in existing Agendas.

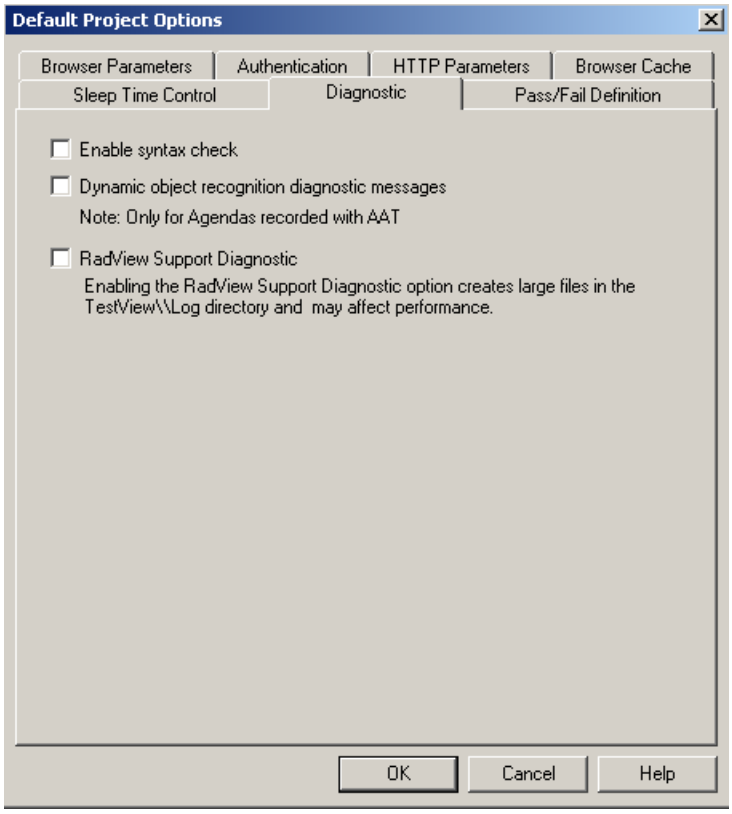
To set Diagnostic options:

1. Select Tools | Default/Current Project Options.

   The Default/Current Project Options dialog box opens.

2. Select the Diagnostic tab.

   The Diagnostic tab moves to the front of the dialog box.



3. Set the Enable syntax check option, see Enabling Syntax Checking (on page 137).
4. Set the Dynamic object recognition diagnostic messages option, see Enabling Dynamic Object Recognition Diagnostic Messages (on page 139).
5. Set the RadView Support Diagnostic option, see Enabling RadView Support Diagnostic (on page 139).
6. Click OK.

Enabling Syntax Checking

Enable syntax checking to perform the following tests on an Agenda while it is running.

Test	Description
Type Inspection	WebLOAD IDE 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".`
Value Inspections	WebLOAD IDE 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 IDE 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 IDE objects and properties are case sensitive. When syntax checking is enabled, WebLOAD IDE 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 an Agenda.

To enable syntax checking:

1. Select Tools | Default/Current Project Options.
2. Select the Diagnostic tab.
3. Select the Syntax Check checkbox.

Enabling Dynamic Object Recognition Diagnostic Messages

Enable this option to set WebLOAD IDE to send InfoMessages on the status of Dynamic Object Recognition (DOR) IdentifyObject to the log window.

Note: This option is for backward compatibility purposes only, to continue support of scripts that were developed using the AAT.

To enable Dynamic Object Recognition Diagnostic Messages:

1. Select Tools | Default/Current Project Options.
2. Select the Diagnostic tab.
3. Select the Dynamic object recognition diagnostic messages checkbox.

Enabling RadView Support Diagnostic

Enabling the RadView support diagnostic option creates large files in the WebLOAD IDE\User\Log directory that may affect Load Generator performance.

To enable RadView Support diagnostic:

1. Select Tools | Default/Current Project Options.
2. Select the Diagnostic tab.
3. Select the RadView Support Diagnostic checkbox.

Configuring the Settings

WebLOAD IDE enables you to specify settings for WebLOAD IDE.

Opening the Settings

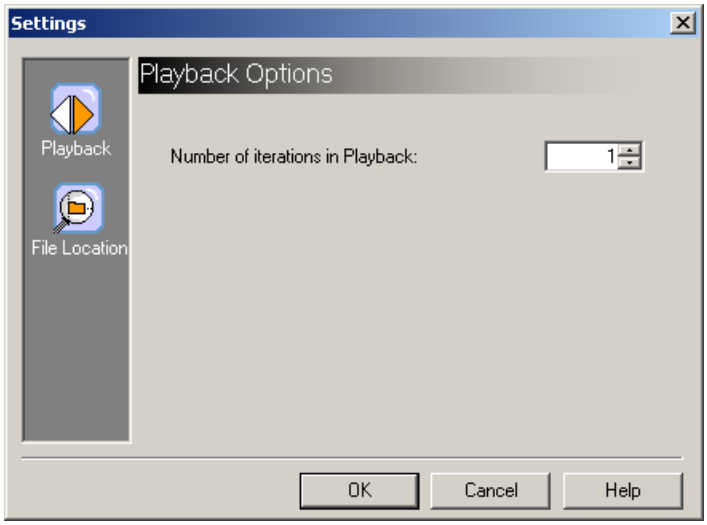
To open the Record Options dialog box:

- Select Tools | Settings

  -Or-

  Click the Settings  toolbar button.

The Settings dialog box opens.



The following table describes the options in the Settings dialog box.

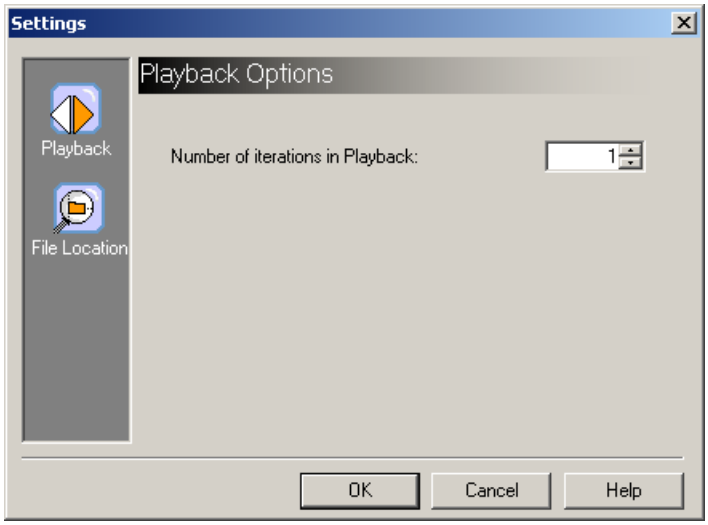
Setting	Description
Playback	Set the number of iterations to run when running Agendas with WebLOAD IDE. (Default:1)
File Location	Define the default file locations during a test session.

Setting Playback Iteration

Set the number of iterations to be run during a test session.

To set Playback iterations:

1. Select Tools | Settings and then click Playback.
2. The Settings dialog box opens with the Playback Options displayed.



3. Specify the number of iterations to run during Agenda playback. The default value is 1.
4. Click OK.

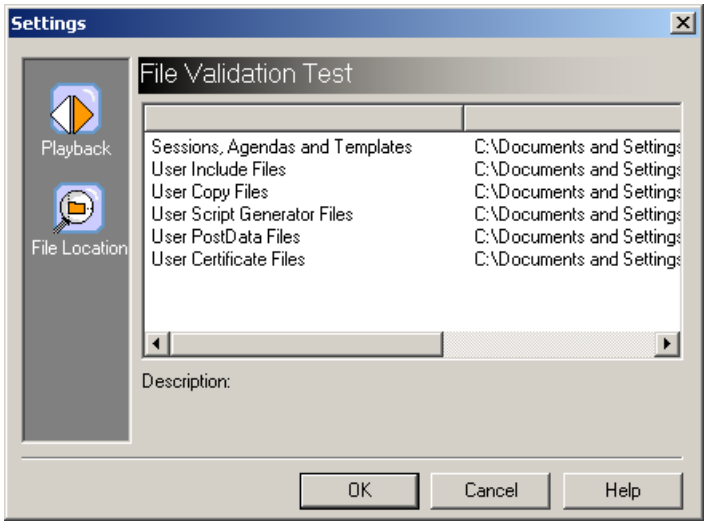
Setting File Locations

Define the default file locations during a test session.

To set the file locations:

1. Select Tools | Settings and then click File Location.

   The Settings dialog box opens with the File Locations displayed.



   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, Agendas, and Templates: Default storage location for WebLOAD IDE session, project, and Agenda files.
   - User Include Files: Default path for user `Include` files.
   - User Copy Files: Default path for user `Copy` files.
   - User Script Generator Files: Default path for user `Script Generator` 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.

Customizing the Toolbars

Use the Customize dialog box to:

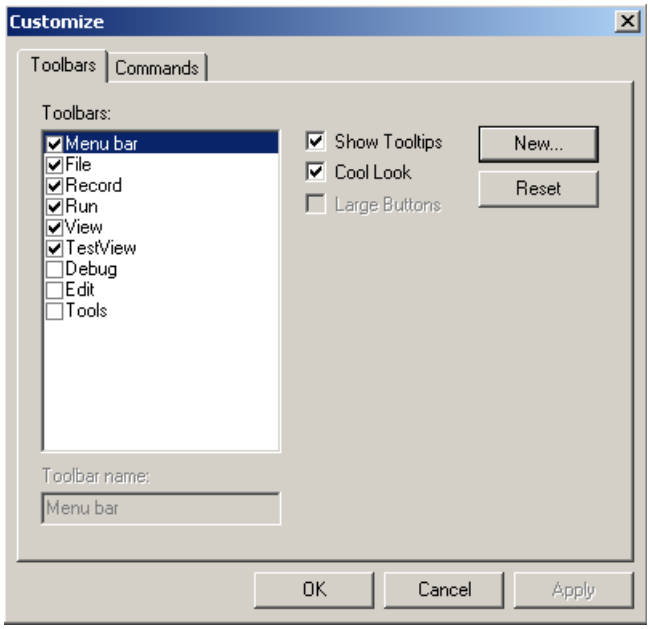
- Show/hide the various toolbars.
- Specify which buttons appear on your toolbars.
- Configure the appearance of those buttons.
- Create new toolbars.

Opening the Customize Toolbars

To open the Customize dialog box:

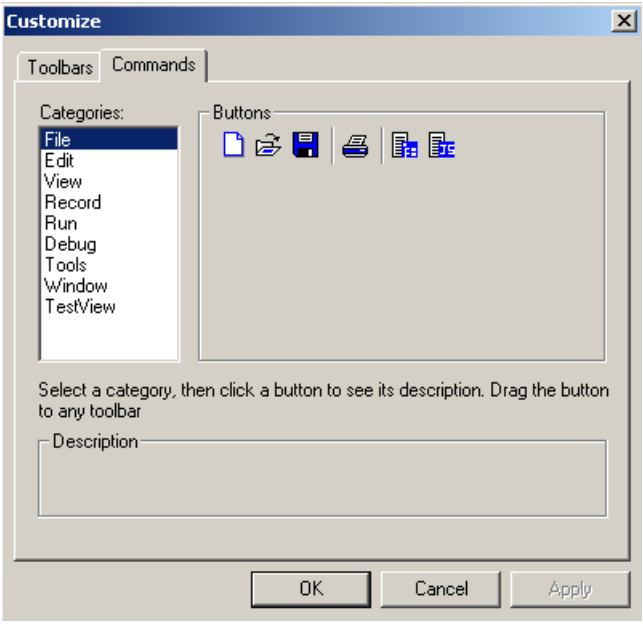
1. Select Tools | Customize.

   The Customize dialog box opens with the Toolbars tab displayed.



2. To view the buttons for each category, click the Commands tab.

   The Commands tab to move it to the front of the dialog box.



The following table describes the fields and options on the Toolbars tab in the Customize dialog box.

Field	Description
Toolbars	Lists all the toolbars in the WebLOAD IDE. The default toolbars are the Menu bar, File, Record, and Tools.
Toolbar name	Displays the name of the toolbar selected in the Toolbars list. You can edit the names of custom toolbars but not the names of standard toolbars.
Show Tooltips	Select to display a text definition of the toolbar buttons when you scroll the mouse over them.
Cool Look	Select to make the buttons appear flat. Clear to make the buttons appear to have depth.
New	Opens the New Toolbar dialog box where you can enter the name of a new toolbar.
Reset	Sets the selected toolbar back to its default configuration.

The following table describes the fields and options on the Commands tab of the Customize dialog box.

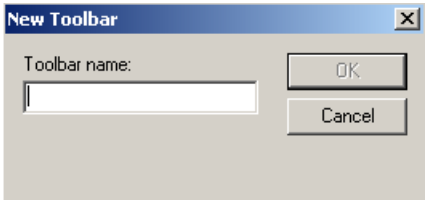
Field	Description
Categories	Lists the categories of buttons that you can work with to customize your toolbars.
Buttons	Shows the buttons in the selected category. You can select them and then drag-and-drop them onto the displayed toolbars.
Description	Shows the name or tooltip of the selected button.

Adding a New Toolbar

To add a new toolbar:

1. Select Tools | Customize and then select the Toolbars tab.
2. Click New.

   The New Toolbar dialog box appears.



3. In the Toolbar name field, type the name of the new toolbar.
4. Click OK.

   The New Toolbar dialog box is closed, the toolbar name appears in the Toolbar list, and the empty toolbar appears in the main window.
5. To add buttons to the new toolbar, see Adding a Button to a Toolbar (on page 146).
6. Click OK.

Adding a Button to a Toolbar

To add a button to a toolbar:

1. Select Tools | Customize and then select the Commands tab.
2. Select a Category.
3. Click a button from the corresponding list, and then drag it to the toolbar.

   For example:

   To add buttons from the File category to a toolbar, select File, click the Save button from the corresponding list, and drag it to the toolbar.
4. Click OK.

   The button is added to the toolbar.

Changing the Name of a Custom Toolbar

You can change the name of a custom toolbar, but you cannot change the name of a standard toolbar.

To change the name of a custom toolbar:

1. Select Tools | Customize and then select the Toolbars tab.
2. From the Toolbars list, select the toolbar whose name you want to change .

   The name of the toolbar appears in the Toolbar name field.
3. In the Toolbar name field, type a new name.
4. Click OK.

   The name of the toolbar is changed.

Deleting a Custom Toolbar

You can delete a custom toolbar, but you cannot delete a standard toolbar.

To delete the name of a custom toolbar:

1. Select Tools | Customize and then select the Toolbars tab.
2. From the Toolbars list, select the toolbar that you want to delete.
3. Click Delete.

   The toolbar is deleted.

4. Click OK.

Removing a Button from a Toolbar

To remove a button from a toolbar:

1. Select Tools | Customize and then select the Commands tab.
2. In the main window, select the button in the toolbar, and then drag it to the Commands tab in the Customize dialog box.
3. Click OK.

   The button is removed.

Restoring the Defaults for a Standard Toolbar

You can reset a standard toolbar (such as File, Record, Tools, or Edit) back to its original settings.

To restore the defaults for a standard toolbar:

1. Select Tools | Customize and then select the Toolbars tab.
2. Select the toolbar whose settings you want to restore.
3. Click Reset.

   The default toolbar buttons appear on the toolbar.
4. Click OK.

A P P E N D I X A

The WebLOAD IDE Toolbox Set

This section describes the WebLOAD IDE toolbox set.

In This Appendix

The WebLOAD IDE Toolbox Items

The following are the WebLOAD IDE Toolbox items:

General

- Sleep
- JavaScriptObject
- Global Input File
- Catch
- Message
- Comment
- Try

Load

- Begin Transaction
- End Transaction
- Set Timer
- Send Timer
- Synchronization Point
- Send Measurement
- Round Per Time Interval
- URL Screening
- Value Extraction

Database

- Open DB
- Oracle Open DB
- Execute Command
- Fetch Data
- DB GetLine
- Oracle DB GetLine
- DB Load
- Oracle DB Load

IPP

- 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-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

DDoS Load

- wlGenericAttack
- LandAttack
- Plague
- Knight

- Tfn2k
- Flitz
- Trinoo
- Stachelraht26
- Carko
- Tfn
- Juno
- Blitznet
- Stachelraht4
- Omega3

The WebLOAD IDE General Toolbox

The following table describes the purpose of each of the WebLOAD IDE General Toolbox items:

Agenda 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 Agenda. Message objects can also be used to print out the values of variables.
JavaScript Object		Lets you insert JavaScript code directly into the Agenda. You can code directly in JavaScript. To add a JavaScript Object to the Agenda, drag the JavaScript Object icon into the Agenda tree, and then open the object to insert JavaScript code.
Comment		Places comments in your Agenda. The comment will appear in the JavaScript View pane when viewing the entire Agenda.
GlobalInputFile		Lets you choose a file that its fields will be used as global variables throughout the entire script. The fields will appear in the JavaScript View pane when viewing the entire Agenda.

Agenda Item		Purpose
Try…Catch		Places Try and Catch statements in your Agenda. You can use the Try...Catch statements for structured exception handling.

Sleep

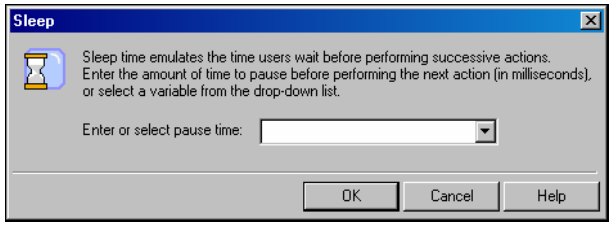
Users vary their activity when accessing a Web application, sometimes pausing between transactions and occasionally only accessing the server intermittently. The time a user waits between performing consecutive actions is known as sleep time.

When you record an Agenda, WebLOAD IDE automatically records the actual sleep time and inserts sleep icons into the Agenda. You can edit the recorded sleep times manually, add more sleep statements, and control how WebLOAD is influenced by the sleep timers in the Agenda.

To insert sleep timers:

1. Drag the Sleep  icon from the General toolbox into the Agenda Tree at the desired location.

   The Sleep dialog box opens.



2. In the Pause Time field, enter or select the duration of the sleep. The default value is 1000 milliseconds.

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

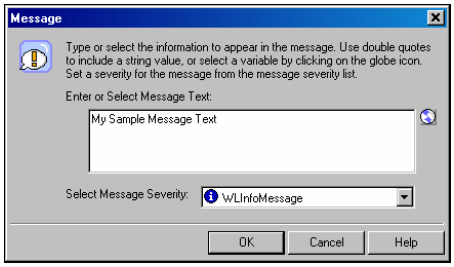
Message

While running a test session, WebLOAD IDE and WebLOAD IDE's Log windows display information about session execution. You can include Message nodes in your Agenda, 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 Agenda Tree at the desired location.

   The Message dialog box opens.



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)

5. Click OK.

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

JavaScriptObject

JavaScript Objects enable you to insert JavaScript code directly into the Agenda, giving you access to advanced functionality not available through the WebLOAD IDE 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 Agenda Tree at the desired location.

   The JavaScriptObject item appears in the Agenda Tree and the WebLOAD IDE protocol block is added to the Agenda.

2. Open the object in JavaScript Editing mode to insert JavaScript code, as described in Using the JavaScript Editor.

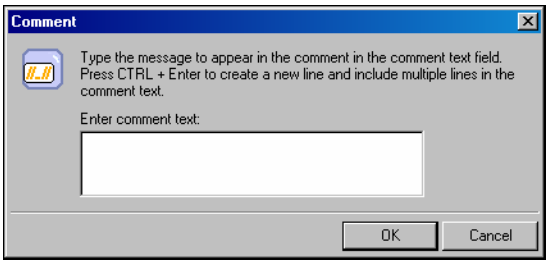
Comment

WebLOAD IDE enables you to add comments to your Agenda 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 Agenda Tree at the desired location.

   The Comment dialog box opens.



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

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

GlobalInputFile

WebLOAD IDE enables you to insert a file that its fields will be used as global variables throughout the entire script. To select the Global Input File, see Selecting a Global Input File (on page 155). To create a new Data File, see Creating a New Data File (on page 156).

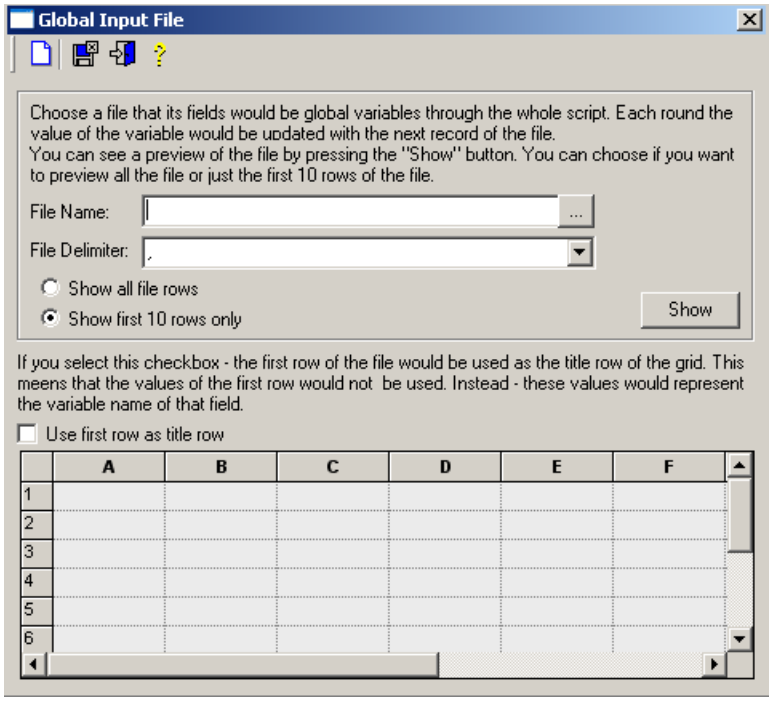
Selecting a Global Input File

You can select a Global Input File.

To select a global input file:

1. Drag the Global Input File  icon from the General toolbox into the Agenda Tree at the appropriate location.

   The Global Input File dialog box opens.



2. To create a new Data File, see Creating a New Data File (on page 156).
3.
4. a file name into the File Name field, or click the browse button to select an existing file.
5. Select a file delimiter from the drop-down list.
6. Select Show all file rows or Show first 10 rows only, and click Show.

   The file populates the grid.

7. Select the User first row as title row checkbox to set the first row of the file s the title row of the grid.

8. Click the Save and Close icon.

   The Global Input File item appears in the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode. Note that the code varies depending on the parameters selected.

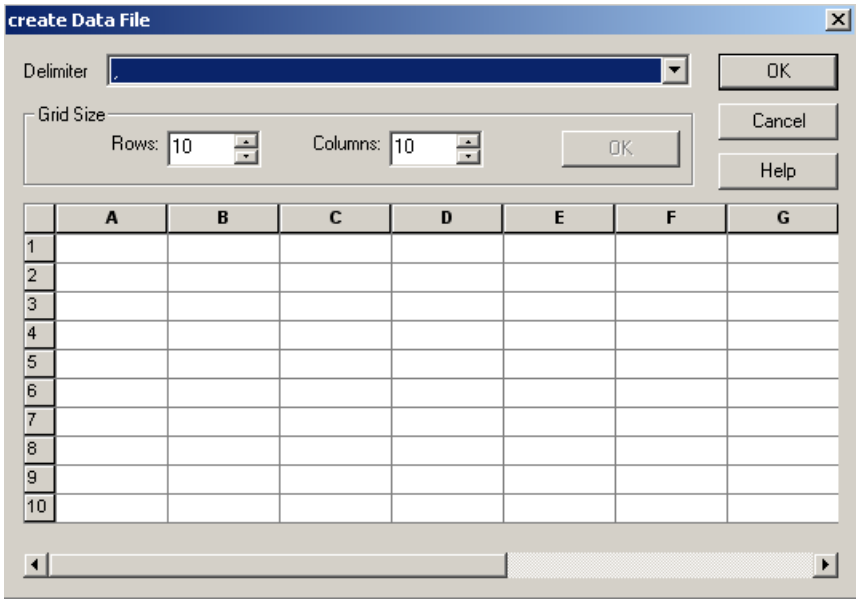
Creating a New Data File

You can create a new data file.

To create a new data file:

1. From the Global Input File dialog box (see Selecting a Global Input File (on page 155)), click the Create New Data File icon.

   The Create Data File dialog box appears.



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 each cell, type a value.
7. Click OK.

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 Agenda 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 Agenda Tree directly after the last action you want to include in the Try...Catch block.

   The Try and Catch items appear in the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The WebLOAD IDE Load Toolbox

The following table describes the purpose of each of the WebLOAD IDE Load Toolbox items:

Agenda Item		Purpose
Begin Transactions		Adds named transactions to the Agenda to measure the performance of logical actions in your Agenda, such as a Login process. By inserting named transactions into your Agenda, 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 Transactions		

Agenda Item		Purpose
Set Timer		Timers let you time any operation or group of operations in an Agenda and send the time statistics to the WebLOAD Console.
Send Time		
Synchronization Points		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.
Send Measurement		Create a new measurement name and assign a value to the measurement available in WebLOAD reports.
Round per Time Interval		Specify the maximum number of times a round may be executed per time interval.
URL Screening		Specify the URLs the emulation engine should ignore (not fetch).
Value Extraction		Extract a value from a string using a prefix and suffix.

Begin and End Transaction

In addition to the automatic transactions provided by WebLOAD, you can use the WebLOAD IDE GUI to easily add named transactions to the Agenda to measure the performance of logical actions in your Agenda, such as a Login process. By inserting named transactions into your Agenda, 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.

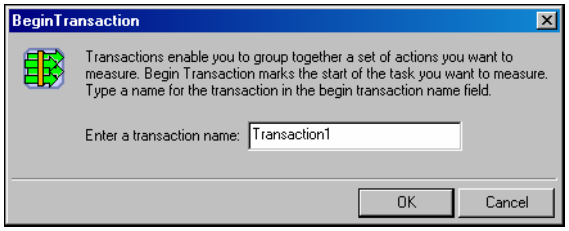
To measure transactions, you must mark the beginning and end of the transaction in your Agenda. During runtime, WebLOAD measures the time it takes to complete the transaction and reports the results in the WebLOAD Integrated reports, Statistics reports, and Data Drilling report.

Note: You can add an unlimited number of transactions into your Agenda, each with a different name.

To mark the beginning of a transaction:

1. Drag the Begin Transaction icon from the Load toolbox into the Agenda Tree, directly above the first action you want to include in the transaction.

   The Begin Transaction dialog box opens.



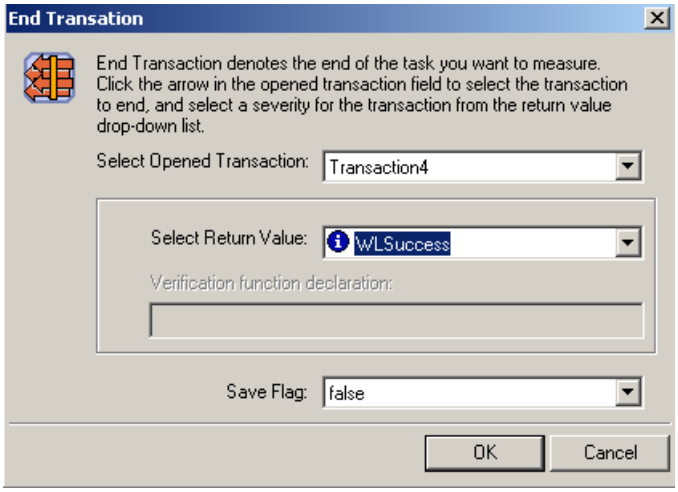
2. Enter a logical name for the transaction; for example, Login.
3. Click OK.

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

To mark the end of a transaction:

1. Drag the End Transaction icon from the Load toolbox into the Agenda Tree, directly after the last action you want included in the Agenda.

   The End Transaction dialog box opens.



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 *TestView Programmer's 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 Flag field. Select False (default) to save only results of failed transaction instances that triggered some sort of error flag.
5. Click OK.

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

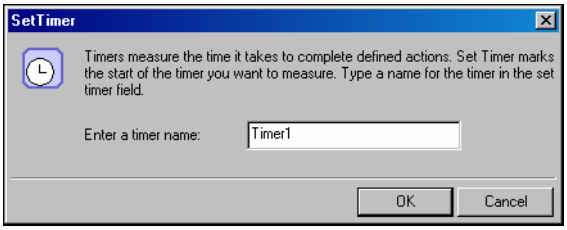
Set and Send Timer

Timers let you time any operation or group of operations in an Agenda 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 an Agenda directly through the WebLOAD IDE GUI.

To mark the beginning of a timer:

1. Drag the Set Timer  icon from the Load toolbox into the Agenda Tree directly before the first action you want to include in the timed task.

   The Set Timer dialog box opens.



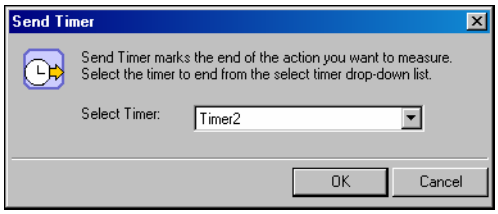
2. Type a name for the timer in the Timer Name field.
3. Click OK.

   The Set Timer item appears in the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

To mark the end of the timer:

1. Drag the Send Timer icon from the Load toolbox into the Agenda Tree directly after the last action you want included in the timed task.

   The Send Timer dialog box opens.



2. From the Select Timer drop-down list, to select the timer to end.
3. Click OK.

   The Send Timer item appears in the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda 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 an Agenda. 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 Agenda simultaneously.

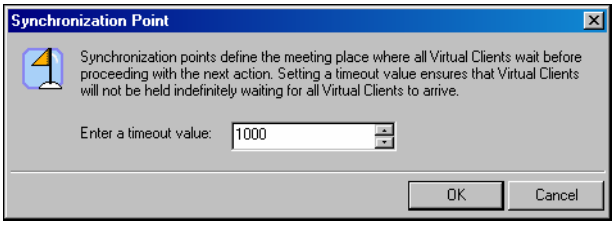
For example, suppose that you want to simulate 500 users, all trying to access a form on the same Web page simultaneously. To maximize the impact of this test situation, all 500 Virtual Clients must access the form at exactly the same time. Add a Synchronization Point before the form entry node to ensure that all the Virtual Clients log in simultaneously.

WebLOAD IDE 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 Agenda Tree directly before the action you want all Virtual Clients to perform simultaneously.

   The Synchronization Point dialog box opens.



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 Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

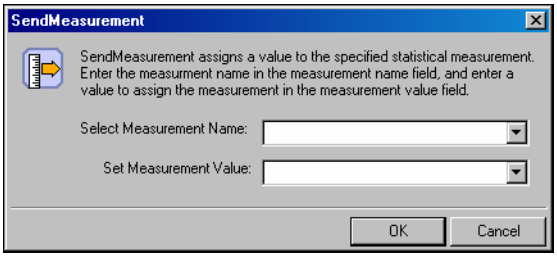
Send Measurement

WebLOAD IDE enables you to insert Send Measurement actions into your Agenda 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 Agenda Tree at the desired location.

   The Send Measurement dialog box opens.



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 Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

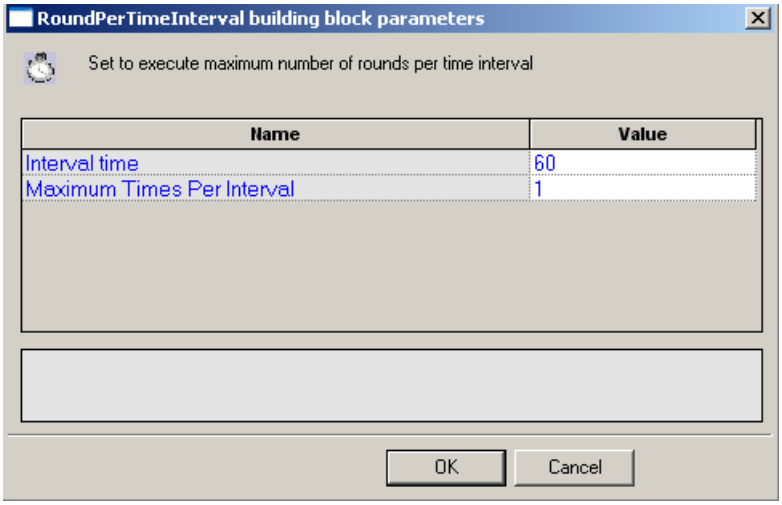
Round Per Time Interval

WebLOAD IDE enables you to specify the maximum to the number of times a round can be executed per time interval.

To define a round number maximum:

1. Drag the Round Per Time Interval  icon from the Load toolbox into the Agenda Tree at the desired location.

   The Round Per Time Interval dialog box opens.



2. In the Interval Time field, enter the interval time (in minutes).
3. In the Maximum Times Per Interval field, enter the maximum number of times that a round should be executed within the time interval
4. Click OK.

   The Round Per Time Interval building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

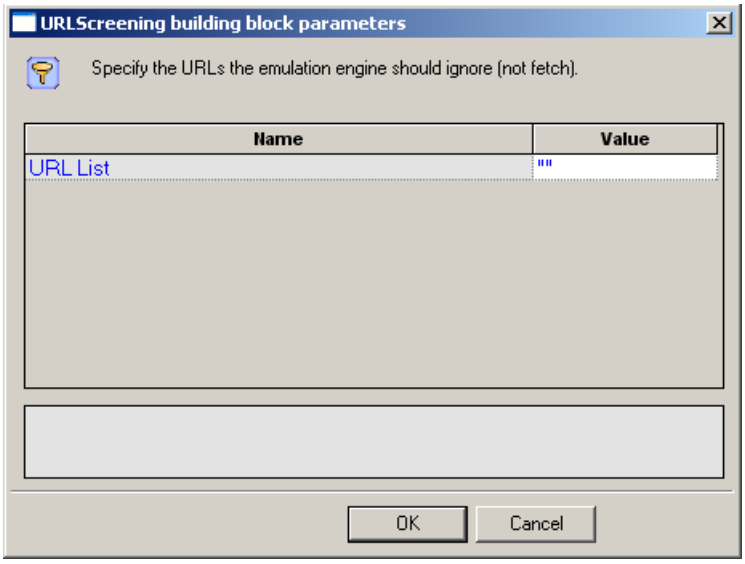
URL Screening

WebLOAD IDE enables you to add URL screening to an Agenda to define the URLs that the emulation 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 create and set the value for a measurement:

1. Drag the URL Screening  icon, from the Load toolbox, into the Agenda Tree at the desired location.

   The URL Screening dialog box opens.



2. Enter the URLs to ignore, separated by commas, in the Value field.
3. Click OK.
4. The URL Screening building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

Note: Fields that were not assigned a value in the dialog box are left as empty fields in the Agenda code.

Value Extraction

WebLOAD IDE enables you to add value extraction to an Agenda to define the

To create and set the value for a measurement:

1. Drag the Value Extraction  icon, from the Load toolbox, into the Agenda Tree at the desired location.

   The Value Extraction dialog box opens.

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 agenda.
6. Click OK.

   The Value Extraction building block added to the Agenda Tree. The JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

Note: Fields that were not assigned a value in the dialog box are left as empty fields in the Agenda code.

The WebLOAD IDE Database Toolbox

The WebLOAD IDE Database Toolbox includes a complete set of Database Utility Building Blocks. Use the WebLOAD IDE Database Utility Building Blocks to simply and easily add database activities to your test session Agenda.

To add database building blocks to a test Agenda directly through the WebLOAD IDE GUI:

- Drag the selected database utility icon from the Database toolbox and drop it into the Agenda Tree at the appropriate point.

The following database activity dialog boxes are described here:

- OpenDB
- Oracle OpenDB
- Execute
- Fetch Data
- DB GetLine
- Oracle DB GetLine
- DB Load
- Oracle DB Load

Each database utility 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: 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 Agenda code.

Once you have finished defining the new database activity, the new activity is reflected in the Agenda Tree. A database utility icon is added to the Agenda Tree for each database activity defined. WebLOAD IDE automatically adds the corresponding JavaScript code to your test session Agenda.

To see the complete sequence of JavaScript code for all the Database Utility building blocks that have been added to the Agenda tree, click the Agenda root node in the Agenda tree and select the JavaScript view tab.

Notes:
The JavaScript code for each of the Database Utility building blocks can be found in the `DBBuildingBlocks.js` library file, `which` is part of the `Include` directory under the TestView Suite installation directory. The JavaScript code that implements these Database Utilities is automatically inserted to the appropriate locations within the Agenda script. Code lines may be added to the initialization phase (within the `InitAgenda()` function), in the main body of the Agenda, 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.

The field descriptions in this section assume a basic familiarity with database terminology. To take full advantage of the Database Utility building blocks, testers must understand how to work with ADO objects and have a basic knowledge of SQL command syntax. WebLOAD IDE automatically inserts into the test session Agenda 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.

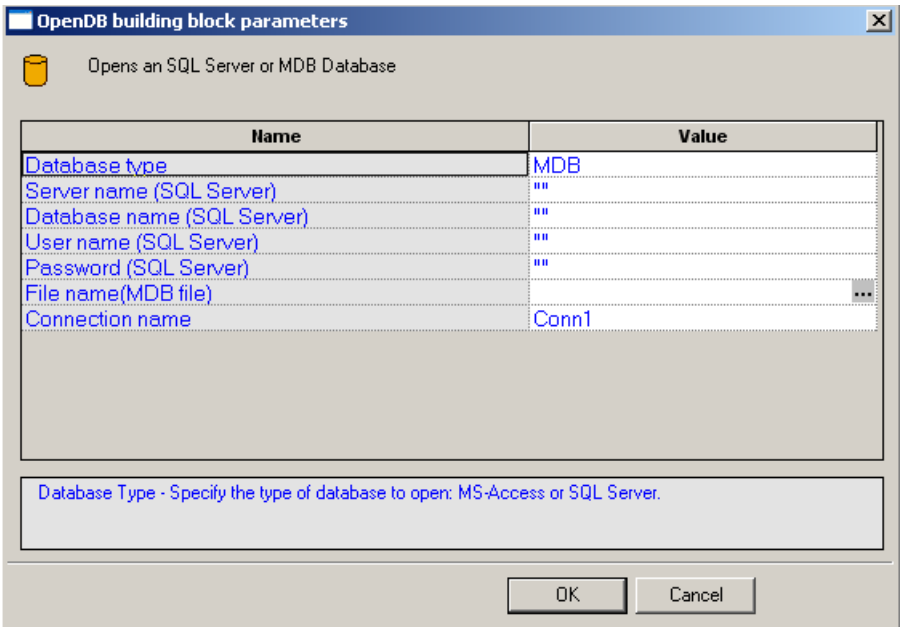
OpenDB

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 Agenda Tree at the desired location.

   The OpenDB building block parameters dialog box opens.



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 in the table below.

   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 added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda 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:

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	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	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	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	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	Specify the full path for an MDB file. Select the appropriate file from the Browser window that appears when you click the browse button 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 Agenda file to access and work with this database.

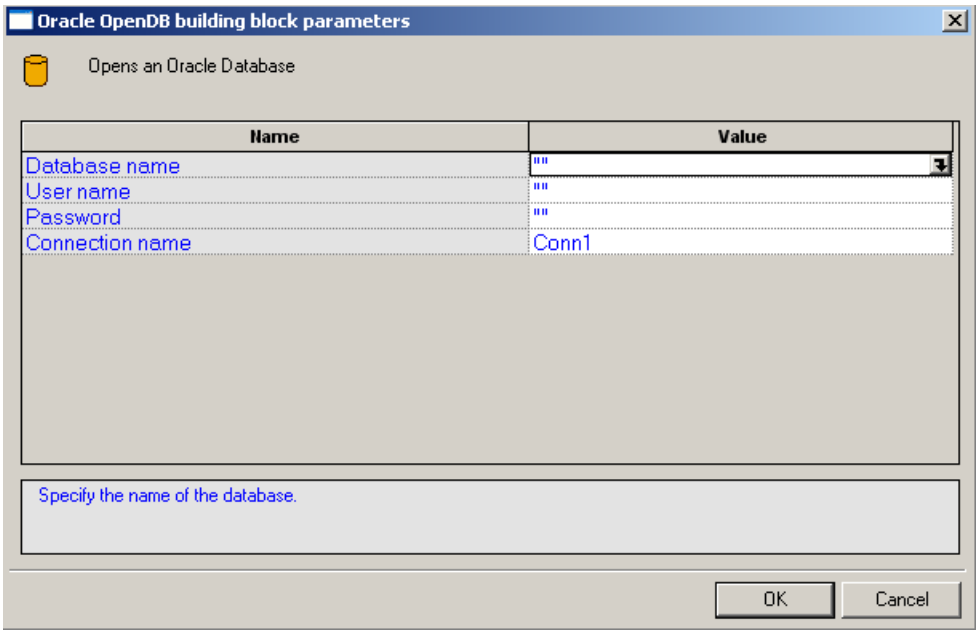
Oracle OpenDB

Use the Oracle OpenDB building block to open and close an Oracle database.

To enter a value:

1. Drag the Oracle OpenDB icon from the Database toolbox into the Agenda Tree at the desired location.

   The OpenDB building block parameters dialog box opens.



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 in the table below.

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 added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda 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:

Field Name	Description
Database Name	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	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	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.
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 Agenda file to access and work with this database.

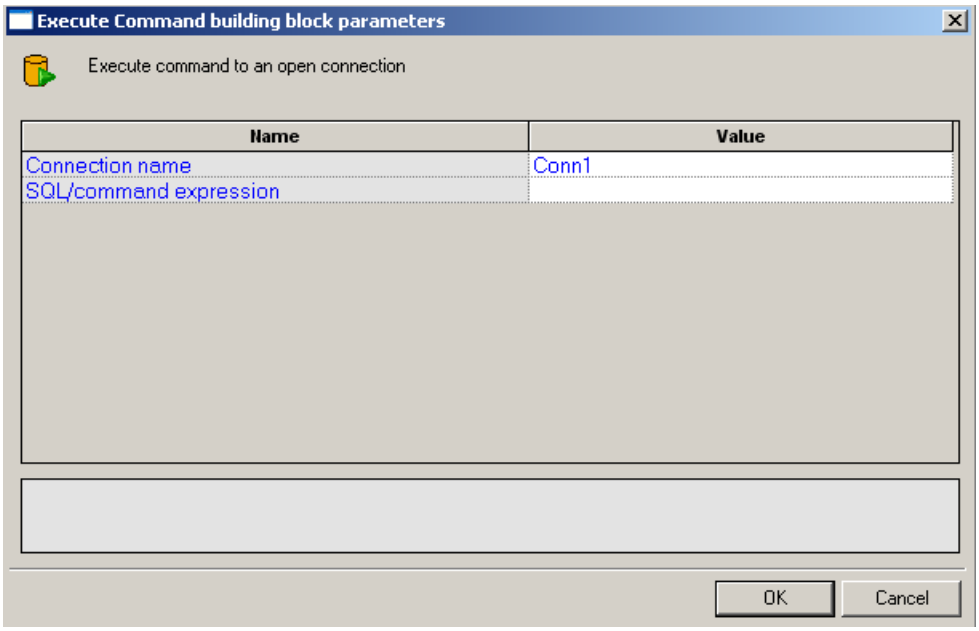
Execute Command

Use the Execute Command building block to add simple database commands to your test session Agenda. 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 Agenda Tree at the desired location.

   The Execute Command building block parameters dialog box opens.

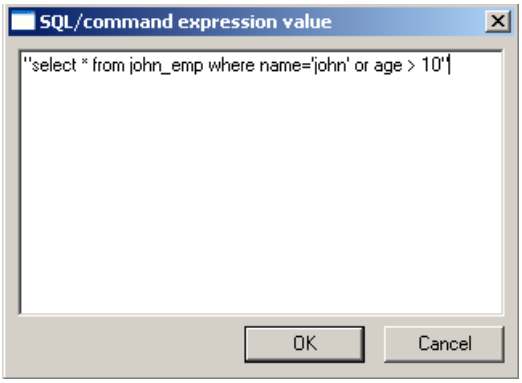


2. Click the name of an input field in the left-hand column to see an explanation of that field in the comment area at the bottom of the dialog box.

   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 in the table below.

   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.



4. Click OK .

   The Execute Command building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda 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:

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 Agenda 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.

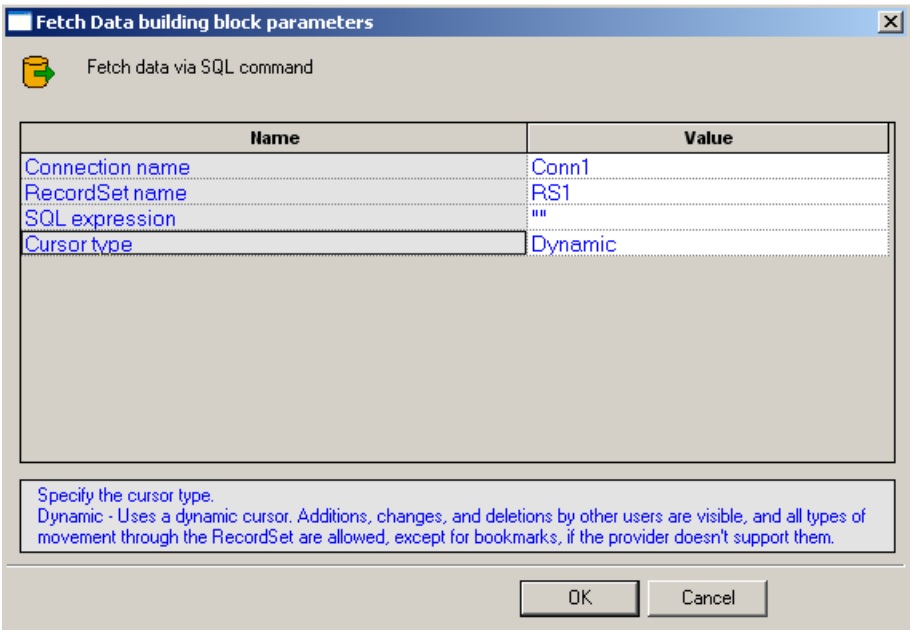
Fetch Data

Use the Fetch Data building block to add database commands that return data values to the Agenda. 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 Agenda Tree at the desired location.

   The Fetch Data building block parameters dialog box opens.



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 activity.

3. Enter the appropriate field value into the Value column next to the field name, as described in the table below.

   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 added to the Agenda Tree. The JavaScript code, including the `TerminateClient()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The fields in the Fetch Data building block parameters dialog box are described in the following table:

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 Agenda 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 Agenda 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 activity. 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

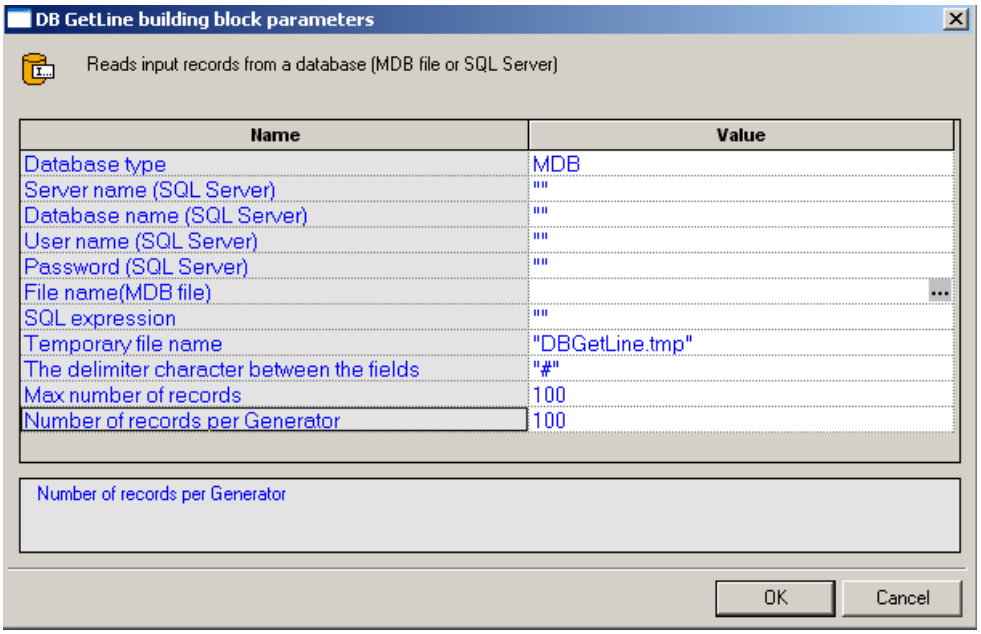
command. To read large amounts of input data from an MS-Access or SQL Server database, WebLOAD uses the equivalent DB GetLine database building block.

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 Agenda Tree at the desired location.

   The DB GetLine building block parameters dialog box opens.



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 the table below.
4. Click OK.

   The DB GetLine building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()` and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The fields in the DB GetLine building block parameters dialog box are described in the following table:

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	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	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	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	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	Specify the full path for an MDB file. Select the appropriate file from the Browser window that appears when you click the browse 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 Utility building blocks, testers must understand how to work with ADO objects and have a basic knowledge of SQL command syntax. WebLOAD IDE automatically inserts into the test session Agenda the appropriate JavaScript code to implement specified database commands. However, it is the tester who must specify the database commands to be inserted. WebLOAD IDE can not 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 Agenda. Data from this file is read line by line, where each data record is a separate line.
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.
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 IDE'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.

Oracle DB GetLine

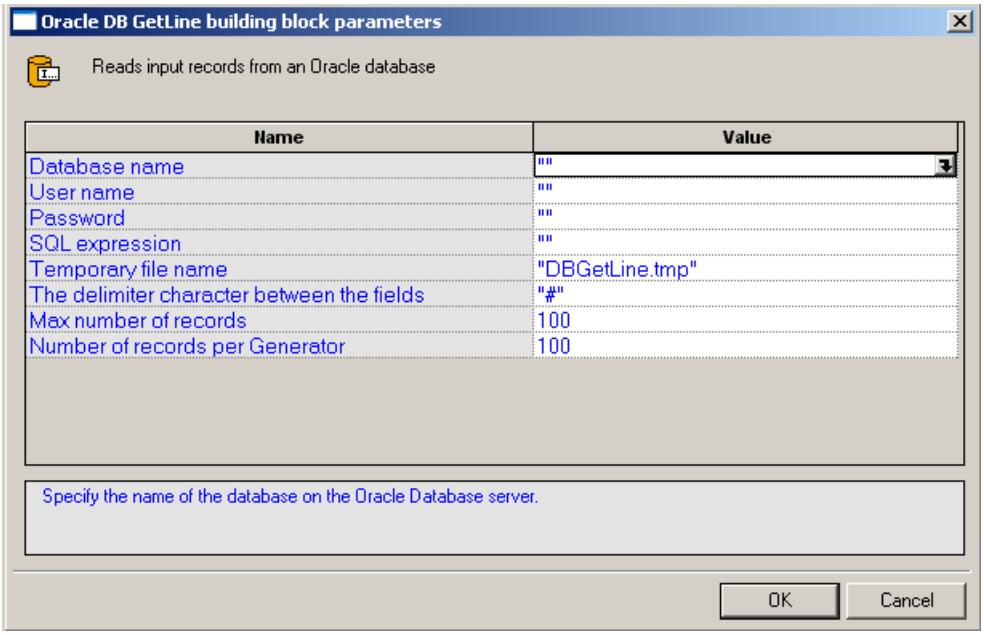
When running large load tests, the user input is usually read automatically from an input file. To read many rows of input data from a simple text file, WebLOAD uses the Oracle `GetLine()` I/O command. To read large amounts of input data from an Oracle database, WebLOAD uses the equivalent Oracle DBGetLine 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.

To enter a value:

1. Drag the Oracle DB GetLine icon from the Database toolbox into the Agenda Tree at the desired location.

   The Oracle DB GetLine building block parameters dialog box opens.



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 the table below.
4. Click OK.

   The Oracle DB GetLine building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()` and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The fields in the Oracle DB GetLine building block parameters dialog box are described in the following table:

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 Utility building blocks, testers must understand how to work with ADO objects and have a basic knowledge of SQL command syntax. WebLOAD IDE automatically inserts into the test session Agenda the appropriate JavaScript code to implement specified database commands. However, it is the tester who must specify the database commands to be inserted. WebLOAD IDE can not 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 Agenda. Data from this file is read line by line, where each data record is a separate line.
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
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 IDE'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.

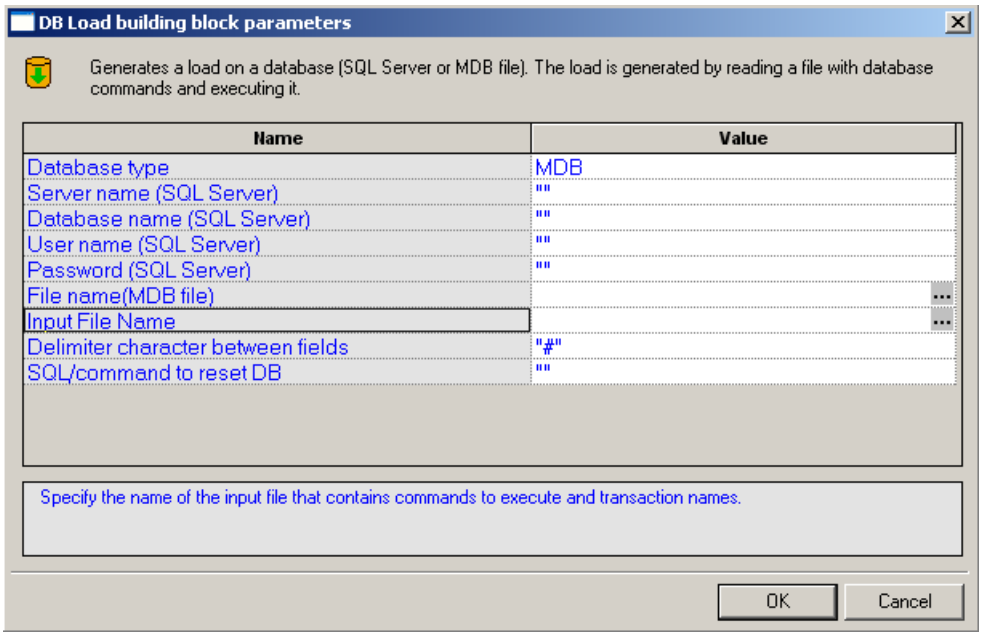
DB Load

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 IDE.

To enter a value:

1. Drag the DB Load icon from the Database toolbox into the Agenda Tree at the desired location.

   The DB Load building block parameters dialog box opens.



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 the table below.
4. Click OK.

   The DB Load building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The fields in the DB Load building block parameters dialog box are described in the following table:

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.

Field Name	Description
Server Name	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	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	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	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	Specify the full path for an MDB file. Select the appropriate file from the Browser window that appears when you click the browse button 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 the browse 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 Agenda 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.

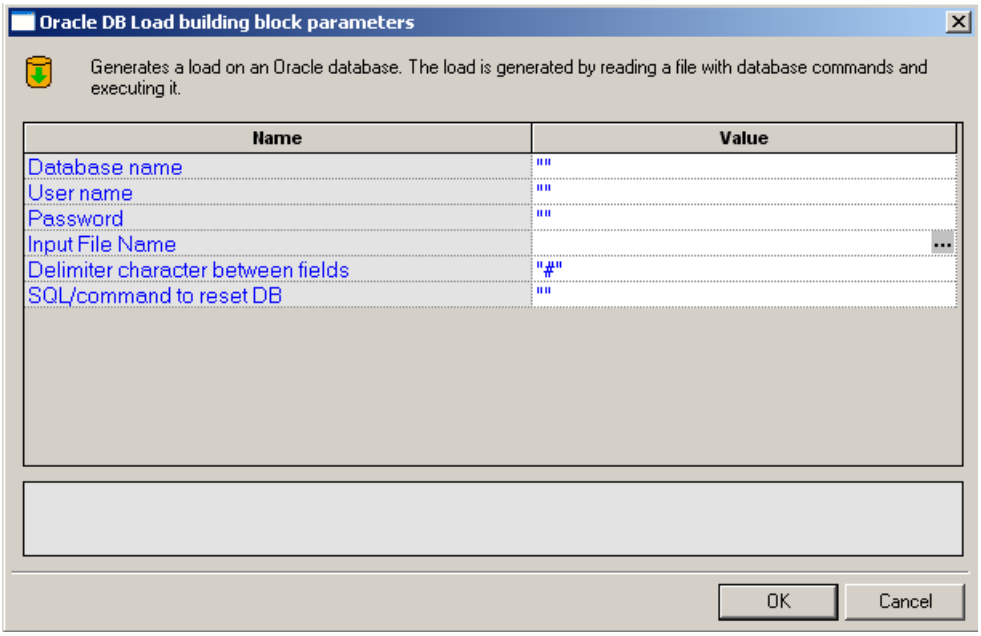
Oracle DB Load

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 IDE.

To enter a value:

1. Drag the Oracle DB Load icon from the Database toolbox into the Agenda Tree at the desired location.

   The Oracle DB Load building block parameters dialog box opens.



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 the table below.
4. Click OK.

   The Oracle DB Load building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The fields in the Oracle DB Load building block parameters dialog box are described in the following table:

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.

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 the browse 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 Agenda 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 IDE IPP Toolbox

Use the WebLOAD IDE IPP building blocks to simply and easily add IPP functionality to your test session Agenda without having to write numerous lines of code.

To add IPP building blocks to a test Agenda directly through the WebLOAD IDE GUI:

- Drag the selected IPP icon from the IPP toolbox and drop it into the Agenda Tree at the appropriate point.

  WebLOAD IDE automatically adds the appropriate JavaScript code to your test session Agenda.

WebLOAD IDE 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 IDE 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 IPP building blocks displayed in the IPP toolbox correspond to only a small part of the WebLOAD IDE IPP function set. These building blocks are provided for the most commonly used IPP activities. For a description of the complete set of IPP functions supported by WebLOAD IDE, see the *TestView Internet Protocols Reference* in the *TestView JavaScript Reference Manual*.

The following IPP dialog boxes are described here:

IPP Protocol	Building Blocks	
FTP		
	FTP-Connect (on page 191)	FTP-Upload (on page 193)

IPP Protocol	Building Blocks	
	 FTP-Download (on page 195)	 FTP-Disconnect (on page 197)
SMTP	 SMTP-Send Message (on page 197)	
POP	 POP-Retrieve (on page 200)	 POP-Delete (on page 202)
IMAP	 IMAP-Connect (on page 205)	 IMAP-Retrieve (on page 207)
	 IMAP-Delete (on page 208)	 IMAP-CreateMailbox (on page 210)
	 IMAP-ListMailboxes (on page 210)	 IMAP-DeleteMailbox (on page 212)
	 IMAP-Search (on page 213)	
NNTP	 NNTP-Connect (on page 217)	 NNTP-GetArticle (on page 219)

IPP Protocol	Building Blocks	
		
	NNTP-GetArticleCount (on page 221)	NNTP-PostArticle (on page 223)
TCP		
	TCP-Connect (on page 225)	TCP-Send (on page 227)
		
	TCP-Receive (on page 229)	TCP-Erase (on page 230)
TELNET		
	TELNET-Connect (on page 230)	TELNET-Receive (on page 232)
		
	TELNET-Send (on page 234)	TELNET-Erase (on page 235)
UDP		
	UDP-Bind (on page 236)	UDP-Broadcast (on page 238)
		
	UDP-Receive (on page 239)	UDP-Send (on page 240)
		
	UDP-Erase (on page 241)	

Each IPP 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 Agenda code.

Once you have finished defining the new IPP activity, the new action is reflected in the Agenda Tree. An IPP icon is added to the Agenda Tree for each IPP activity defined. WebLOAD IDE automatically adds the corresponding JavaScript code to your test session Agenda.

To see the complete sequence of JavaScript code for all the IPP building blocks that have been added to the Agenda tree, click the Agenda root node in the Agenda tree.

Note: The JavaScript code for each of the IPP building blocks can be found in the IPP library files, part of the `Include` directory under the TestView Suite 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 Agenda 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.

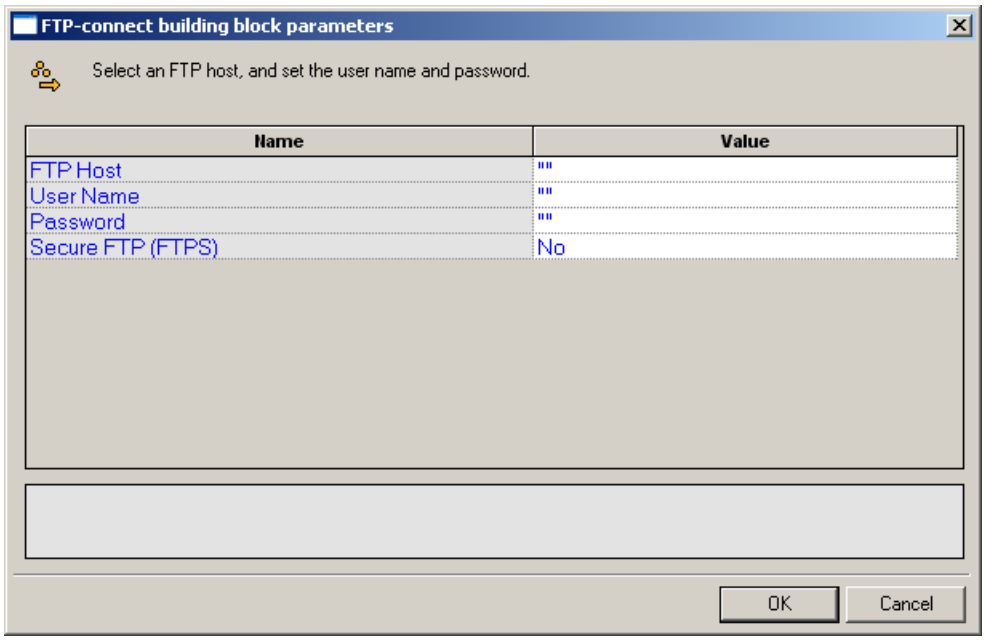
FTP-Connect

Use the FTP-Connect building block to open an FTP connection.

To enter a value:

1. Drag the FTP-Connect icon from the IPP toolbox into the Agenda Tree at the desired location.

   The FTP-Connect building block parameters dialog box opens.



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 IDE 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 in the table below.

4. Click OK.

   The FTP-Connect building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()` and `InitClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, the `InitAgenda()` function notes that the connection will be utilizing SSL security, and therefore includes the WebLOAD IDE 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 Agenda, 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:

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.

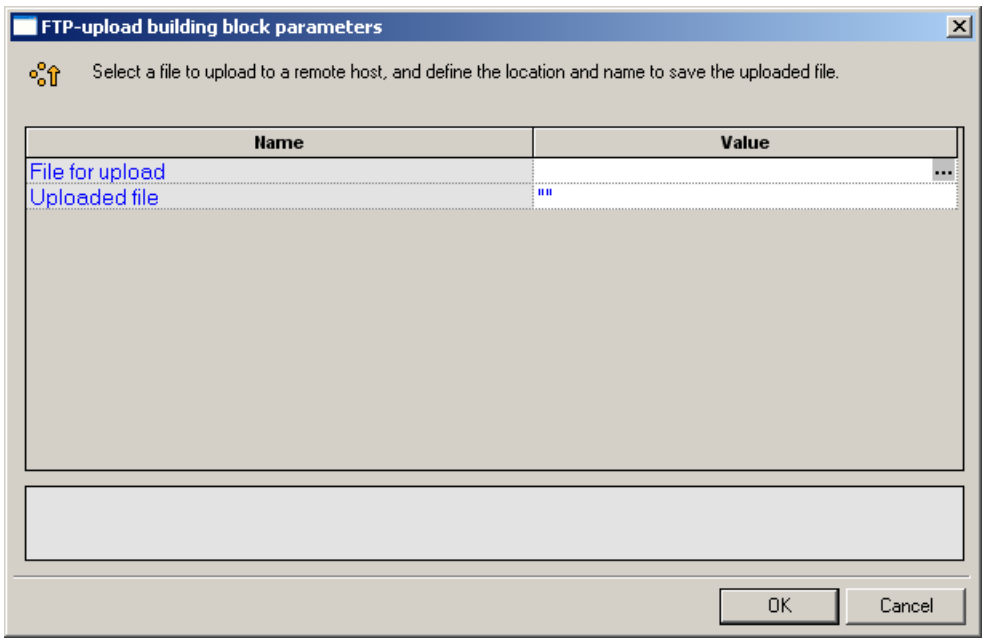
FTP-Upload

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 IPP toolbox into the Agenda Tree at the desired location.

   The FTP-Upload building block parameters dialog box opens.



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 in the table below.

   Note: If the Agenda 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 building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

Note: The WebLOAD IDE 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:

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 browse 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.

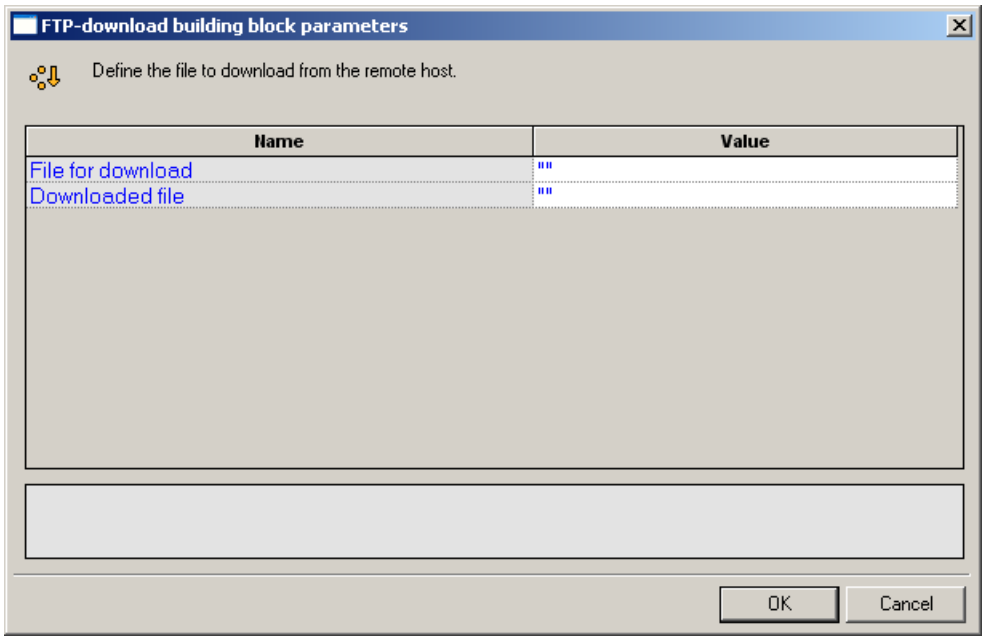
FTP-Download

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 IPP toolbox into the Agenda Tree at the desired location.

   The FTP-Download building block parameters dialog box opens.



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 in the table below.

   Note: If the Agenda 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 added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

In the Agenda, 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:

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 IPP toolbox into the Agenda Tree at the desired location.

  The FTP-Disconnect building block added to the Agenda Tree. The JavaScript code, including the `TerminateClient()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

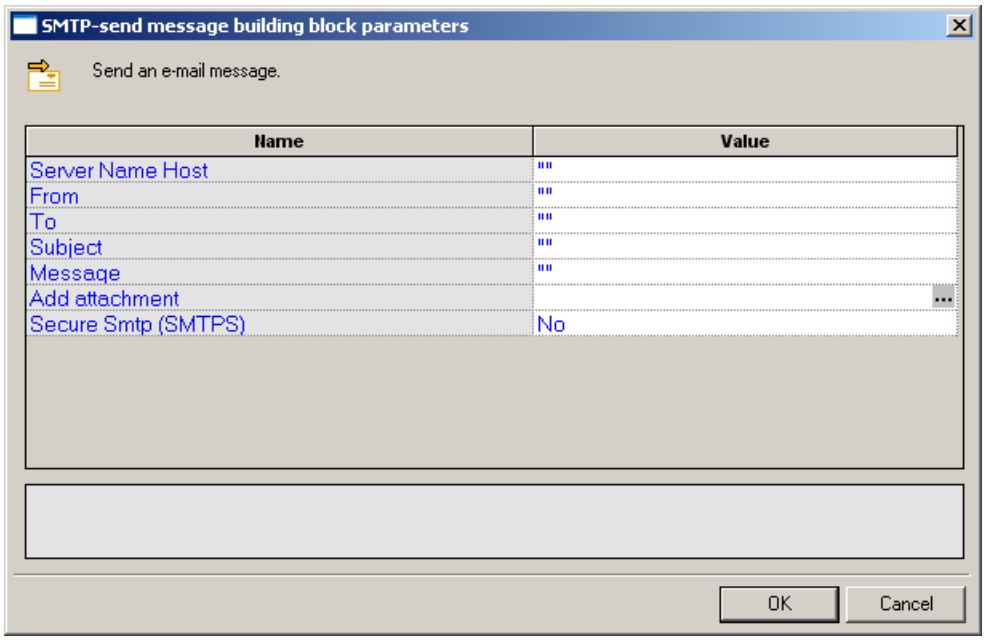
SMTP-Send Message

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 IPP toolbox into the Agenda Tree at the desired location.

   The SMTP-Send Message building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The SMTP-Send Message building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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.
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.
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.

Field Name	Description
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 browse 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 Agenda 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.

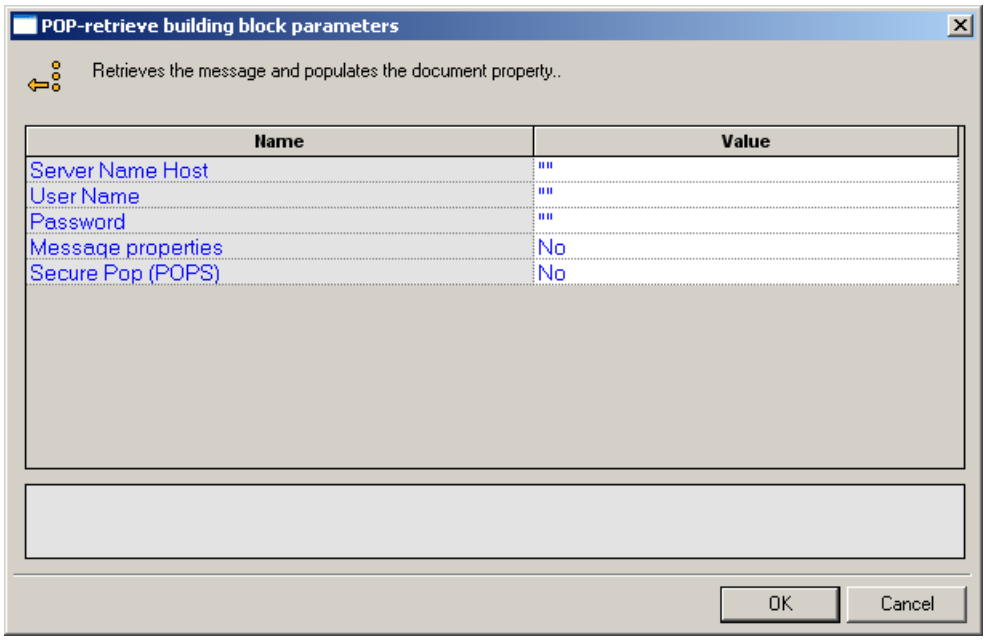
POP-Retrieve

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 IPP toolbox into the Agenda Tree at the desired location.

   The POP-Retrieve building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The POP-Retrieve building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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.

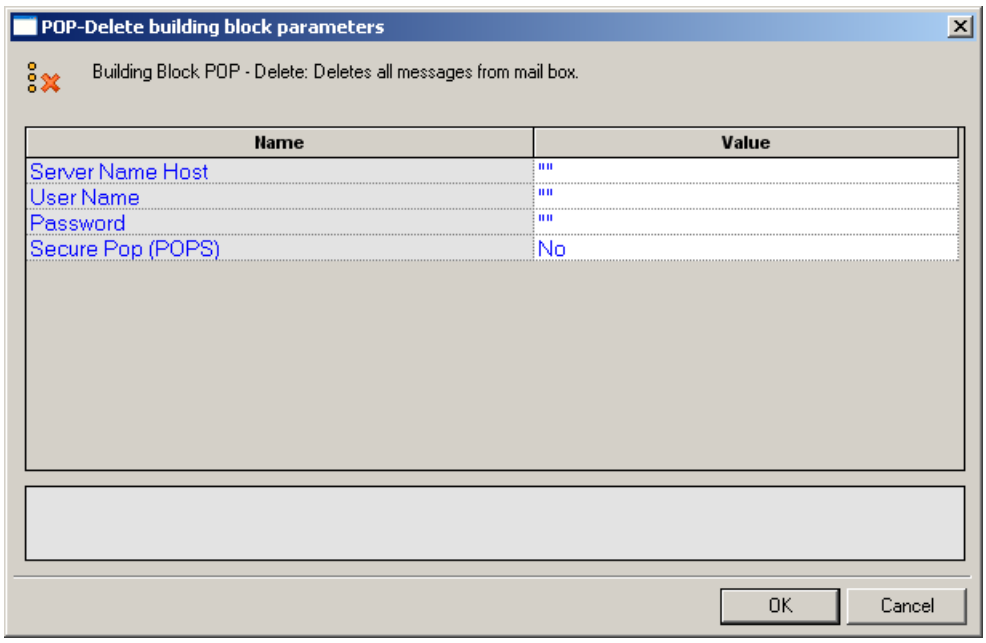
POP-Delete

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 IPP toolbox into the Agenda Tree at the desired location.

   The POP-Delete building block parameters dialog box opens.



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 IDE 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 the table below.

4. Click OK.

   The POP-Delete building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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 Agenda 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-Search: Search for a specific email item within an IMAP mailbox.

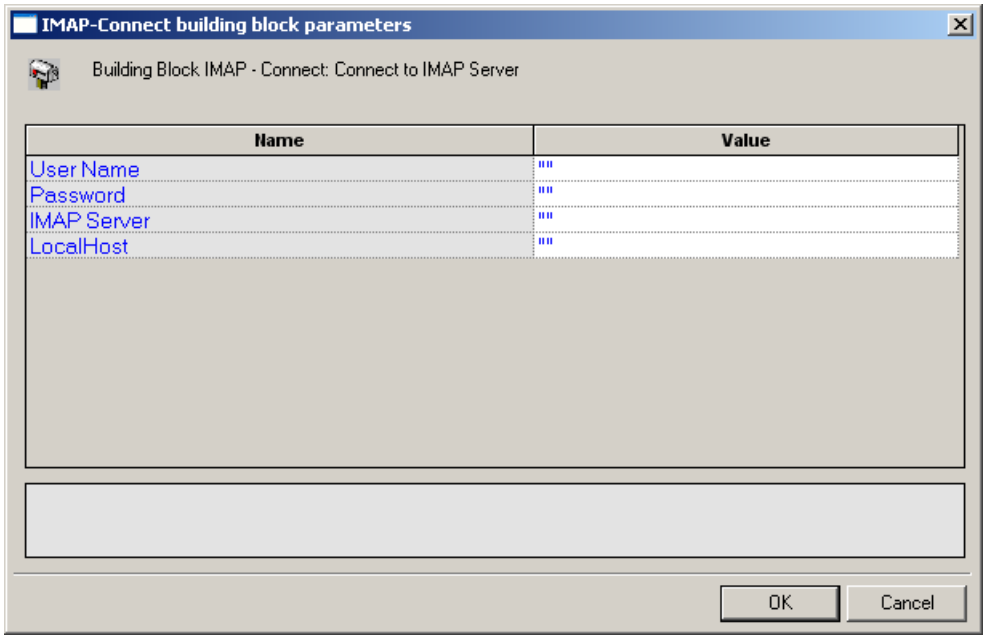
IMAP-Connect

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 IPP toolbox into the Agenda Tree at the desired location.

   The IMAP-Connect building block parameters dialog box opens.



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 IDE 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 in the table below.

4. Click OK.

   The IMAP-Connect building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

In the Agenda, 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:

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.
Local Host	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.

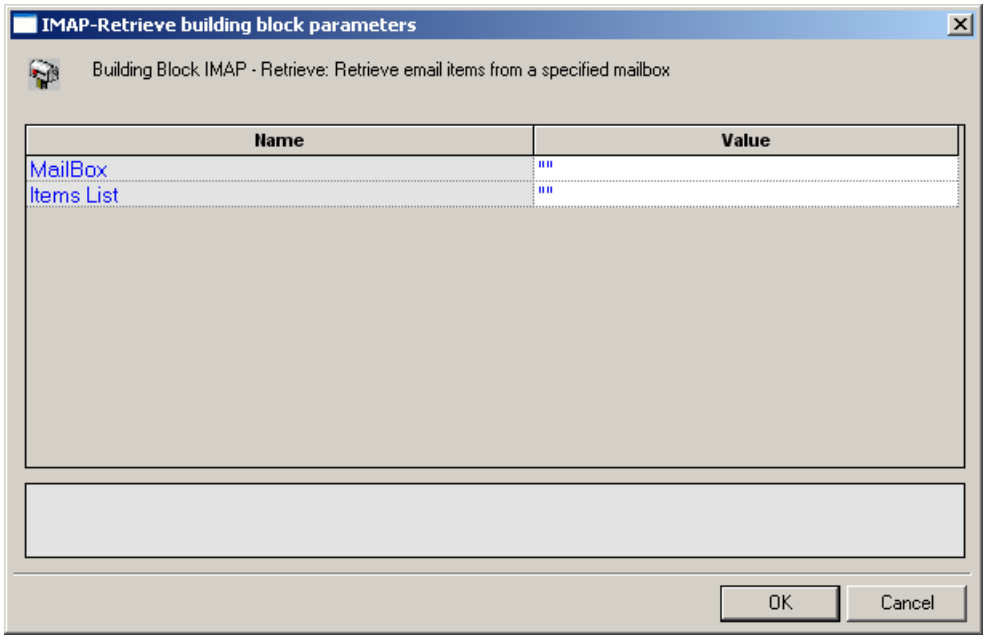
IMAP-Retrieve

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 IPP toolbox into the Agenda Tree at the desired location.

   The IMAP-Retrieve building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The IMAP-Retrieve building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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.

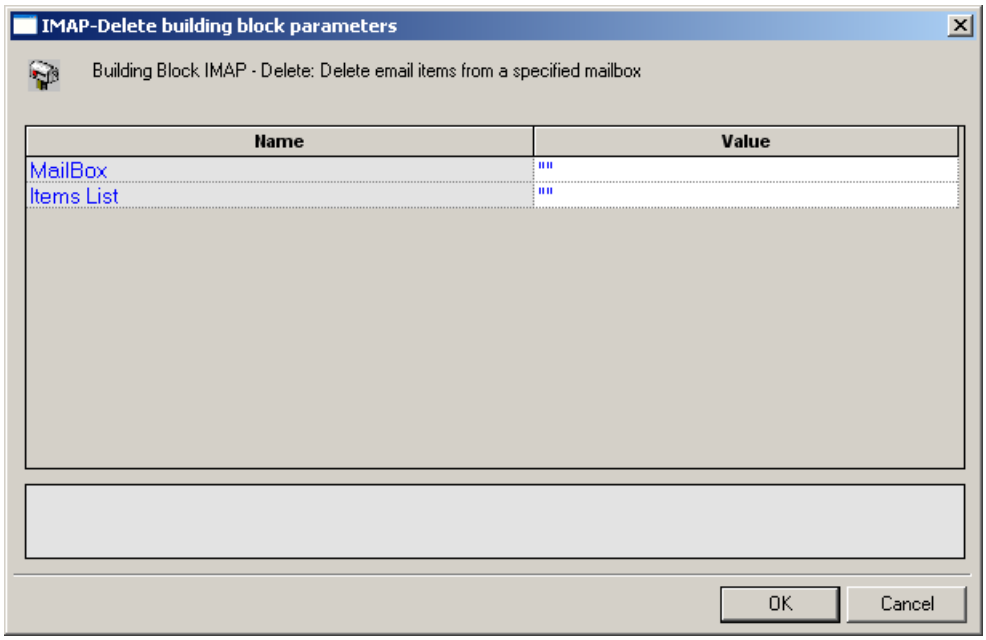
IMAP-Delete

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 IPP toolbox into the Agenda Tree at the desired location.

   The IMAP-Delete building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The IMAP-Delete building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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.
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-ListMailboxes

Use the IMAP-ListMailboxes building block to generate a complete list of all IMAP mailboxes accessed through the current IMAP server.

To enter a value:

- Drag the IMAP-ListMailboxes icon from the IPP toolbox into the Agenda Tree at the desired location.

  The IMAP-ListMailboxes building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

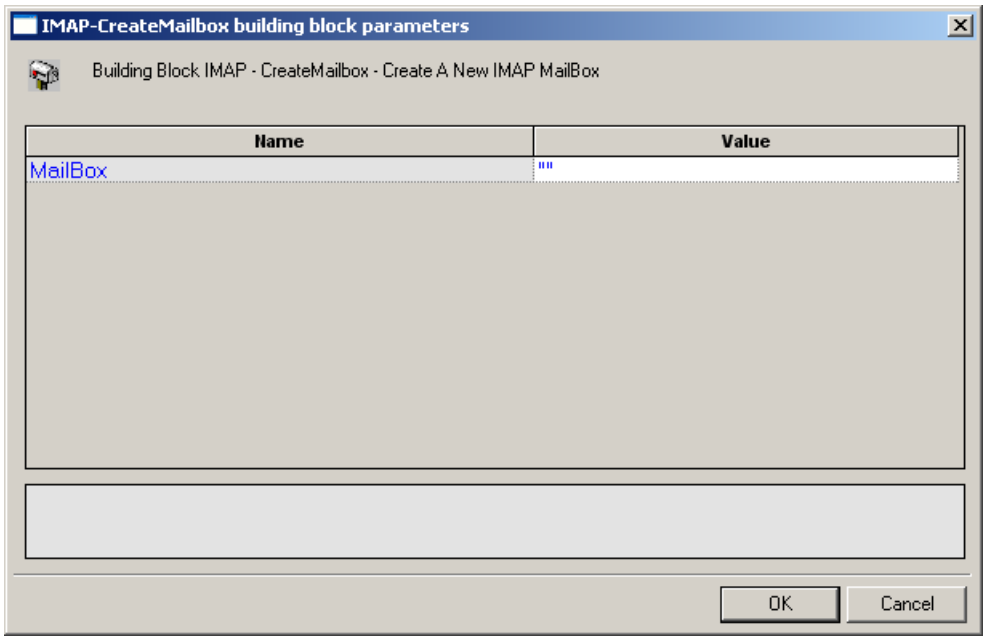
IMAP-CreateMailbox

Use the IMAP-CreateMailbox building block to create a new IMAP mailbox.

To enter a value:

1. Drag the IMAP-CreateMailbox icon from the IPP toolbox into the Agenda Tree at the desired location.

   The IMAP-CreateMailbox building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The IMAP-CreateMailbox building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, a new mailbox is created using the name specified by the user.

The fields in the IMAP-CreateMailbox building block parameters dialog box are described in the following table:

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.

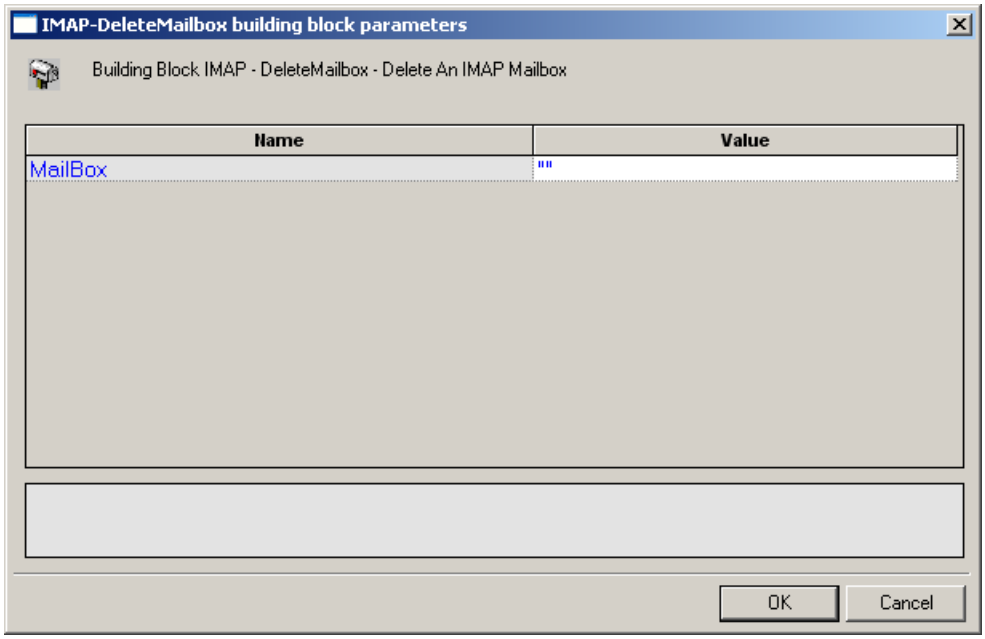
IMAP-DeleteMailbox

Use the IMAP-DeleteMailbox building block to delete an IMAP mailbox.

To enter a value:

1. Drag the IMAP-DeleteMailbox icon from the IPP toolbox into the Agenda Tree at the desired location.

   The IMAP-DeleteMailbox building block parameters dialog box opens.



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 in the table below.
4. Click OK.

   The IMAP-DeleteMailbox building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, the mailbox specified by the user is deleted.

The fields in the IMAP-DeleteMailbox building block parameters dialog box are described in the following table:

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.

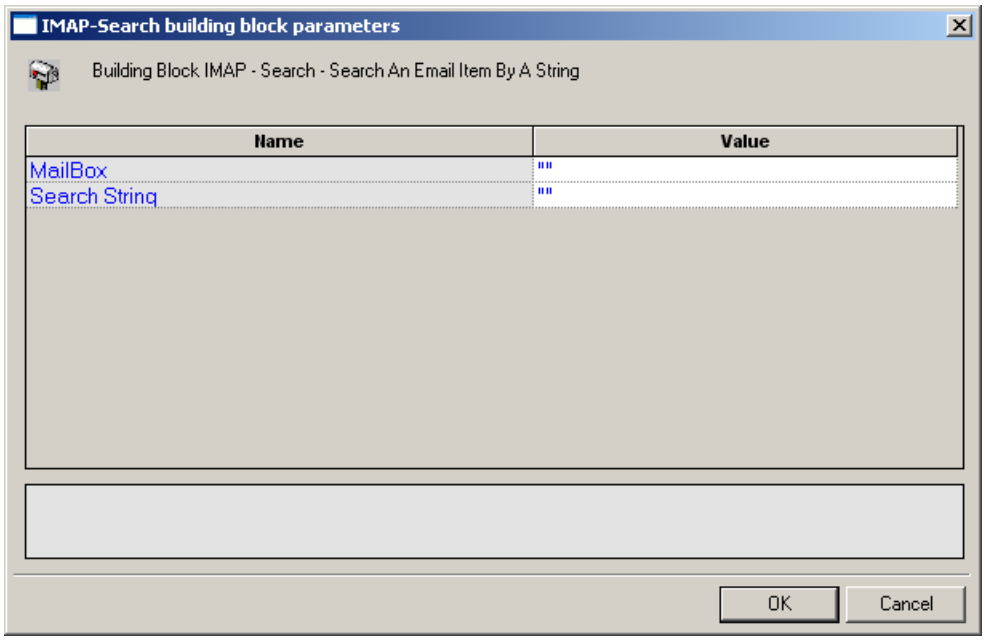
IMAP-Search

Use the IMAP-Search building block to search for a specific email item within an IMAP mailbox.

To enter a value:

1. Drag the IMAP-Search icon from the IPP toolbox into the Agenda Tree at the desired location.

   The IMAP-Search building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The IMAP-Search building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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. 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.

Field Name	Description
	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.

Field Name	Description
	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. 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 Agenda 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.

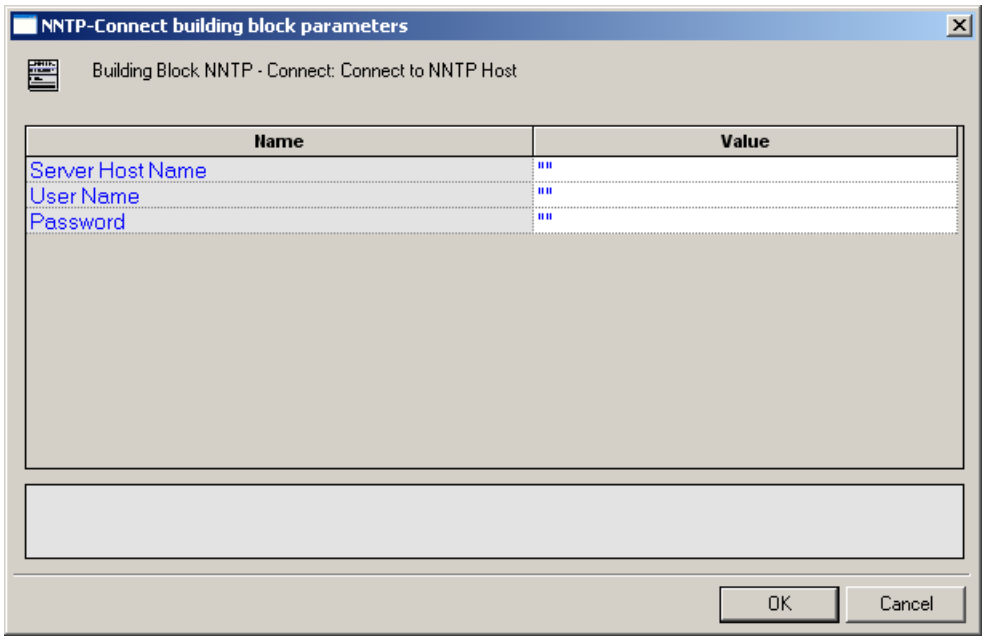
NNTP-Connect

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 IPP toolbox into the Agenda Tree at the desired location.

   The NNTP-Connect building block parameters dialog box opens.



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 IDE 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 in the table below.

4. Click OK.

   The NNTP-Connect building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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.

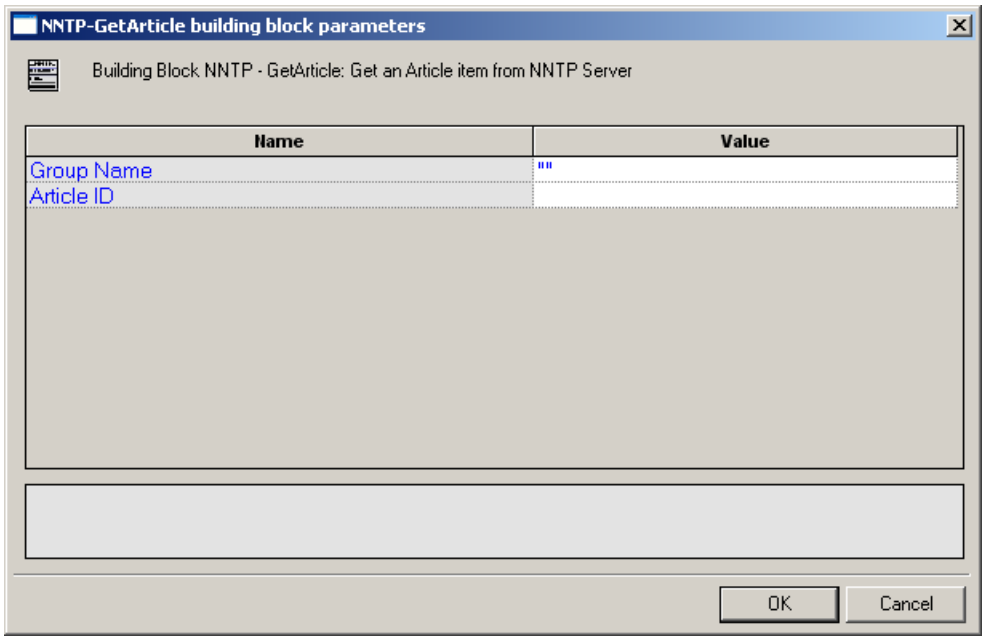
NNTP-GetArticle

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 IPP toolbox into the Agenda Tree at the desired location.

   The NNTP-GetArticle building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The NNTP-GetArticle building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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.

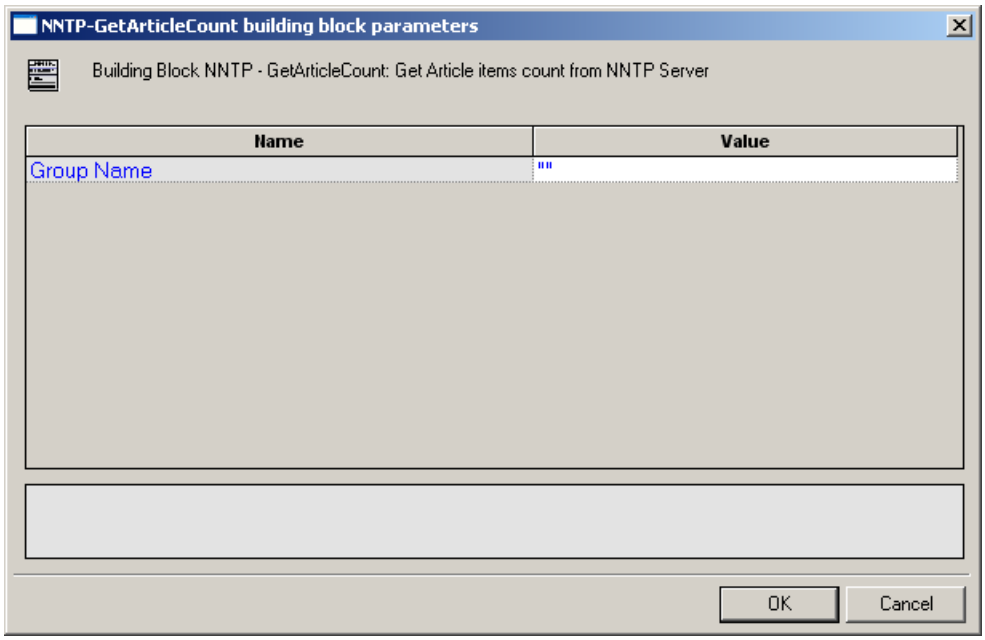
NNTP-GetArticleCount

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 IPP toolbox into the Agenda Tree at the desired location.

   The NNTP-GetArticleCount building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The NNTP-GetArticleCount building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, the number of articles appearing in the specified news group is returned.

The fields in the NNTP-GetArticleCount building block parameters dialog box are described in the following table:

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.

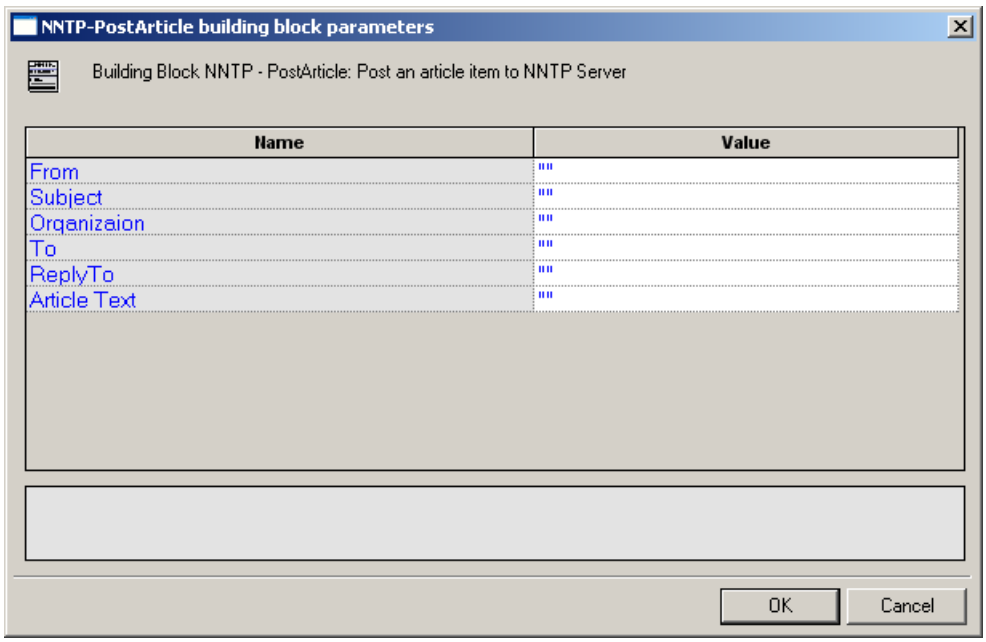
NNTP-PostArticle

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 IPP toolbox into the Agenda Tree at the desired location.

   The NNTP-PostArticle building block parameters dialog box opens.



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 in the table below.
4. Click OK.

   The NNTP-PostArticle building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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.
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 browse button to the right of the Value input area for this field.

Field Name	Description
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.

TCP

Dragging a TCP icon into your Agenda 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.

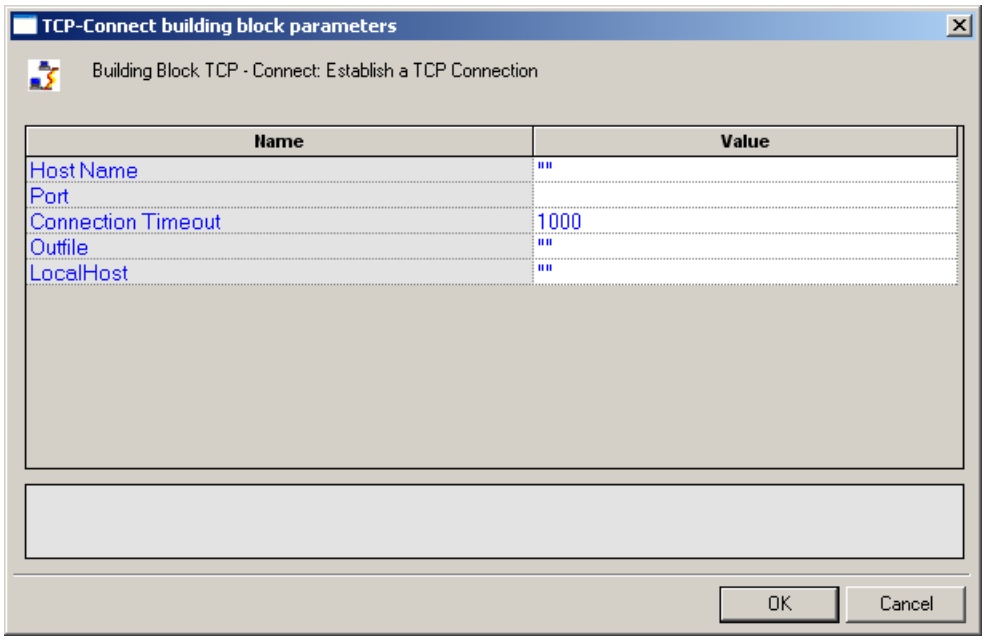
TCP-Connect

Use the TCP-Connect building block to open a TCP connection.

To enter a value:

1. Drag the TCP-Connect icon from the IPP toolbox into the Agenda Tree at the desired location.

   The TCP-Connect building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The TCP-Connect building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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.
Local Host	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.

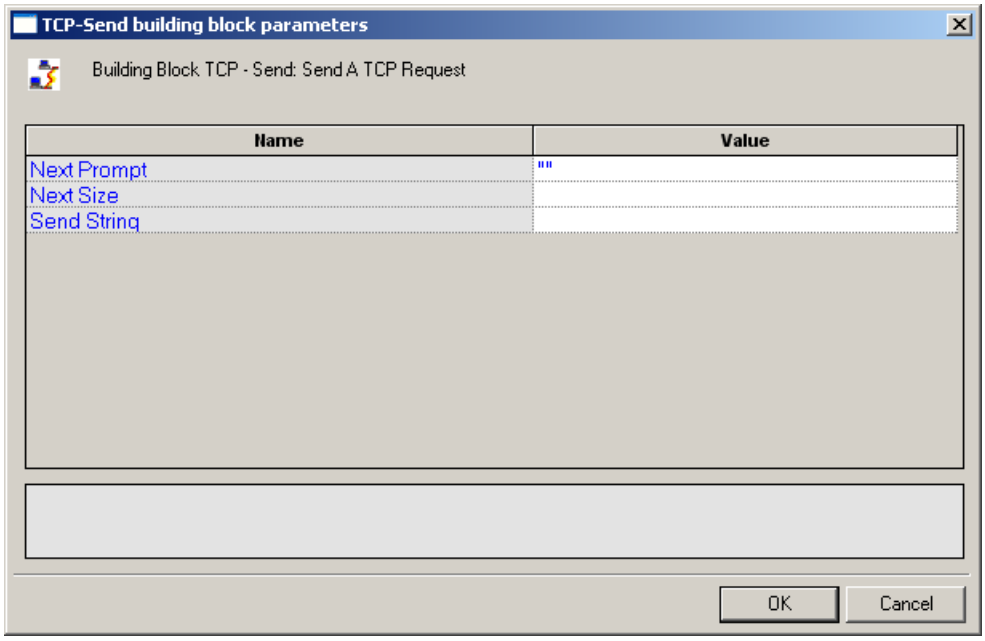
TCP-Send

Use the TCP-Send building block to send a TCP request.

To enter a value:

1. Drag the TCP-Send icon from the IPP toolbox into the Agenda Tree at the desired location.

   The TCP-Send building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The TCP-Send building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The fields in the TCP-Send building block parameters dialog box are described in the following table:

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 Agenda 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 Agenda 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 IPP toolbox into the Agenda Tree at the desired location.

  The TCP-Receive building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda 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 IPP toolbox into the Agenda Tree at the desired location.

  The TCP-Erase building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

TELNET

Dragging a TELNET icon into your Agenda 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.

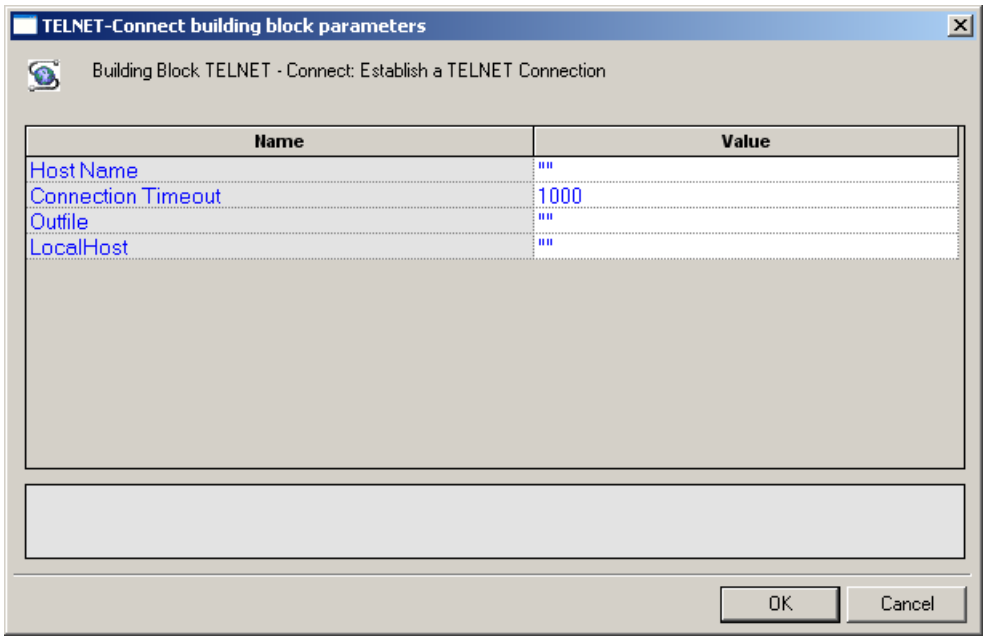
TELNET-Connect

Use the TELNET-Connect building block to open a TELNET connection.

To enter a value:

1. Drag the TELNET-Connect icon from the IPP toolbox into the Agenda Tree at the desired location.

   The TELNET-Connect building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The TELNET-Connect building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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.
Local Host	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.

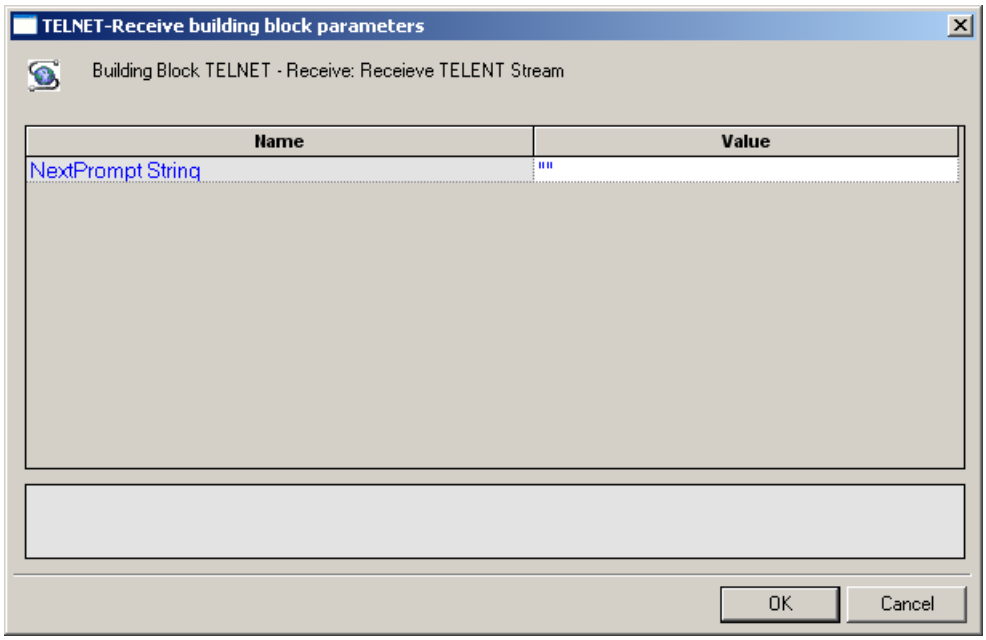
TELNET-Receive

Use the TELNET-Receive building block to receive a TELNET communication.

To enter a value:

1. Drag the TELNET-Receive icon from the IPP toolbox into the Agenda Tree at the desired location.

   The TELNET-Receive building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The TELNET-Receive building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The fields in the TELNET-Receive building block parameters dialog box are described in the following table:

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 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.

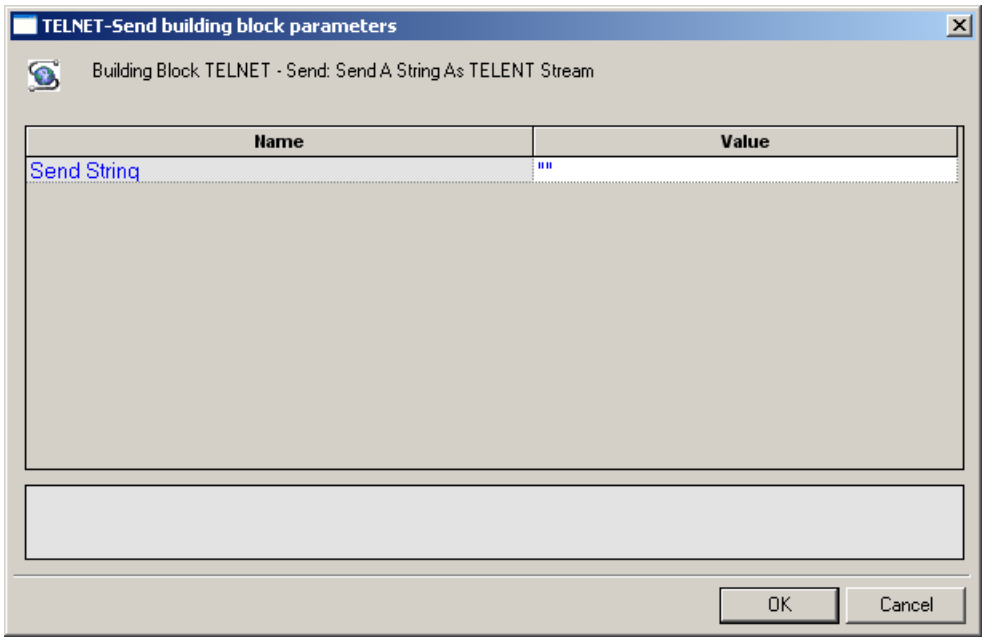
TELNET-Send

Use the TELNET-Send building block to send a TELNET communication.

To enter a value:

1. Drag the TELNET-Send icon from the IPP toolbox into the Agenda Tree at the desired location.

   The TELNET-Send building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The TELNET-Send building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The fields in the TELNET-Send building block parameters dialog box are described in the following table:

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 IPP toolbox into the Agenda Tree at the desired location.

  The TELNET-Erase building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

UDP

Dragging a UDP icon into your Agenda 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.

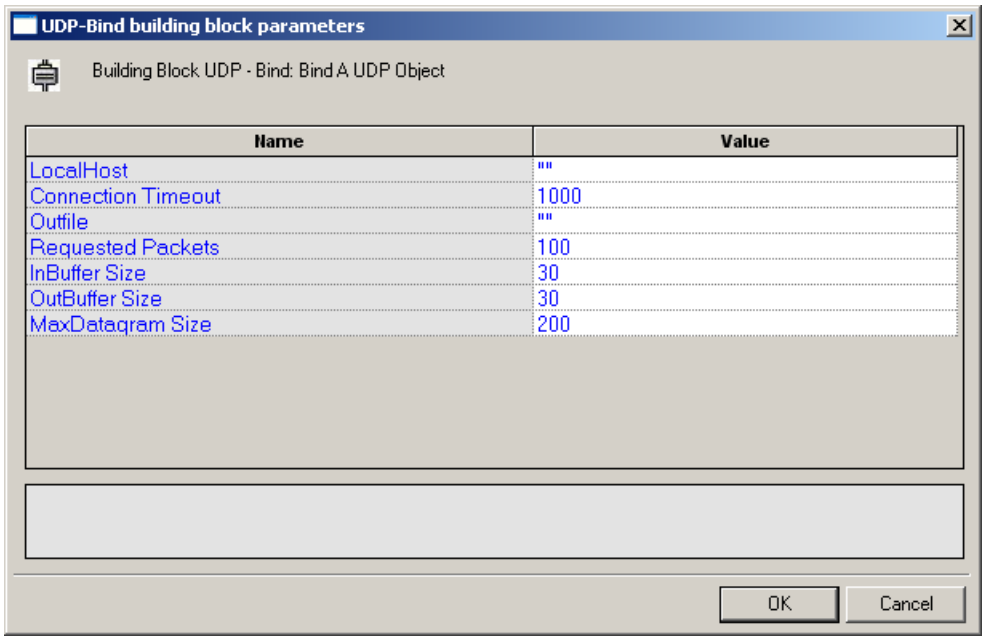
UDP-Bind

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 IPP toolbox into the Agenda Tree at the desired location.

   The UDP-Bind building block parameters dialog box opens.



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 in the table below.
4. Click OK.

   The UDP-Bind building block added to the Agenda Tree. The JavaScript code, including the `InitAgenda()`, `InitClient()`, and `TerminateClient()` functions, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, the `InitAgenda()` function includes commands to include the WebLOAD IDE 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 Agenda, 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:

Field Name	Description
Local Host	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.

Field Name	Description
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.

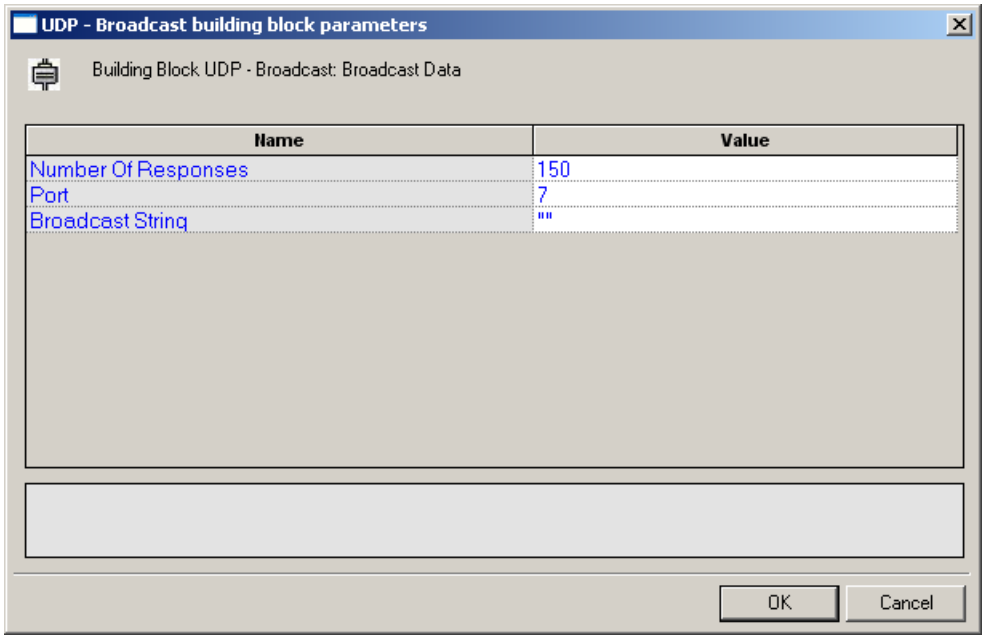
UDP-Broadcast

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 IPP toolbox into the Agenda Tree at the desired location.

   The UDP-Broadcast building block parameters dialog box opens.



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 in the table below.

4. Click OK.

   The UDP-Broadcast building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   In the Agenda, 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:

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 IPP toolbox into the Agenda Tree at the desired location.

  The UDP-Receive building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

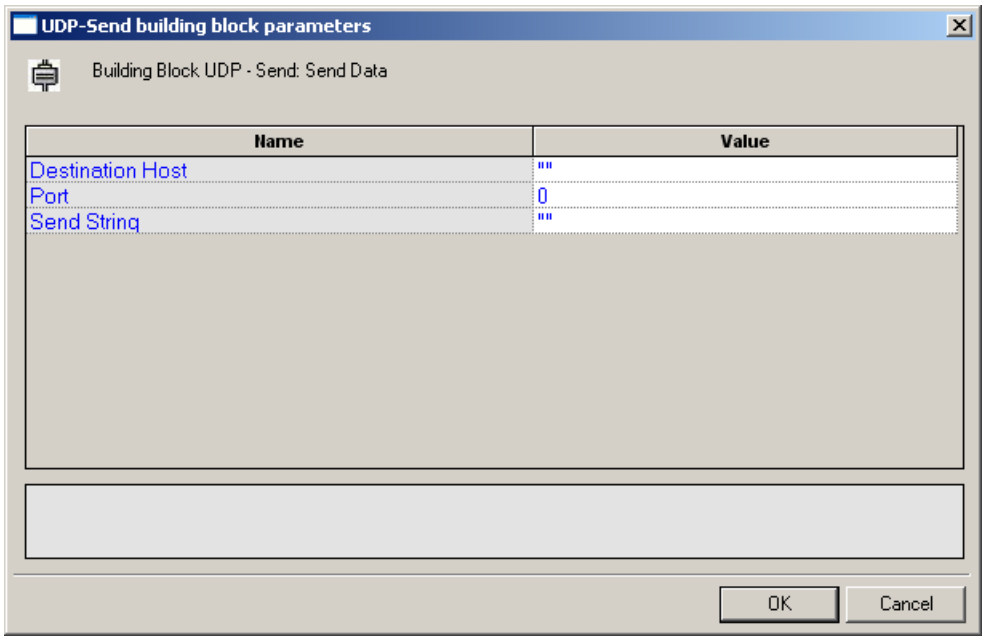
UDP-Send

Use the UDP-Send building block to send a UDP communication.

To enter a value:

1. Drag the UDP-Send icon from the IPP toolbox into the Agenda Tree at the desired location.

   The UDP-Send building block parameters dialog box opens.



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 in the table below.
4. Click OK.

   The UDP-Send building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The fields in the UDP-Send building block parameters dialog box are described in the following table:

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 IPP toolbox into the Agenda Tree at the desired location.

  The UDP-Erase building block added to the Agenda Tree and the JavaScript code is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

A P P E N D I X B

DDoS LOAD Testing

This section describes DDoS LOAD testing.

In This Appendix

Introducing DDoS LOAD

RadView's DDoS LOAD testing tool is designed to challenge your applications with a simulation of distributed denial of service (DDoS) attacks. DDoS LOAD uses a simple, intuitive interface that provides users with a comprehensive set of testing and verification tools literally at their fingertips, through point-and-click or drag-and-drop convenience.

This section provides a general introduction to application testing with DDoS LOAD.

Note: This section assumes a basic familiarity with WebLOAD IDE Agenda authoring tools and features. For more information about all the options available in WebLOAD IDE, see the rest of the *WebLOAD IDE User's Guide* and the *WebLOAD IDE Online Help*.

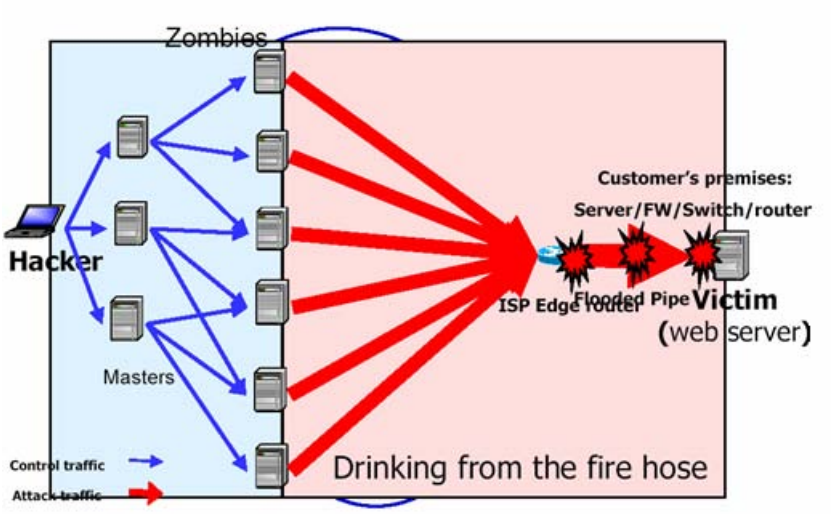
What is DDoS LOAD?

A distributed denial-of-service (DDoS) attack is one in which a multitude of compromised systems attack a target system. The flood of incoming messages to the target system essentially

forces it to slow or shut down completely, thereby denying service to legitimate users of the targeted system. DDoS LOAD is a performance and capacity assessment tool that challenges your Web applications to stand up to the load and complexity of distributed denial of service (DDoS) attacks. It delivers performance, realism, and ease of use in an easy-to-use tool. RadView's DDoS LOAD helps you evaluate your system (including hardware, defense tools, and software) resilience to real world DDoS attacks.

Popular Internet sites have been subjected to a form of cyber vandalism called distributed denial-of-service attacks, or DDoS attacks. DDoS attacks flood a network and overwhelm a target server with an immense volume of traffic that prevents normal users from accessing the server. The attacker breaks into a fleet of computers distributed around the internet and installs DDoS software on them, creating “zombie” or “agent” computers that unwittingly join forces to flood the victim server. The attacker then initiates a coordinated assault that repeatedly sends vast amounts of data packets from this network of “zombie computers” to flood a system or even a complete network-degrading performance or shutting it down completely. The attacks typically exhaust bandwidth, router processing capacity, or network stack resources, breaking network connectivity to the victims.

A typical attack scenario is illustrated in the following figure:



RadView's DDoS LOAD simulates various DDoS attacks in a controlled and safe way, enabling you to conduct tests to help your organization minimize the risk of a hacker causing damage. DDoS attack simulation is an essential tool when preparing your application for use. For example, DDoS attack simulation may help you determine whether or not you need to implement a DDoS defense tool. Defense devices can be set up to counteract DDoS attacks, but these devices can also serve as a bottleneck because of the burden placed on them to monitor all incoming and outgoing traffic. DDoS attack simulation while modeling peak conditions with high traffic volume can help you choose between the different defense solutions available in the market, using a common, objective tool that offers a common interface and common statistics to evaluate each one. DDoS LOAD serves as that common tool, while covering all current DDoS attacks, assessing the level of defense, and testing the Quality of Service (QoS) during an attack,

with the added benefit of support, maintenance, and ongoing updates that can handle the latest DDoS attack innovations.

DDoS LOAD testing includes the following features:

- DDoS LOAD is capable of simulating attacks of 300K to 500K packets per second (PPS) from a single load machine. Load machines can be infinitely aggregated. The system testers maintain full control of the PPS rate.
- System reports produced after a test session provide a clear analysis of all relevant information, including the PPS rate, the total number of packets sent, an evaluation of the Quality of Service (QoS) factors from the Probing Client component, and a complete set of server statistics through the PMM module.
- The WebLOAD IDE DDoS Toolbox provides a simple, intuitive means of adding DDoS LOAD testing components to your Agenda.
- The DDoS LOAD engine covers most of the known DDoS attacks as well as supporting simulation of generic (IDP, SYN, or ICMP) attacks.

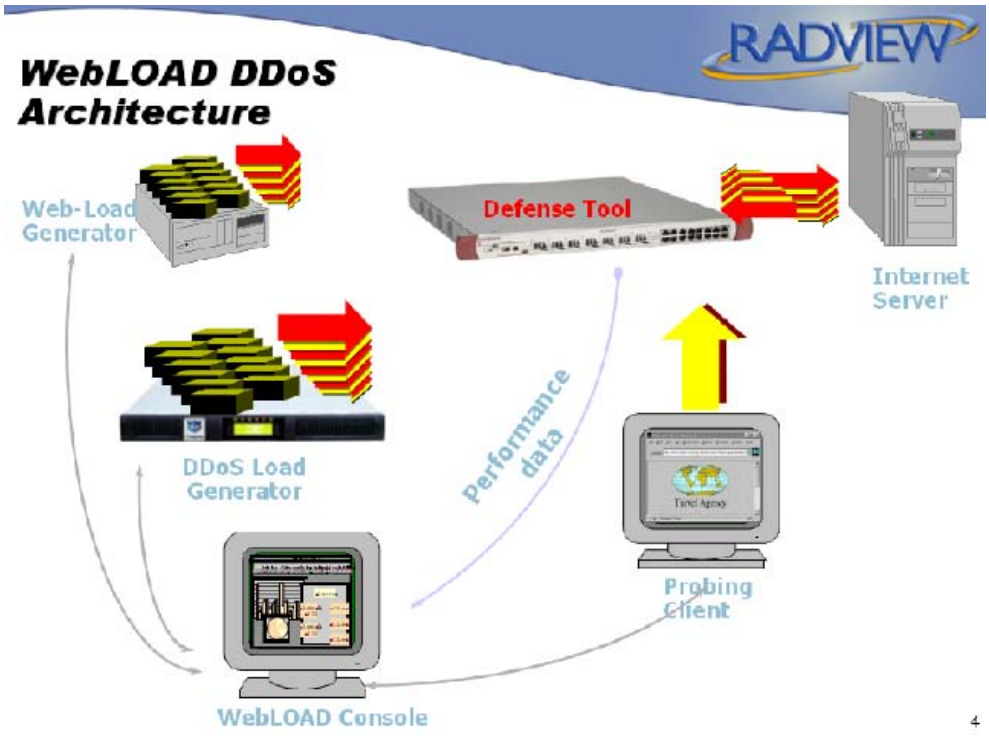
  Some of the DDoS attack tools supported include:

  - Blitznet
  - Carko
  - Flitz
  - Juno
  - Knight
  - LandAttack
  - Omega3
  - Plague
  - Stacheldracht4
  - Stacheldracht26
  - TFN
  - TFN2K
  - Trinoo

  See the release notes or the RadView web site for the most up-to-date list of the currently supported DDoS attack types.

DDoS LOAD Architecture

The following diagram illustrates the configuration for a typical DDoS LOAD test.



The WebLOAD Console sets up, runs and controls a test session. At the Console, you can define the hosts participating in the load test, specify the test scripts (Agendas) that the load test executes, schedule tests, and view performance reports.

During DDoS testing, WebLOAD tests the performance of your Web application under stress. WebLOAD uses the Probing Client component (a type of Virtual Client, emulating Web browser activities) to test for Quality of Service while under a simulated DDoS attack. The simulated DDoS attack is produced by the DDoS Load Generator. The DDoS Load Generator is a Linux machine that "bombards" the application being tested with a huge number of Packets Per Second (PPS), simulating the actions of a specific type of DDoS attack. The WebLOAD architecture enables testers to concatenate multiple Load Generators to generate practically infinite load levels. WebLOAD DDoS testing is thus able to test effectively for even a "worst case scenario".

WebLOAD records performance data collected throughout the course of a test session, producing a complete statistical analysis of the response time (and possible causes of potential bottlenecks) at each stage of the application being tested.

For example, a typical test strategy may include multiple Virtual Clients running a relatively simple Agenda to create a background load on a server, combined with a Probing Client running a more complex Agenda, programmed through the WebLOAD IDE, to precisely measure the

performance of specific accesses to the server, all being run concurrently with a high-intensity DDoS attack simulated by the DDoS Load Generator. Use the test reports produced by the WebLOAD Console to analyze how your web site would cope with being accessed by large numbers of users while under active malicious attack.

Programming DDoS Functionality into your JavaScript Agenda

The WebLOAD IDE DDOS Toolbox

The dialog boxes for the DDoS building blocks are documented in this section.

Blitznet

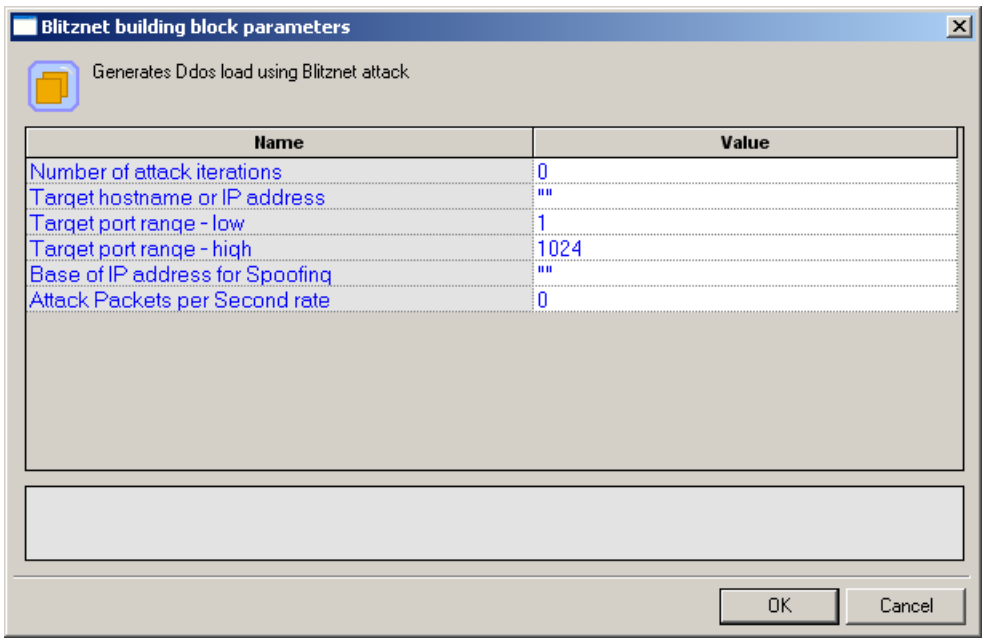
A Blitz Network launches a spoofed SYN flood attack via Slice2. The attack originates from many different computers without the user actually logging on to any of them. This deceptive attack sourcing is accomplished by loading the actual “attacker” daemon files onto a large number of attack computers, and then controlling all the attack computers through a camouflaged host computer.

Use the Blitznet building block to generate a Blitznet DDoS load test.

To enter a value:

1. Drag the Blitznet icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Blitznet building block parameters dialog box opens.



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 Number of Attack Iterations field is used to specify the number of packets to be sent per round.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The Blitznet building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the Blitznet DDoS attack method with the parameter values specified by the user, where the first parameter is the number of attack iterations (set to 60), the second parameter is a string with the target host name, the third and fourth parameters are lower and upper bounds of the target port number range, the last parameter is the base value for the spoofing address, and the PPS value of zero (0) reflects a field deliberately set to zero to take advantage of the default maximization options.

The fields in the Blitznet building block parameters dialog box are described in the following table:

Field Name	Description
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Range (low)	Specify the lower bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Port Range (high)	Specify the upper bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Base of IP Address for Spoofing	Specify a base value for an IP address that will be used to generate spoofed source addresses. For example, specifying “11.22.” will cause the DDoS attack to generate IP source addresses with that as the prefix value, i.e. of the format “11.22.*.*”. Type the appropriate IP 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.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

Carko

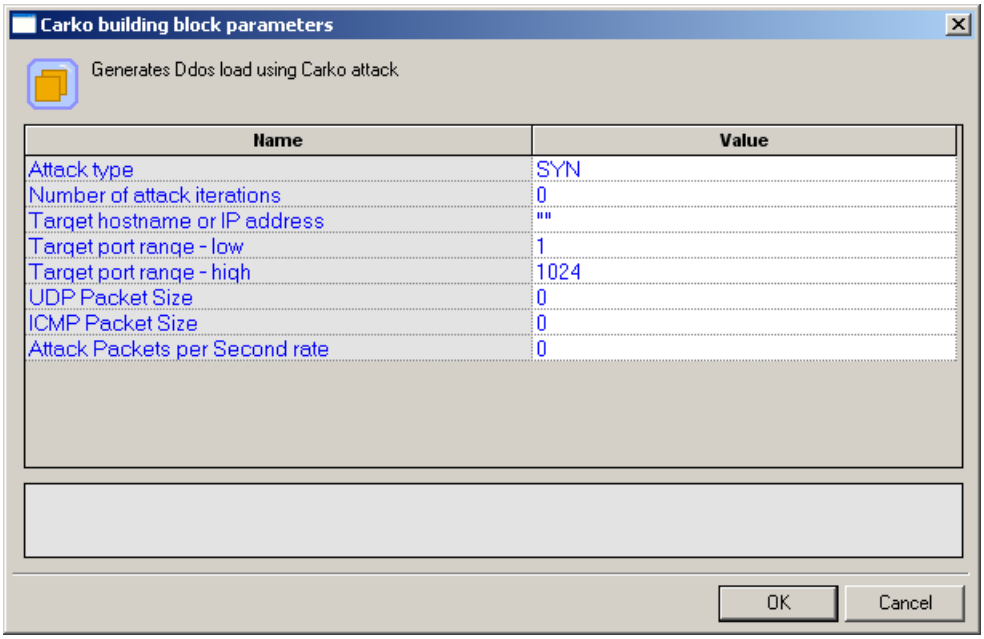
Carko is a DDoS attack that started to appear in the late summer of 1999 and combines features of Trinoo (on page 279) and TFN (on page 273). Carko also contains some advanced features, such as encrypted attacker-master communication and automated agent updates. The types of attacks possible are similar to those of TFN; ICMP flood, SYN flood, UDP flood, and SMURF attacks.

Use the Carko building block to generate a Carko DDoS load test.

To enter a value:

1. Drag the Carko icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Carko building block parameters dialog box opens.



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 Attack Type field is used to specify the type of Carko DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The Carko building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The Agenda calls the Carko DDoS attack method with the parameter values specified by the user, where the first parameter is the type of Stacheldraht4 attack to simulate (Random), the second parameter is the number of attack iterations (set to 72), the third parameter is a string with the target host name, and the PPS value of zero (0) reflects a field deliberately set to zero to take advantage of the default maximization options.

The fields in the Carko building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of Carko DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ SYN ◆ UDP ◆ ICMP ◆ Random ◆ Stream ◆ Havoc ◆ IP ◆ ACK ◆ NUL
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.

Field Name	Description
Target Port Range (low)	Specify the lower bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Port Range (high)	Specify the upper bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
UDP Packet Size	Specify the size of the UDP packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Relevant for UDP-type attacks only.
ICMP Packet Size	Specify the size of the ICMP packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Relevant for ICMP-type attacks only.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

Flitz

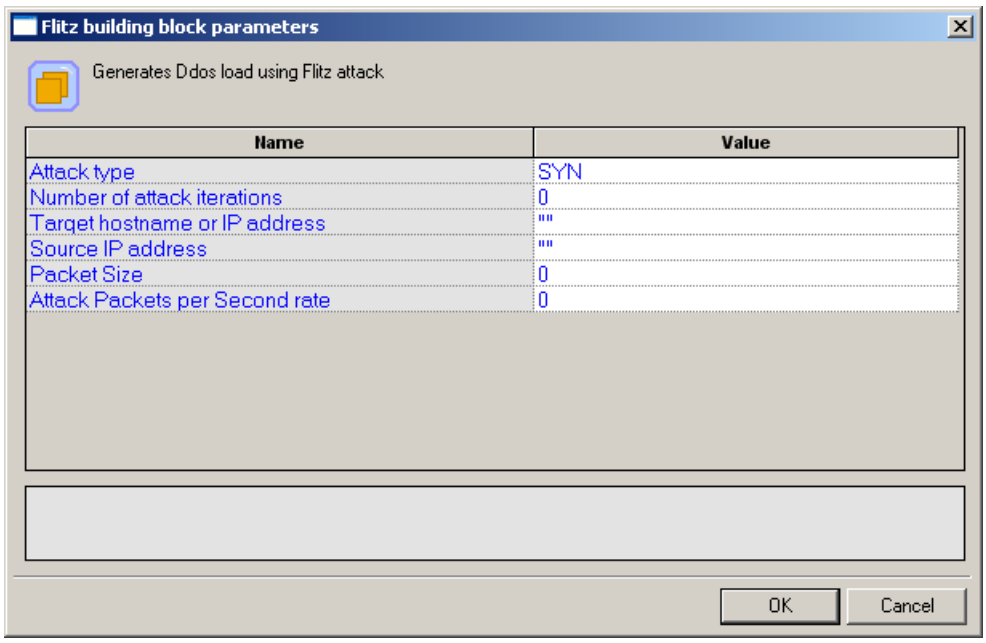
Flitz is a DDoS tool which features spoofed IP, TCP, and/or UDP flooding, including parallel flooding capabilities, distributed Smurf attacks, and status reports of the participating slaves. With a single `Stop` command, you are able to stop all slaves simultaneously.

Use the Flitz building block to generate a Flitz DDoS load test.

To enter a value:

1. Drag the Flitz icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Flitz building block parameters dialog box opens.



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 Attack Packets per Second Rate field is used to set the rate of packet transmission.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The Flitz building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the Flitz DDoS attack method with the parameter values specified by the user. The PPS value of zero (0) reflects a field deliberately set to zero to take advantage of the default maximization options.

The fields in the Flitz building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of Flitz DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ SYN ◆ UDP ◆ ICMP
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Source IP Address	Specify the IP address that will be set as source address of all sent packets. Type the appropriate name or IP 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. Leave empty to randomize.
Packet Size	Specify the size of the packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.

Field Name	Description
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

Juno

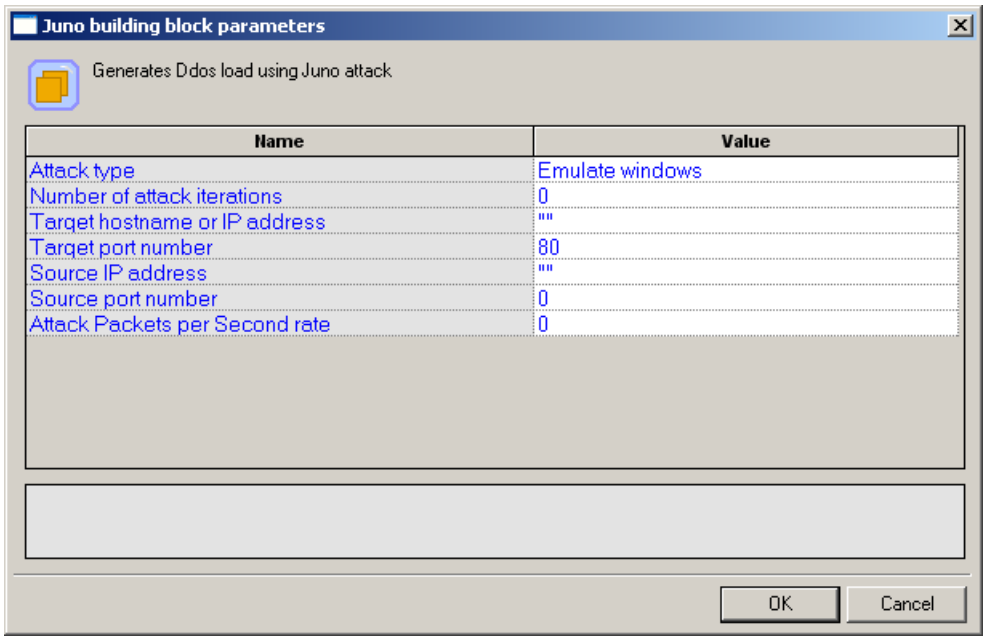
Juno is a SYN flooding DDoS attack that is available in both Windows and Linux flavors.

Use the Juno building block to generate a Juno DDoS load test.

To enter a value:

1. Drag the Juno icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Juno building block parameters dialog box opens.



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 Attack Type field is used to specify the type of Juno DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.
4. Click OK.

   The Juno building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the Juno DDoS attack method with the parameter values specified by the user. The PPS value of zero (0) reflects a field deliberately set to zero to take advantage of the default maximization options.

The fields in the Juno building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of Juno DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include Windows or Linux emulation.
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Number	Specify the number of the Target port that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero to randomize.
Source IP Address	Specify the IP address that will be set as source address of all sent packets. Type the appropriate name or IP 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. Leave empty to randomize.

Field Name	Description
Source Port Number	Specify the number that will be set as source port of all sent packets. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero to randomize.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

Knight

Knight is a distributed denial of service client that is both very light weight and very powerful. Knight goes on IRC and joins a channel, then accepts commands via IRC to avoid being detected and caught.

Knight includes such features as:

- An automatic updater via HTTP or FTP
- A checksum generator
- A SYN flooder
- A TCP flooder
- A UDP flooder
- Slice2
- Spoofing to subnets, and more.

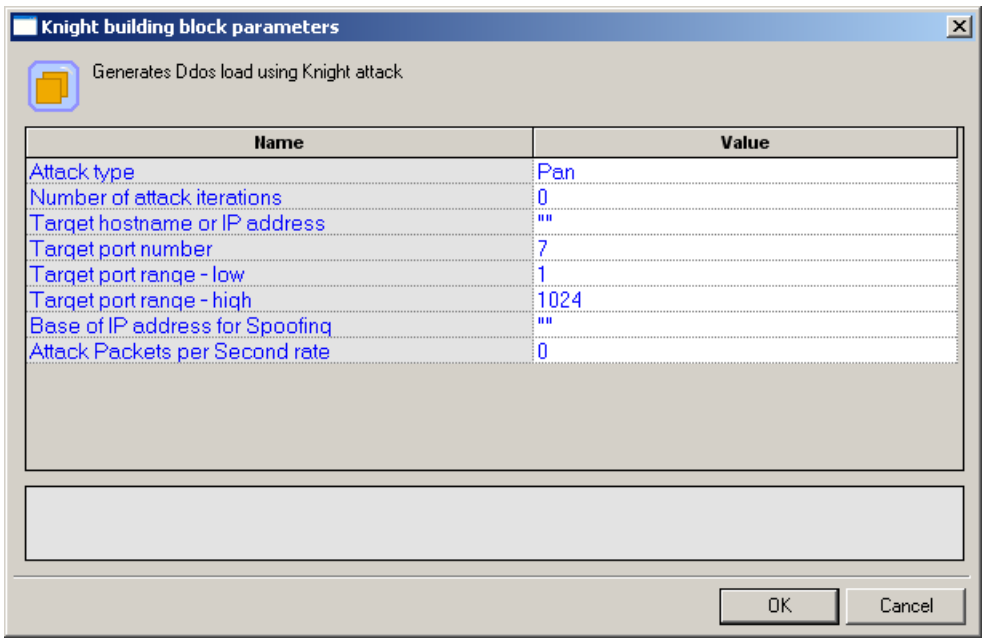
Knight has been used to create DDoS nets of over 1000 clients.

Use the Knight building block to generate a Knight DDoS load test.

To enter a value:

1. Drag the Knight icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Knight building block parameters dialog box opens.



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 Attack Type field is used to specify the type of Knight DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The Knight building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the Knight DDoS attack method with the parameter values specified by the user.

The fields in the Knight building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of Knight DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ PAN ◆ Majin ◆ UDP ◆ Slice2
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Number	Specify the number of the Target port that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero to randomize. Relevant only for UDP-type attacks.
Target Port Range (low)	Specify the lower bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Relevant only for Slice2-type attacks.
Target Port Range (high)	Specify the upper bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Relevant only for Slice2-type attacks.

Field Name	Description
Base of IP Address for Spoofing	Specify a base value for an IP address that will be used to generate spoofed source addresses. For example, specifying “11.22.” will cause the DDoS attack to generate IP source addresses with that as the prefix value, i.e. of the format “11.22.*.*”. Type the appropriate IP 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.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

LandAttack

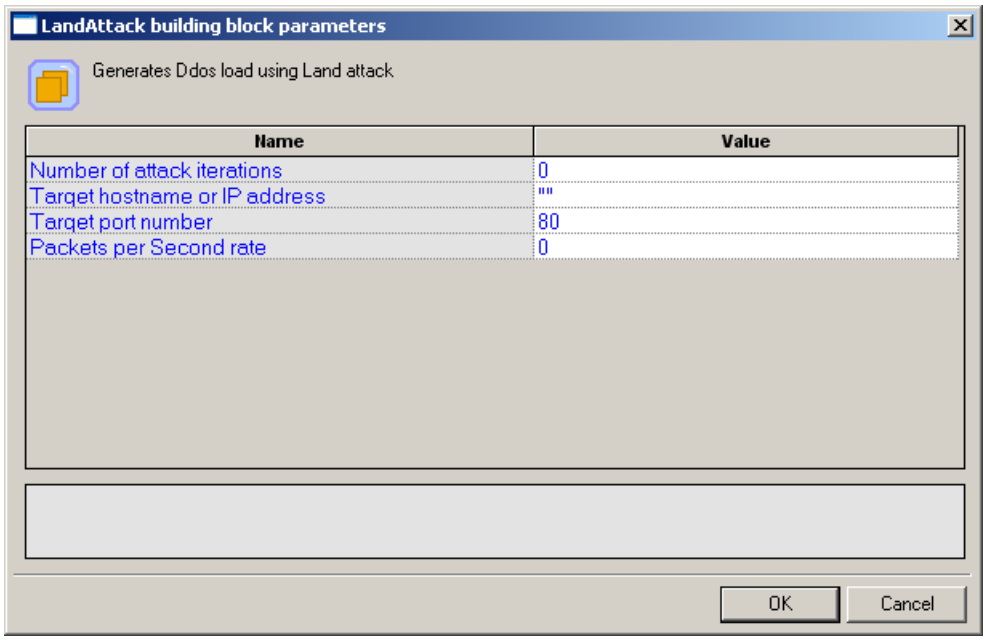
During a LandAttack, the attacker sends a forged TCP SYN packet with the same source and destination IP address. This confuses systems with outdated versions of the TCP/IP stack, because the system receives a TCP connection request that appears to come from itself. This may cause the target system to crash.

Use the LandAttack building block to generate a LandAttack DDoS load test.

To enter a value:

1. Drag the LandAttack icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The LandAttack building block parameters dialog box opens.



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 Target Hostname or IP Address field is used to specify the name or address of the intended attack target.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The LandAttack building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the LandAttack DDoS attack method with the parameter values specified by the user. The "EmptyParameter" values reflect the fields deliberately left empty to take advantage of the default randomization or maximization options.

The fields in the LandAttack building block parameters dialog box are described in the following table:

Field Name	Description
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Number	Specify the number of the Target port that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero to randomize.
Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

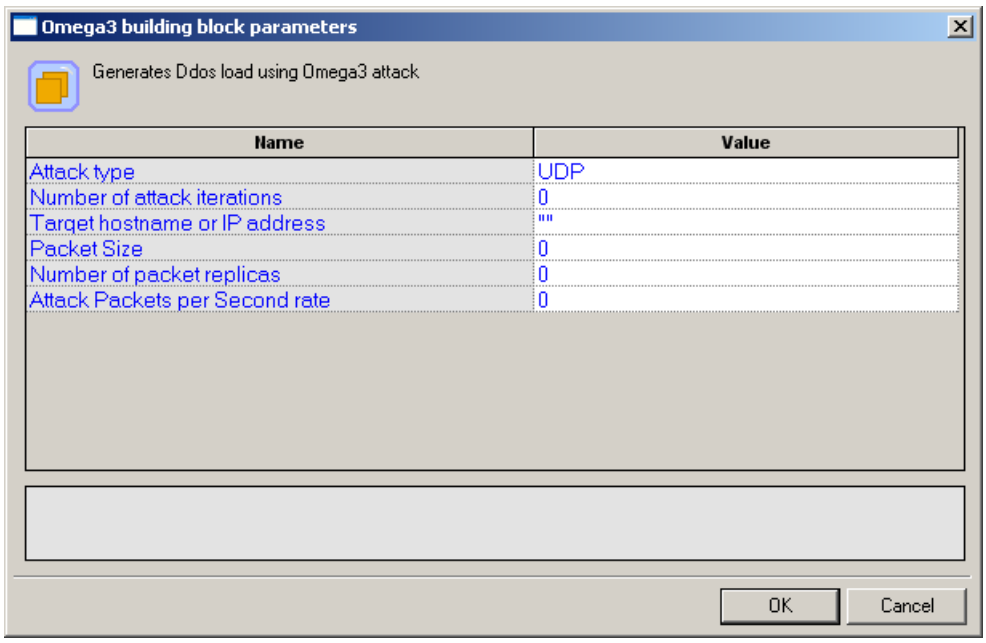
Omega3

Use the Omega3 building block to generate an Omega3 DDoS load test.

To enter a value:

1. Drag the Omega3 icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Omega3 building block parameters dialog box opens.



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 Attack Type field is used to specify the type of Omega3 DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The Omega3 building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the Omega3 DDoS attack method with the parameter values specified by the user, where the first parameter is the type of Omega3 attack to simulate (All), the second parameter is the number of attack iterations (set to 36), the third parameter is a string with the target host name, the fourth parameter is the packet size (set to 250), the fifth parameter is the number of identical packets in each packet-replica group (set to 3), and the PPS value of zero (0) reflects a field deliberately set to zero to take advantage of the default maximization options.

The fields in the Omega3 building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of Omega3 DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ IGMP ◆ UDP ◆ ICMP ◆ Stream ◆ All
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Packet Size	Specify the size of the packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Not relevant for SYN-type attacks.
Number of Packet Replicas	Specify the number of identical packets to include within a group of identical packets. Keep this number low (recommend <5) for better packet dispersal. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

Plague

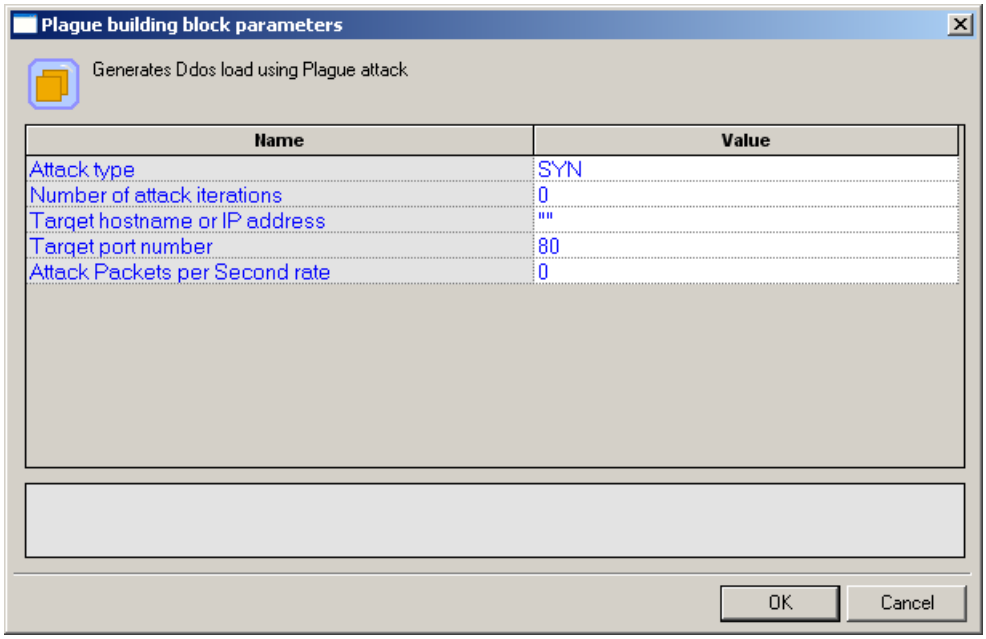
A Plague attack coordinates a number of compromised hosts in a distributed attack that produces denial of service difficulties for the current victim, combined with a sophisticated scan of the Internet for potential targets for future compromise. The Plague program runs on the user's own machine, communicating with a master server which is responsible for coordinating a set of ghost daemons. Plague attacks include such features as a stream (ACK) flooder and a SYN flooder.

Use the Plague building block to generate a Plague DDoS load test.

To enter a value:

1. Drag the Plague icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Plague building block parameters dialog box opens.



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 Attack Type field is used to specify the type of Plague DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The Plague building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

The Agenda calls the Plague DDoS attack method with the parameter values specified by the user. In this case, a stream-type Plague attack was specified.

The fields in the Plague building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of Plague DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ SYN ◆ Stream
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Number	Specify the number of the Target port that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero to randomize.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

Stacheldraht

Stacheldraht is a DDoS attack that started to appear in the late summer of 1999 and combines features of Trinoo (on page 279) and TFN (on page 273). Stacheldraht also contains some advanced features, such as encrypted attacker-master communication and automated agent updates. The types of attacks possible are similar to those of TFN; ICMP flood, SYN flood, UDP flood, and SMURF attacks.

Stacheldraht attack types include:

- UDP: This attack can be used to exploit the fact that for every UDP packet sent to a closed port, there will be an ICMP unreachable message sent back, multiplying the attack potential. Source IP address is spoofed. UDP source and destination ports are in the 1-9999 range.
- SYN: This attack sends continuous bogus connection requests. Possible effects include denial of service on one or more targeted ports, filled up TCP connection tables, and additional attack through potential multiplication by TCP/RST responses to non-existent hosts.
- ICMP: This attack sends ping requests from bogus (spoofed) source IPs, to which the victim replies with equally large response packets.
- Stream:
- IP: This attack sends empty IP messages, setting header length to 10 (40 bytes).
- Random: Sends TCP messages that include as many random parameters as possible.
- Smurf: This attack sends out ping requests that include the source address of the victim to broadcast amplifiers, hosts that reply with a drastically multiplied bandwidth back to the source.
- ACK: This attack sends out TCP ACK messages.
- NUL: This attack sends out TCP NUL messages.

Two variations of the Stacheldraht attack type included in the WebLOAD DDoS tool set are Stacheldraht4 (on page 270) and Stacheldraht26 (on page 267).

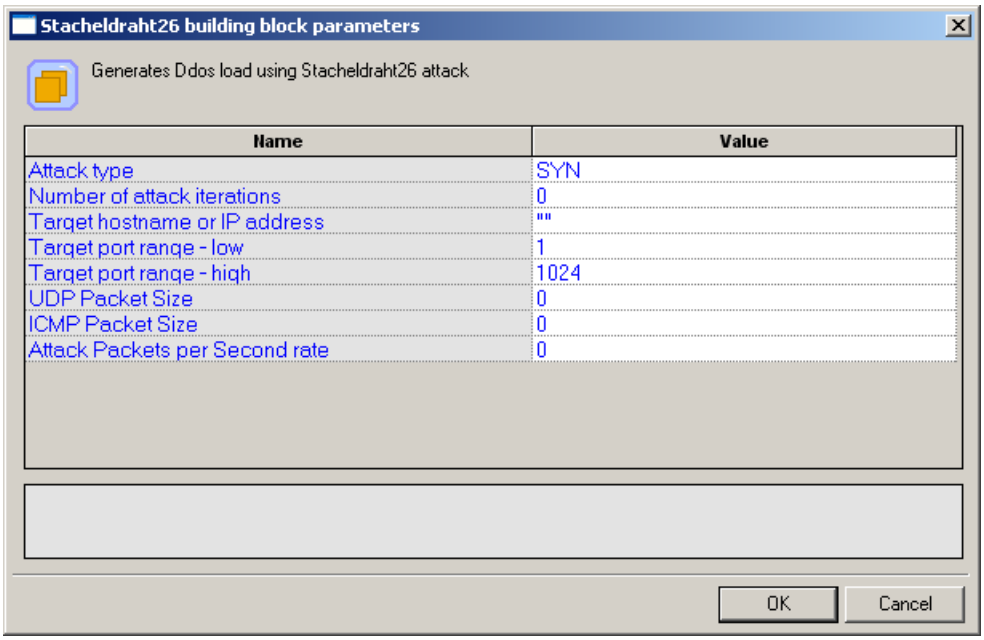
Stacheldraht26

Use the Stacheldraht26 building block to generate a Stacheldraht26 DDoS load test.

To enter a value:

1. Drag the Stacheldraht26 icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Stacheldraht26 building block parameters dialog box opens.



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 Attack Type field is used to specify the type of Stacheldraht26 DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The Stacheldraht26 building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the Stacheldraht26 DDoS attack method with the parameter values specified by the user, where the first parameter is the type of Stacheldraht26 attack to simulate (ICMP), the second parameter is the number of attack iterations (set to 65), the third parameter is a string with the target host name, the last parameter is the ICMP packet size (set to 365), and the PPS value of zero (0) reflects a field deliberately set to zero to take advantage of the default maximization options.

The fields in the Stacheldraht26 building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of Stacheldraht26 DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ SYN ◆ UDP ◆ ICMP ◆ Random ◆ Stream ◆ Havoc ◆ IP ◆ ACK ◆ NUL
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Range (low)	Specify the lower bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Port Range (high)	Specify the upper bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.

Field Name	Description
UDP Packet Size	Specify the size of the UDP packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Relevant for UDP-type attacks only.
ICMP Packet Size	Specify the size of the ICMP packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Relevant for ICMP-type attacks only.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

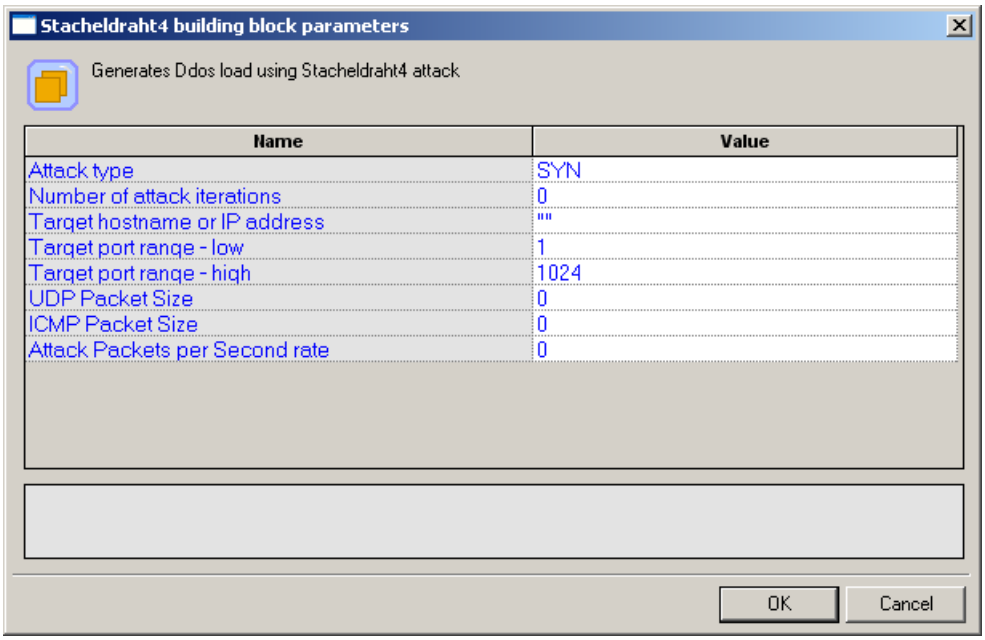
Stacheldraht4

Use the Stacheldraht4 building block to generate a Stacheldraht4 DDoS load test.

To enter a value:

1. Drag the Stacheldraht4 icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Stacheldraht4 building block parameters dialog box opens.



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 Attack Type field is used to specify the type of Stacheldraht4 DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The Stacheldraht4 building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the Stacheldraht4 DDoS attack method with the parameter values specified by the user, where the first parameter is the type of Stacheldraht4 attack to simulate (SYN), the second parameter is the number of attack iterations (set to 25), the third parameter is a string with the target host name, and the PPS value of zero (0) reflects a field deliberately set to zero to take advantage of the default maximization options.

The fields in the Stacheldraht4 building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of Stacheldraht4 DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ SYN ◆ UDP ◆ ICMP
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Range (low)	Specify the lower bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Port Range (high)	Specify the upper bound for a range of potential Target port numbers that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
UDP Packet Size	Specify the size of the UDP packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Relevant for UDP-type attacks only.

Field Name	Description
ICMP Packet Size	Specify the size of the ICMP packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Relevant for ICMP-type attacks only.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

TFN

TFN (Tribal Flood Network) attacks started to appear after the Trinoo (on page 279) attack. TFN client and daemon programs implement a DDoS network capable of employing a number of attacks, such as ICMP flood, SYN flood, UDP flood, and SMURF-style attacks. TFN is noticeably different from Trinoo in that all communication between the client (attacker), handlers, and agents is by way of ICMP ECHO and ECHO REPLY packets. Communication from the TFN client to daemons is accomplished via ICMP ECHO REPLY packets. The absence of TCP and UDP traffic sometimes makes these packets difficult to detect because many protocol monitoring tools are not configured to capture and display ICMP traffic.

TFN attack types include:

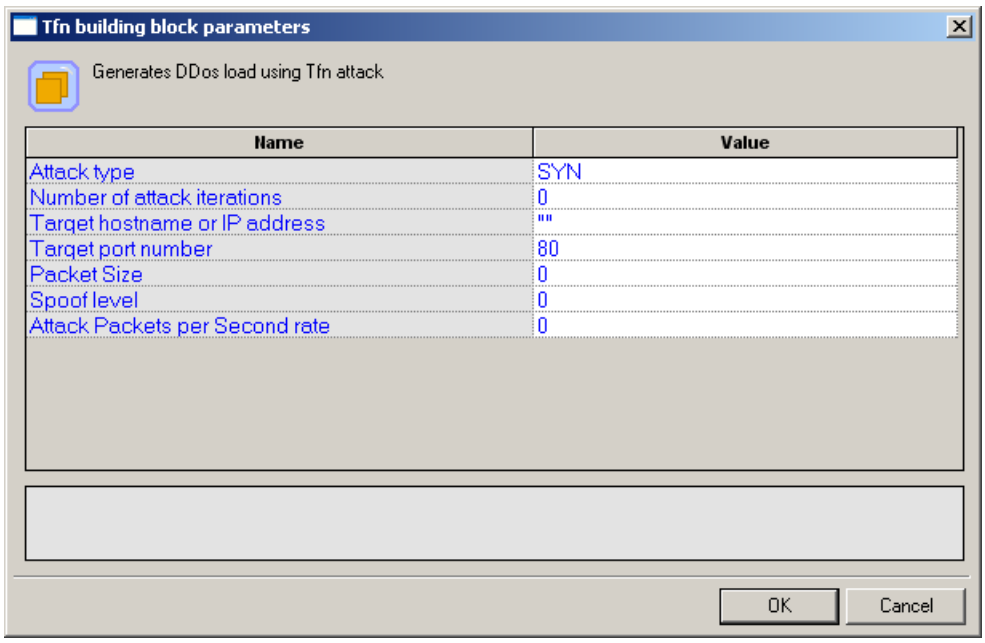
- TCP SYN: This attack sends continuous bogus connection requests. Possible effects include denial of service on one or more targeted ports, filled up TCP connection tables, and additional attack through potential multiplication by TCP/RST responses to non-existent hosts.
- ICMP Echo: This attack sends ping requests from bogus (spoofed) source IPs, to which the victim replies with equally large response packets.
- UDP: This attack can be used to exploit the fact that for every UDP packet sent to a closed port, there will be an ICMP unreachable message sent back, multiplying the attack potential. Source IP address is spoofed. UDP source and destination ports are in the 1-9999 range.
- Smurf: This attack sends out ping requests that include the source address of the victim to broadcast amplifiers, hosts that reply with a drastically multiplied bandwidth back to the source.

Use the TFN building block to generate a TFN DDoS load test.

To enter a value:

1. Drag the TFN icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The TFN building block parameters dialog box opens.



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 Attack Type field is used to specify the type of TFN DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The TFN building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the TFN DDoS attack method with the parameter values specified by the user.

The fields in the TFN building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of TFN DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ SYN ◆ UDP ◆ ICMP
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Number	Specify the number of the Target port that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero to randomize. Relevant for SYN-type attacks only.
Packet Size	Specify the size of the packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Not relevant for SYN-type attacks.
Spoof Level	Specify the level of spoofing. Values range from 0 to 3 where 0 is the maximum spoof level. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

TFN2K

TFN2K (Tribal Flood Network 2K) is a complex variant of the original TFN (on page 273) attack, with features designed specifically to make TFN2K traffic difficult to recognize and filter. TFN2K attacks are able to remotely execute commands, hide the true source of the attack using IP address spoofing, and transport TFN2K traffic over multiple transport protocols including UDP, TCP, and ICMP. TFN2K attacks include both flooding (as in TFN) and those designed to crash or introduce instabilities in systems by sending malformed or invalid packets, such as those found in the Teardrop and LandAttack (on page 260) attacks.

TFN2K attack types include:

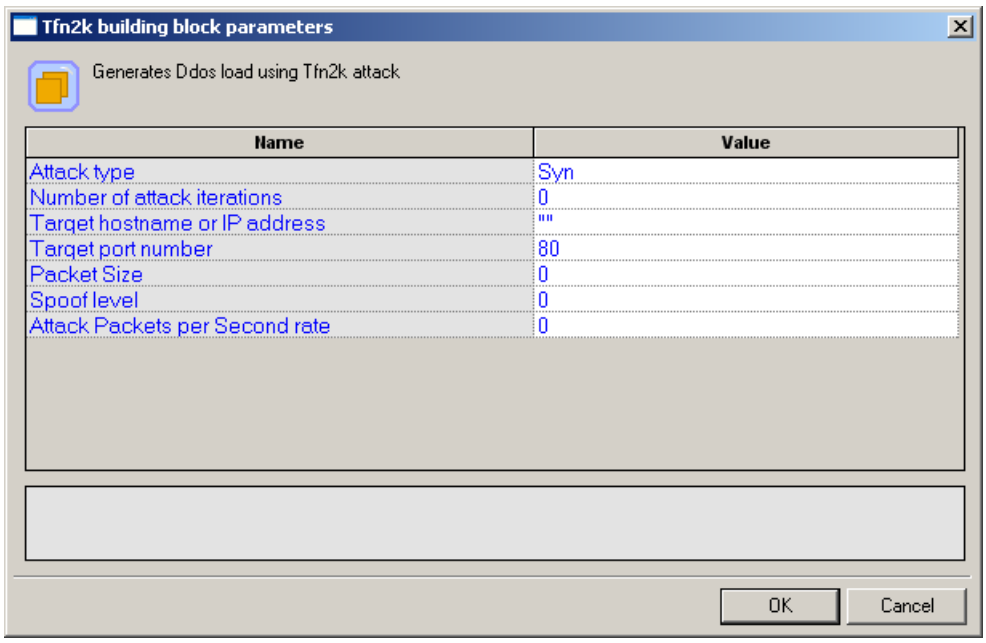
- TCP SYN: This attack sends continuous bogus connection requests. Possible effects include denial of service on one or more targeted ports, filled up TCP connection tables, and additional attack through potential multiplication by TCP/RST responses to non-existent hosts.
- ICMP Echo: This attack sends ping requests from bogus (spoofed) source IPs, to which the victim replies with equally large response packets.
- UDP: This attack can be used to exploit the fact that for every UDP packet sent to a closed port, there will be an ICMP unreachable message sent back, multiplying the attack potential. Source IP address is spoofed. UDP source and destination ports are in the 1-9999 range.
- Smurf: This attack sends out ping requests that include the source address of the victim to broadcast amplifiers, hosts that reply with a drastically multiplied bandwidth back to the source.
- Mix: This attack sends UDP, SYN, and ICMP packets interchanged on a 1:1:1 ratio. These can be particularly hazardous to routers and other packet forwarding devices or NIDS and sniffers.
- TARGA3: This attack uses random packets with IP-based protocols and values that are known to be critical or bogus, and can cause some IP stack implementations to crash, fail, or show other undefined behavior.

Use the TFN2K building block to generate a TFN2K DDoS load test.

To enter a value:

1. Drag the TFN2K icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The TFN2K building block parameters dialog box opens.



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 Attack Type field is used to specify the type of TFN2K DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The TFN2K building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the TFN2K DDoS attack method with the parameter values specified by the user. The “EmptyParameter” value reflects a field deliberately left empty to take advantage of the default randomization or maximization options.

The fields in the TFN2K building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of TFN2K DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ SYN ◆ UDP ◆ ICMP ◆ Targa ◆ Mix
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Number	Specify the number of the Target port that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero to randomize. Relevant for SY-type attacks only.
Packet Size	Specify the size of the packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Not relevant for SYN-type attacks.
Spoof Level	Specify the level of spoofing. Values range from 0 to 3 where 0 is the maximum spoof level. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.

Field Name	Description
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

Trinoo

Trinoo is a distributed SYN DDoS attack, using the UDP flood method, with communication between clients, handlers and agents via unencrypted UDP.

Trinoo attack types include:

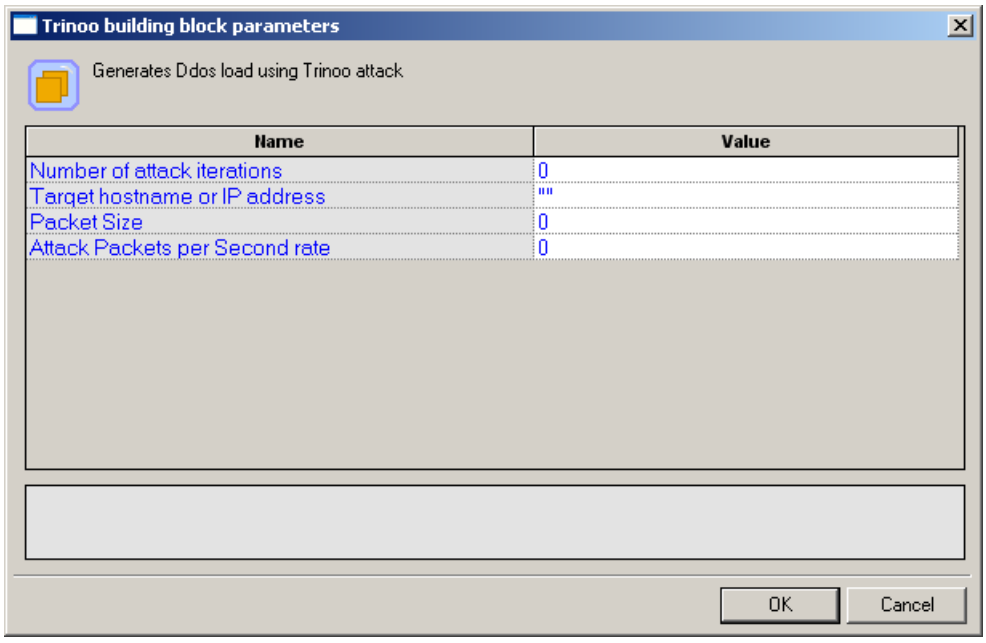
- UDP SYN: This attack sends UDP packets to random (0-65534) UDP ports on the specified IP addresses for a period of time.

Use the Trinoo building block to generate a Trinoo DDoS load test.

To enter a value:

1. Drag the Trinoo icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The Trinoo building block parameters dialog box opens.



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 Target Hostname or IP Address field is used to specify the IP address of the intended victim of the Trinoo DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The Trinoo building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

   The Agenda calls the Trinoo DDoS attack method with the parameter values specified by the user, where the first parameter is the number of attack iterations (set to 75), the second parameter is a string with the target host name, the third parameter is the packet size (set to 250), and the PPS value of zero (0) reflects a field deliberately set to zero to take advantage of the default maximization options.

The fields in the Trinoo building block parameters dialog box are described in the following table:

Field Name	Description
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Packet Size	Specify the size of the packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Attack Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

wlGenericAttack

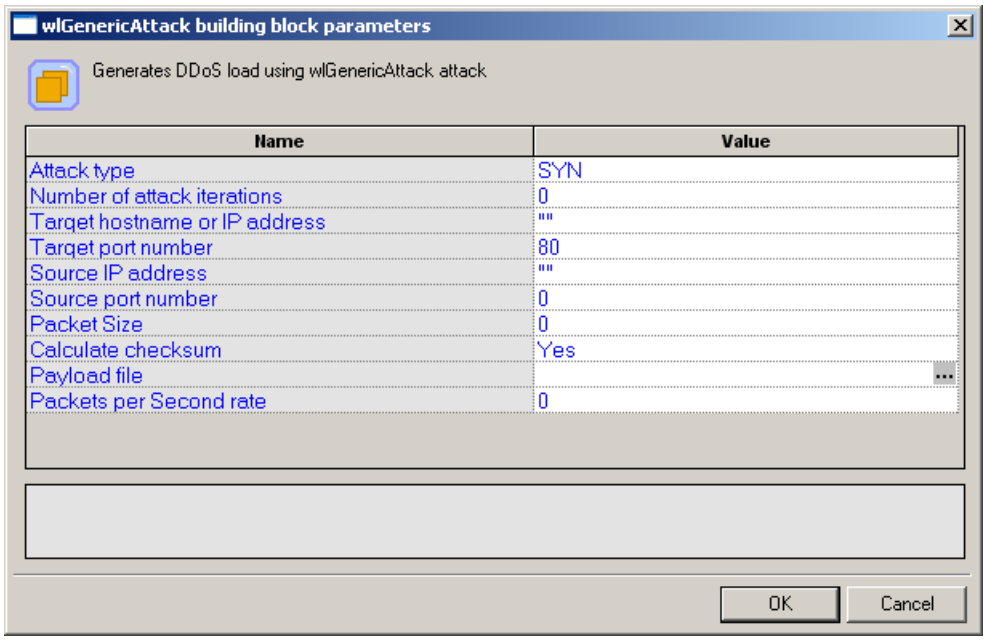
The wlGenericAttack tool enables simulating generic SYN, UDP, and ICMP (Ping) attacks. wlGenericAttack works by setting various packet properties, thereby creating new DDoS attacks based on these packets. The wlGenericAttack tool is also used to imitate new attacks as soon as they are detected, until RadView adds them to the supported attack tool set.

Use the wlGenericAttack building block to generate a DDoS load test using a generic DDoS prototype model.

To enter a value:

1. Drag the wlGenericAttack icon from the DDOS toolbox into the Agenda Tree at the desired location.

   The wlGenericAttack building block parameters dialog box opens.



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 Attack Type field is used to specify the type of generic DDoS attack to be simulated.

3. Enter the appropriate field value into the Value column next to the field name.

4. Click OK.

   The wlGenericAttack building block appears in the Agenda Tree. The JavaScript code, including the `InitAgenda()` function, is added to the Agenda. To see the new JavaScript code, view the Agenda in JavaScript Editing mode.

In the Agenda, the `InitAgenda()` function includes commands to work with the built-in DDoS include file and assigns the payload file specified by the user. The Agenda calls the generic DDoS attack method with the parameter values specified by the user.

The fields in the wlGenericAttack building block parameters dialog box are described in the following table:

Field Name	Description
Attack Type	Specify the type of generic DDoS attack to be simulated. Select the appropriate value from the drop-down list that appears when you click the Value input area for this field. The options include the following attack types: ◆ SYN ◆ UDP ◆ ICMP
Number of Attack Iterations	Specify the number of packets to send per round. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected.
Target Hostname or IP Address	Specify the hostname or IP address of the attack target station (victim). Type the appropriate name or IP 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.
Target Port Number	Specify the number of the Target port that will be attacked. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero to randomize.
Source IP Address	Specify the IP address that will be set as source address of all sent packets. Type the appropriate name or IP 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. Leave empty to randomize.

Field Name	Description
Source Port Number	Specify the number that will be set as source port of all sent packets. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero to randomize.
Packet Size	Specify the size of the packets to send to the target. Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Not relevant for SYN-type attacks.
Calculate Checksum	Specify whether or not packets should be generated with a valid checksum value. Select either Yes or No from the drop-down list that appears when you click the Value input area for this field.
Payload File	Specify the full path for a file to be used as payload in the packets being sent. Select the appropriate file from the Browser window that appears when you click the browse button to the right of the Value input area for this field.
Packets per Second Rate	Specify the rate of packet transmission in packets per second (PPS). Type the number into the value field or move to the right number using the up/down arrows that appear to the right of the value field when selected. Set to zero for maximum PPS value.

A P P E N D I X C

WebLOAD IDE File Types

The following is a list of files associated with a WebLOAD IDE project.

WebLOAD IDE Extension	WebLOAD IDE File Type
.WLP Files	WebLOAD IDE Project Files
.WLS Files	WebLOAD IDE Session Files
.WLA Files	Actual Repository Files
.WLE Files	Expected Repository Files
.LOG Files	Saved Log Window Files

A P P E N D I X D

Launching WebLOAD IDE Testing through the Command Line Interface

This section provides instructions and examples for using Command Line Interface (CLI) to launch WebLOAD IDE testing.

In This Appendix

Launching WebLOAD IDE Testing through the CLI

You can also initiate WebLOAD IDE testing directly through the CLI. You can enter the WebLOAD IDE launch command into a batch file or into an external script and WebLOAD IDE will run directly, without user intervention, using the parameters specified. This will enable you to perform unattended WebLOAD IDE testing at prescribed times.

To launch WebLOAD IDE testing through the CLI:

- Enter the following commands:

```
webloadIDE.exe /a v:\test\1.wlp v:\test\1.wls 5
```

including the full path where the webloadIDE.exe file is located together with a series of optional parameters (described below) into your external script to automatically launch a WebLOAD IDE test. When your script runs, the executable file will invoke WebLOAD IDE and run the specified test according to the specified parameters.

When running a test invoked by the executable, you can specify the following parameters:

Parameter	Description
Flags	/a - auto run Automatically run the WebLOAD IDE test without waiting for user input. If this flag is not specified, WebLOAD IDE 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 IDE and does not run a test. If the autorun flag flag is not set, the < Session name to save to>, and the < Number of rounds to run > parameters are ignored.

Syntax

Use the following syntax to define the parameters for running a WebLOAD IDE 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 289).

Examples

Example 1:

```
C:\Program Files\RadView\WebLOAD IDE2.0\bin\webloadide.exe
test1.wlp
```

This command opens WebLOAD IDE with the test1 project file and waits for user input.

Example 2:

```
C:\Program Files\RadView\WebLOAD IDE2.0\bin\webloadide.exe /a
test1.wlp test1.fts 3
```

This command:

- Opens WebLOAD IDE and automatically runs a test using the test1.wlp project file.
- Runs the project for three iterations.
- Saves the test results in the WebLOAD IDE session file test1.wls, which includes all of the test data and results.

Example 3:

```
C:\Program Files\RadView\WebLOAD IDE2.0\bin\webloadide.exe /a
test1.wlp test1.wls 3 /a test2.wlp test2.wls 2
```

This command:

- Opens WebLOAD IDE 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 IDE session file test1.wls, which includes all of the test data and results.
- Opens the WebLOAD IDE project file test2.wlp
- Runs the project test2.wlp for two iterations
- Saves the test results in the WebLOAD IDE session file test2.wls, which includes all of the test data and results.

Examples

I N D E X

Index

P

R

S

T

U

V

W