Cincom®

Cincom Smalltalk™

Web Server Configuration Guide

P46-0141-06

SIMPLIFICATION THROUGH INNOVATION®

Part Number: P46-0141-06

Software Release 7.6

Cincom Systems, Inc.

55 Merchant Street

Cincinnati, Ohio 45246

Phone: (513) 612-2300

Fax: (513) 612-2000

World Wide Web: http://www.cincom.com

Contents

Chapter 2 Servers

Chapter 3 Application Delivery

Chapter 4 Server Architecture

Chapter 5 Using Front-End HTTP Servers

About This Book

This book gives detailed configuration information for the VisualWorks Application Server™. It includes an overview of the Application Server environment, details about the user interface, installing and configuring server applications, internal architecture of the server, and the interface between VisualWorks and commercial HTTP servers.

Audience

The discussions in this book presuppose that you are familiar with the World Wide Web, the use of web browsers, and the administration of HTTP servers.

Some chapters in this book also presuppose a general knowledge of object-oriented concepts, Smalltalk, and the VisualWorks environment.

For an overview of Smalltalk, and the VisualWorks application architecture, see the *VisualWorks Application Developer's Guide*.

In addition to this book, the documentation set for the VisualWorks Application Server includes the following:

- *Web Application Developer's Guide:* Provides detailed information about building Web applications using the VisualWorks Web Toolkit.
- *Web GUI Developer's Guide:* Provides detailed information about building Web applications using the VisualWorks UI Painter and VisualWave.

Conventions

We have followed a variety of conventions, which are standard in the VisualWorks documentation.

Typographic Conventions

The following fonts are used to indicate special terms:

Example	Description
template	Indicates new terms where they are defined, emphasized words, book titles, and words as words.
`cover.doc`	Indicates filenames, pathnames, commands, and other constructs to be entered outside VisualWorks (for example, at a command line).
`filename.xwd`	Indicates a variable element for which you must substitute a value.
windowSpec	Indicates Smalltalk constructs; it also indicates any other information that you enter through the VisualWorks graphical user interface.
Edit menu	Indicates VisualWorks user-interface labels for menu names, dialog-box fields, and buttons; it also indicates emphasis in Smalltalk code samples.

Special Symbols

This book uses the following symbols to designate certain items or relationships:

Examples	Description
File ⟶ New	Indicates the name of an item (New) on a menu (File).
<Return> key <Select> button <Operate> menu	Indicates the name of a keyboard key or mouse button; it also indicates the pop-up menu that is displayed by pressing the mouse button of the same name.
<Control>-<g>	Indicates two keys that must be pressed simultaneously.
<Escape> <c>	Indicates two keys that must be pressed sequentially.
Integer>>asCharacter	Indicates an instance method defined in a class.
Float class>>pi	Indicates a class method defined in a class.

Mouse Buttons and Menus

VisualWorks supports a one-, two-, or three-button mouse common on various platforms. Smalltalk traditionally expects a three-button mouse, where the buttons are denoted by the logical names <Select>, <Operate>, and <Window>:

<Select> button	*Select* (or choose) a window location or a menu item, position the text cursor, or highlight text.
<Operate> button	Bring up a menu of *operations* that are appropriate for the current view or selection. The menu that is displayed is referred to as the *<Operate> menu*.
<Window> button	Bring up the menu of actions that can be performed on any VisualWorks *window* (except dialogs), such as **move** and **close**. The menu that is displayed is referred to as the *<Window> menu*.

These buttons correspond to the following mouse buttons or combinations:

	3-Button	2-Button	1-Button
<Select>	Left button	Left button	Button
<Operate>	Right button	Right button	<Option>+<Select>
<Window>	Middle button	<Ctrl> + <Select>	<Command>+<Select>

Getting Help

There are many sources of technical help available to users of VisualWorks. Cincom technical support options are available to users who have purchased a commercial license. Public support options are available to both commercial and non-commercial license holders.

Commercial Licensees

If, after reading the documentation, you find that you need additional help, you can contact Cincom Technical Support. Cincom provides all customers with help on product installation. For other problems there are several service plans available. For more information, send email to supportweb@cincom.com.

Before Contacting Technical Support

When you need to contact a technical support representative, please be prepared to provide the following information:

- The *version id,* which indicates the version of the product you are using. Choose **Help → About VisualWorks** in the VisualWorks main window. The version number can be found in the resulting dialog under **Version Id:**.
- Any modifications (*patch files*) distributed by Cincom that you have imported into the standard image. Choose **Help → About VisualWorks** in the VisualWorks main window. All installed patches can be found in the resulting dialog under **Patches:**.
- The complete error message and stack trace, if an error notifier is the symptom of the problem. To do so, select **copy stack** in the error notifier window (or in the stack view of the spawned Debugger). Then paste the text into a file that you can send to technical support.

Contacting Technical Support

Cincom Technical Support provides assistance by:

Electronic Mail

To get technical assistance on VisualWorks products, send email to:

supportweb@cincom.com.

Web

In addition to product and company information, technical support information is available on the Cincom website:

http://supportweb.cincom.com

Telephone

Within North America, you can call Cincom Technical Support at (800) 727-3525. Operating hours are Monday through Friday from 8:30 a.m. to 5:00 p.m., Eastern time.

Outside North America, you must contact the local authorized reseller of Cincom products to find out the telephone numbers and hours for technical support.

Non-Commercial Licensees

VisualWorks Non-Commercial is provided “as is,” without any technical support from Cincom. There are, however, on-line sources of help available on VisualWorks and its add-on components. Be assured, you are *not* alone. Many of these resources are valuable to commercial licensees as well.

The University of Illinois at Urbana-Champaign very kindly provides several resources on VisualWorks and Smalltalk:

- A mailing list for users of VisualWorks Non-Commercial, which serves a growing community of VisualWorks Non-Commercial users. To subscribe or unsubscribe, send a message to:

  vwnc-request@cs.uiuc.edu

  with the SUBJECT of "subscribe" or "unsubscribe".

- An excellent Smalltalk archive is maintained by faculty and students at UIUC, who are long-time Smalltalk users and leading lights in the Smalltalk community, at:

  http://st-www.cs.uiuc.edu/

- A Wiki (a user-editable web site) for discussing any and all things VisualWorks related at:

  http://wiki.cs.uiuc.edu/VisualWorks

- A variety of tutorials and other materials specifically on VisualWorks at:

  http://wiki.cs.uiuc.edu/VisualWorks/Tutorials+and+courses

The Usenet Smalltalk news group, comp.lang.smalltalk, carries on active discussions about Smalltalk and VisualWorks, and is a good source for advice.

Additional Sources of Information

This is but one manual in the VisualWorks library. The Cincom Smalltalk publications website:

http://www.cincom.com/smalltalk/documentation

is a resource for the most up to date versions of VisualWorks manuals and additional information pertaining to Cincom Smalltalk.

Online Help

VisualWorks includes an online help system.

To display the online documentation browser, open the **Help** pull-down menu from the VisualWorks main menu bar and select one of the help options.

News Groups

The Smalltalk community is actively present on the internet, and willing to offer helpful advice. A common meeting place is the comp.lang.smalltalk news group. Discussion of VisualWorks and solutions to programming issues are common.

VisualWorks Wiki

A wiki server for VisualWorks is running and can be accessed at:

http://brain.cs.uiuc.edu:8080/VisualWorks.1

This is becoming an active place for exchanges of information about VisualWorks. You can ask questions and, in most cases, get a reply in a couple of days.

Commercial Publications

Smalltalk in general, and VisualWorks in particular, is supported by a large library of documents published by major publishing houses. Check your favorite technical bookstore or online book seller.

1

Overview

The VisualWorks Application Server is a complete environment for presenting Web applications to clients on the Internet. Applications are served using one of three frameworks: Smalltalk Server Pages, servlets, or VisualWave. The Application Server may also serve files or other resources in the host's operating system.

The Application Server is a VisualWorks *image* running one or more servers that are listening for and responding to Web requests. This image contains both the Application Server itself and any applications that are invoked by the servers responding to Web requests.

This chapter describes:

- Organization of the Application Server
- Application Server Parcels
- Configuring Environment Settings

Organization of the Application Server

The VisualWorks Application Server may be configured as one of two types of images:

- Development image
- Server Deployment image

Development begins with the VisualWorks image, when you load the parcels for the Application Server. The development image contains all the tools needed for building and testing your application.

Initial testing of an application may be done with the developer image, but to enjoy the full server implementation, or to use any CGI relay or load-balancing facilities, you must load some of the optional server parcels. For details on working with CGIs, "Using Front-End HTTP Servers" on page 5-1. For details on setting up load-balanced servers, "Using Multiple HTTP Servers" on page 6-1.

Once your application development project is complete, you can prepare it for deployment by *packaging* the development image. By packaging the finished application, you are saving it either as a parcel or a complete server image. For details, "Application Delivery" on page 3-1.

A variety of options are available for configuring the server image. These are described in "Configuring Environment Settings" on page 1-3.

Application Server Parcels

The VisualWorks Application Server is contained in a set of parcels. To develop Web applications, you must load one or more of these parcels.

The parcels, which are contained in the **`/web`**, **`/wavedev`**, and **`/waveserver`** directories of the VisualWorks distribution, are:

VisualWave	Complete environment for developing VisualWave applications
WaveLoad-Server	Load-balancer component for the Application Server
WaveLoad-Client	Converts an Application Server image into a load-balanced secondary server
WebToolkit	Support for Smalltalk server pages and servlets

Configuring Environment Settings

The Application Server environment allows you to configure a variety of settings, such as:

- Alert and error condition handling
- The web browsers recognized for each look policy
- The maximum amount of memory to be used
- The actions to be taken when specified levels of memory are reached

The Application Server loads *settings* from four sources, in this order:

1. Values saved in the image.
2. Values in a special server *configuration file*, if it exists.
3. Command line parameters.
4. The Server Settings tool.

Values in later sources override those in earlier ones. For example, the command line overrides the configuration file, and the Server Settings tool overrides both the command line and the configuration file.

See "Server Environment Settings" on page B-1 for a complete list of options and their values.

Configuration File

At startup, the Application Server looks in the working directory for a configuration file. The default file is:

- UNIX: **`.vwaverc`**
- Windows 95/98 or NT: **`vwave.ini`**

> **Note:** Values in the configuration file override values saved in the image file.

File Name

The default name for a configuration file can be changed. For details, see "Configuration File Name" on page B-15.

You can also use more than one configuration file. To do so, in one configuration file, set the **`configFileName`** value to point to another configuration file. The **`configFileName`** option acts as an "include." Be careful not to create loops between configuration files.

Contents and Format

The server configuration file is a plain text file. It may contain the complete set of options or a subset. In the file, each setting appears on its own line. A setting can be either of these forms:

```
optionName = value
optionName value
```

Each value must be a number, string, literal array, a boolean, or nil.

Every setting must be under the appropriate section name. The section name is enclosed in brackets and appears on its own line:

```
[alerts]
```

Comments begin with a semicolon (`;`) and continue to the end of the line. Alternately, comments can be enclosed in double quotation marks.

Creation

You can create a server configuration file by:

- Saving all settings from the Server Settings tool, *or*
- Using a text editor to create a new configuration file or edit an existing one.

Command Line

Settings on the command line override any values that are saved in the image and settings in the default configuration file.

To set an option on the command line at startup, include the section name, option name, and option value at the end of the command line:

```
visual websrv.im [section] option=value &
```

You can include more than one section and option/value pair on the same command line.

Some options have command-line shortcuts. “Command-Line Shortcuts” on page B-17 for a complete list.

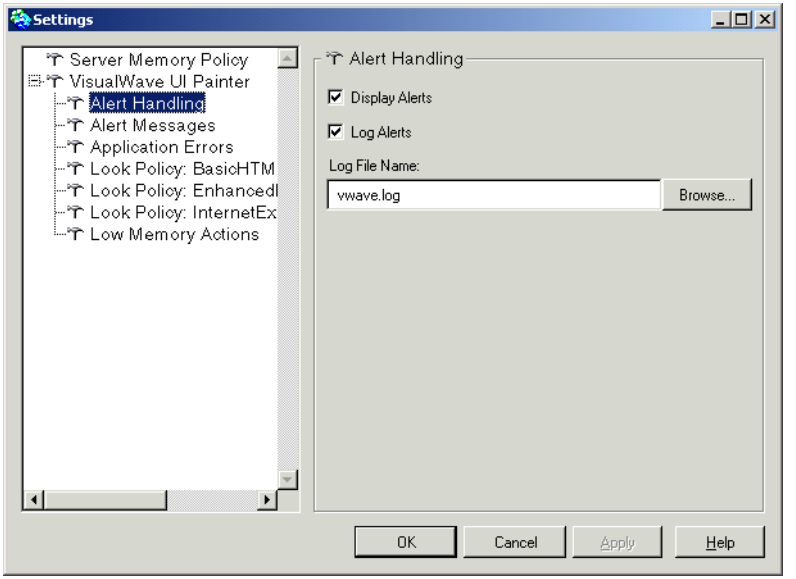
Settings Tool

The Server Settings override settings saved in the image, those set in the default configuration file, and specified on the command line. However, values set in the Server Settings tool are not preserved when you restart the server environment unless you save the image or save the settings in the default configuration file.

Opening the Settings Tool

To open the Server Settings tool, choose **File → Settings…** in the Server Console.

The Server Settings tool presents a list of “pages” on the left. These roughly correspond to sections in the server configuration file.



Server Settings Tool

Making Changes Take Effect

In the Settings Tool, click the **Apply** button.

Saving Changes

Changes you have applied in the Server Settings tool are in effect until you change them again or exit the environment.

Changes that you make using the Settings Tool are saved when you save the image, or can be saved in a configuration file.

To save the settings to a configuration file, use the <Option> menu in the left-hand side of the tool, and click **Save…**.

For additional details on the Settings Tool, “Server Settings Tool” on page A-34.

1

Servers

Most of the work in the VisualWorks Application Server is performed by individual application servers. Servers receive the web requests and control how the requests are resolved and answered.

Each server listens on its own port and acts independently of the others. They share only the server environment and its settings.

This chapter describes how you can create, start, and stop servers. It also explains different ways you can monitor a server's incoming traffic, and how you can exercise control over that traffic.

Servers in VisualWorks

Inside the VisualWorks environment, servers come in two flavors:

Smalltalk HTTP Server

This type of server responds to requests that come directly from web browsers, without the intercession of another web server. By default, Smalltalk HTTP Servers run on port 8008, but you can set them to run on any port number.

External Web Server

This type of server responds to requests that have been forwarded by commercial web servers (e.g., Apache, IIS) either through a CGI relay script or ISAPI relay. By default, External Web Servers run on port 2223, but you can set them to run on any port number.

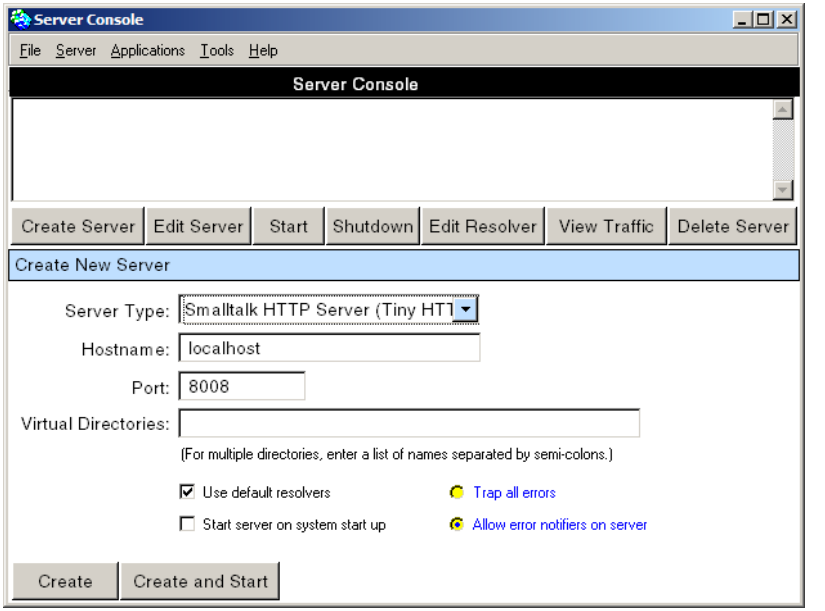
Creating a Server

Before the server environment can respond to any web requests, it must have at least one server created and started within it.

To create a server:

1 In the Server Console, click the **Create Server** button.

The Server Console expands to include the Create New Server panel:



Creating a New Server

2 Choose the **Server Type**:

- A **Smalltalk HTTP Server** receives requests directly from web browsers (without an intermediary HTTP server or CGI script) and are useful for testing applications. Internally, this is represented by an instance of class WaveHTTPRequestBroker.
- An **External Web Server** receives requests from commercial HTTP servers, usually from a CGI relay script. For more details about configuring an external web server and using CGI relays, see "Using Front-End HTTP Servers" on page 5-1. Internally, this is represented by an instance of class WaveIPRequestBroker.

During application development, the Smalltalk HTTP Server is generally easiest to use.

3 In the **Hostname** field, enter the name of the server environment's host machine.

4 In the **Port** field, enter a valid port number.

An Application Server can use any port. Note that port 80 is the default value assumed when the URL does not include a port number. Note, also, that on most systems, low numbered ports (including port 80) require special system privileges. You can avoid this by choosing a higher numbered port. See your system's networking documentation for specific requirements.

5 Decide whether or not to **Use default resolvers**. Note: this option is only applicable when using VisualWave applications.

If this box is checked, the server's PathInfoPrefixResolver is set up to recognize several default paths:

launch: Used to start a new instance of an application. Forwards the web request to an AnySessionLauncher in the Server development environment or a RegisteredSessionLauncher in the server environment.

submit: Used to connect with an already-running instance of an application. Forwards the request to a SessionResolver.

echo: Forwards the request to a RequestEchoer, which returns the web request.

JavaSTGM.class: Used by the Server when generating Java-rendered graphics from Smalltalk graphics.

6 Decide whether or not you want to **Start server on system startup**. If this box is checked, the server is started (active) whenever the server environment is started. If this box is not checked, the server is inactive at system startup, even if it had been active when the server environment was last saved.

7 Click the **Create** or **Create and Start** button.

The server is created and added to the servers list at the top of the Server Console. If you clicked **Create**, the server is inactive and not yet able to receive requests. If you click **Create and Start**, the server becomes active and ready to receive requests.

Starting a Server

Servers must be started (active) before they can receive requests. You can start servers when you create them by clicking the **Create and Start** button. You also can start or restart any inactive server:

1 In the Server Console, select a server.

2 Click the **Start** button.

Changing a Server's Characteristics

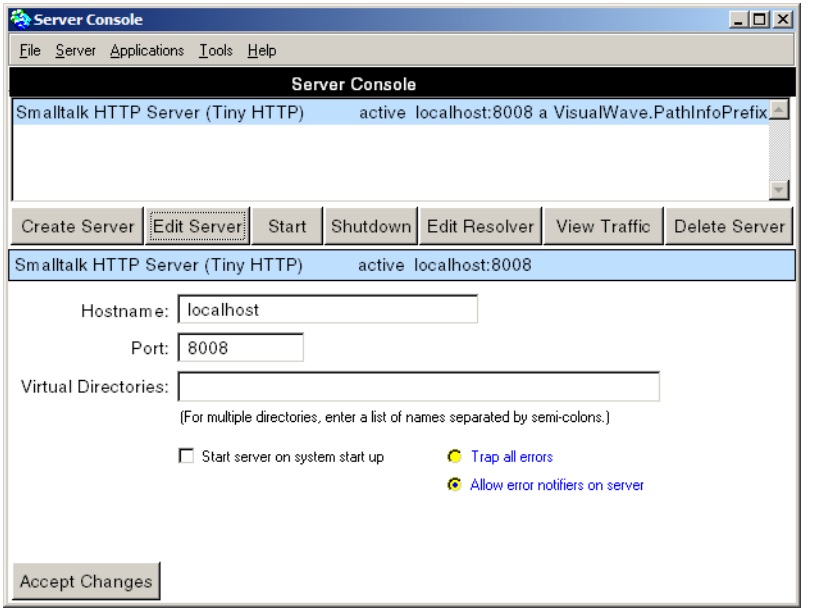
Once a server is created, any changes made to the server's **Hostname** will take effect immediately. Changes to **Port** can only be made to inactive servers, and they take effect when you click **Accept Changes**. Changes to **Start server at system startup** can be made at any time, though you must save the server environment for the change to affect the next startup.

To change characteristics for an existing server:

1 In the Server Console, select the server.

2 Click the **Edit Server** button.

The Server Console expands to include the Edit Server panel:



3 Edit the fields for the characteristics that you want to change.

4 Click the **Accept Changes** button.

Your changes are accepted into the server environment. To preserve them so that they are present the next time you start the server environment, save the server image.

Monitoring Traffic

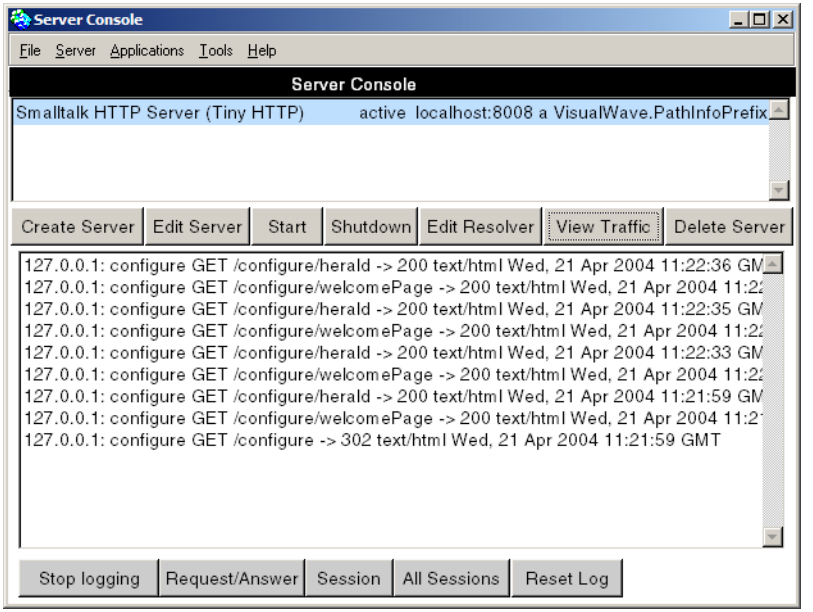
All Traffic by Time

You can monitor all the traffic on a particular server:

1 In the Server Console, select an active server.

2 Click the **View Traffic** button.

The Server Console expands to include the View Traffic panel:



Viewing Server Traffic

The basic View Traffic panel displays a list of the web hits processed by the server, in chronological order with the most recent web hit at the top. For each web hit, the entry includes:

- A summary of the request received, including the web browser's host, the method used for the form data (GET or POST), and the path from the URL.
- A summary of the answer returned, including the error code (or nil if successful), the mime type, and the time.

All Sessions

To view a list of a server's active web sessions:

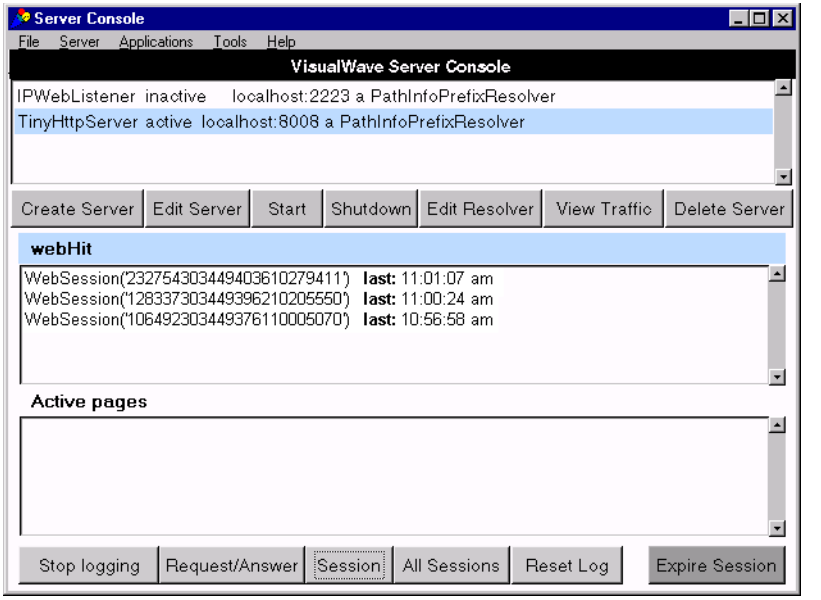
1 In the Server Console, select the server.

2 Click the **View Traffic** button. A list of web requests appears.

3 Select a web hit (request and answer).

4 Click the **All Sessions** button.

The Server Console expands to include the View Session panel with all active sessions listed:



Viewing Servers and Active Sessions

The View All Sessions panel displays details about all the sessions for the selected server. It includes two lists:

- The top is a list of sessions. Each entry includes the session key (in parentheses), the last time input was received from the user, and the referring HTTP server (if any).
- The bottom is a list of the pages that are still in use by the selected session. Each entry includes the kind of page (WebPageController), page key (in parentheses) and the page name (equivalent to the window title). The user cannot interact with pages that are not in this list, even if those pages are cached in his/her web browser. Attempts to interact with stale pages redirect the user to active pages.

Session for a Particular Web Request

To view a list of a server's active sessions:

1 In the Server Console, select the server.

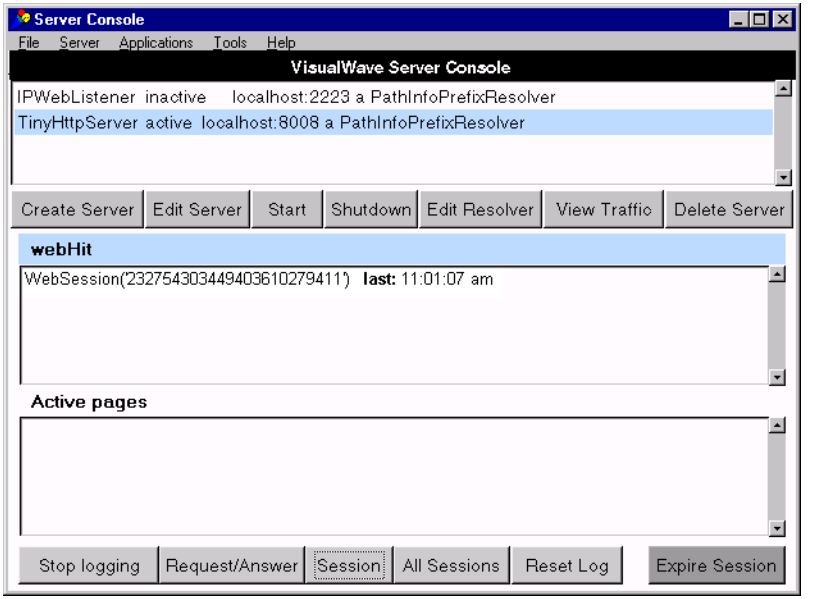
2 Click the **View Traffic** button.

3 Select a web hit from the list.

4 Click the **Session** button.

> **Note:** **Session** is applicable only for requests that are handled by SessionResolvers (submits).

The Server Console expands to include the View Session panel with only the session for the selected web hit:



Viewing Session Details

The View Session panel displays details about the session that handled the selected web hit. The contents are the same as that for the View All Sessions panel:

- The top is a list of sessions, including the session's key (in parentheses), the last time input was received from the user, and the referring HTTP server (if any).
- The bottom is a list of the pages that are still in use by the selected session. Each entry includes the page key (in parentheses) and the page name (equivalent to the window title). The user cannot interact with pages that are not in this list, even if those pages are cached in his/her web browser. Attempts to interact with stale pages redirect the user to active pages.

If the message **No active session for selected web request** is displayed, either the selected request was not handled by a SessionResolver or the session has expired.

Complete Request and Answer Details

To view the details of a web request and its answer:

1 In the Server Console, select the server.

2 Click the **View Traffic** button.

3 Select a web hit from the list.

4 Click the **Request/Answer** button.

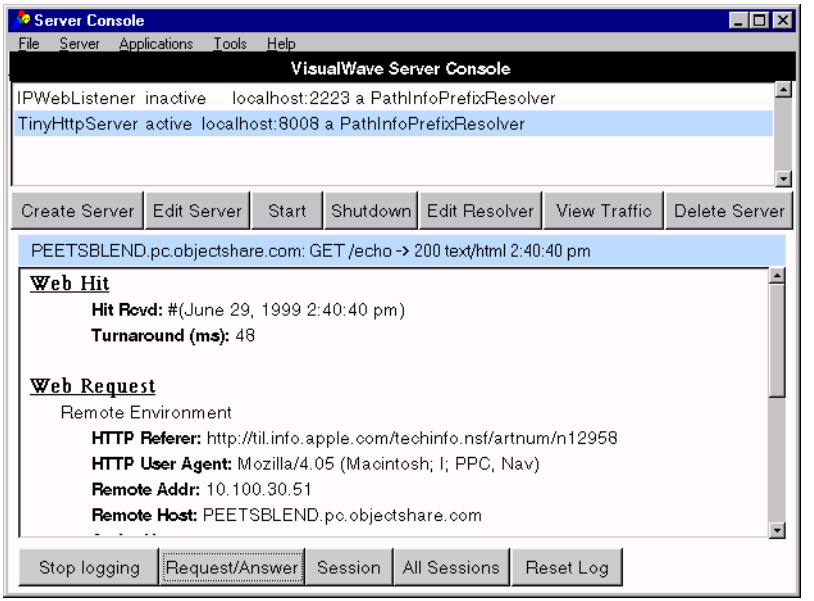
The Server Console expands to include the View Request/Answer panel with information about the selected web hit. The View Request/Answer panel displays details about the selected web hit, including the complete web request, the web answer, and the time it took to handle the web request.

Web Hit:

Includes the time that the web request was received and the time it took to process the web request and return an answer.

Web Request:

Includes the environment variables that make up the web request and their values. For web hits that are continuations of an existing user interaction (submits), the session key and page (inline) key are included under the heading "**Form Data**."



Viewing Web Hit Statistics

Web Answer:

Information about the response returned by the server to the user's browser:

Status:
The status code of the response. An empty string indicates that the request was successfully answered. An error code (HTTP error numbers) indicates that there was a problem answering the request.

Reason:
An empty string indicates that the request was successfully answered. If there was an error, a short explanation appears.

Media:
The kind of object that was returned. For launches, this is the name of the class that was opened.

Controlling Traffic

Control of server traffic is important for preserving resources available for servicing incoming requests. Traffic can be controlled within the server environment and, if you are using a multiple-server configuration, across a group of servers (for a discussion of traffic control in a multiple-server environment, see: "Using Multiple HTTP Servers" on page 6-1).

You can set up the server environment and your servers to do a great deal of traffic control automatically, based on memory use and session inactivity. You can also control traffic explicitly at the session level. The following discussion explains how you can control the criteria for maintaining active sessions with clients.

Automatic Control

Traffic can be controlled automatically by a combination of settings for the server environment and for individual SessionResolvers.

Server Environment Controls

The server environment allows you to set the levels at which memory is considered stressed and the actions to take at various memory stress levels. Those actions include:

- Purging cached resources
- Aggressively purging sessions
- Deferring launches
- Deferring submits

Server environment settings affect all servers and all sessions, launches, and submits for those servers. For details about these settings, see "Server Environment Settings" on page B-1.

SessionResolver Controls

For each SessionResolver, you can set the parameters for timing out inactive sessions. You can set:

- How often the SessionResolver checks for inactive sessions.

  When a launch request occurs, the SessionResolver checks to see when how much time has passed since it last checked for inactive sessions. If the amount of time is more than the SessionResolver's **Poll for inactive sessions** value, the SessionResolver checks for inactive sessions. Note that if there are no launch requests, the SessionResolver does not check for inactive sessions.

- How long a session must go without interaction from the user before it is considered "inactive" and eligible for expiration. You can set values for normal and stressed memory conditions. The memory amount that is considered "stressed" is determined by the server environment's Aggressive Session Expiration value (see "Server Actions" on page B-8.)

To change the SessionResolver's timeout parameters:

1 In the Server Console, select a server.

2 Click the **Edit Resolver** button.

3 In the **Installed paths** list, select a path that is mapped to a SessionResolver.

4 Click the **Configure Resolver** button.

5 Change the appropriate parameters.

6 Click the **Accept Changes** button.

Your changes take effect immediately and affect all sessions handled by that SessionResolver.

Explicit Control of Sessions

You can explicitly expire sessions at any time. You can expire sessions based on a web hit handled, a server, or a SessionResolver.

Expiring a session abruptly ends that interaction with the user. If the user makes a request of an expired session, a Session Expired alert condition is raised. (See "Alert Conditions" on page B-2.)

Expiring Sessions by Web Hit

To expire the session that handled a particular web hit:

1 In the Server Console, select the server that received the web request (must be a submit request).

2 Click the **View Traffic** button.

3 In the traffic log, select the web hit whose session you want to expire.

4 Click the **Session** button.

5 Select the session in the top list. For reference, the session key is displayed in parentheses.

6 Click the **Expire Session** button.

7 You are warned that the session may have active users. Click **Yes** to expire the session. (You can check the time on the web session if you are concerned about expiring a session that has received recent user input.)

The session is expired.

Expiring Sessions by Server

To expire sessions associated with a particular server:

1 In the Server Console, select the server.

2 Click the **View Traffic** button.

3 Select a web hit.

4 Click the **All Sessions** button.

5 Select a session from the top list. For reference, the session keys are displayed in parentheses.

6 Click the **Expire Session** button.

7 You are warned that the session may have active users. Click **Yes** to expire the session. (You can check the time on the web session if you are concerned about expiring a session that has received recent user input.)

8 Repeat steps 4 through 6 for any other sessions you want to expire.

Note: If the server has a single SessionResolver, you can expire all sessions for that server by expiring all sessions for its SessionResolver. See below for details.

Expiring Sessions by SessionResolver

To expire sessions associated with a particular SessionResolver:

1 In the Server Console, select the server.

2 Click the **Edit Resolver** button.

3 In the **Installed paths** list, select a path that is mapped to a SessionResolver.

4 Click the **Configure Resolver** button.

5 Decide whether you want to expire a single session or all sessions:

- To expire a single session, select the session in the **Active sessions** list and click the **Expire Session** button. For reference, the session keys are displayed in parentheses.
- To expire all of the sessions that belong to the SessionResolver, click the **Expire All Sessions** button. Note that, if the server has more than one SessionResolver, this may be a subset of the sessions for that server.

6 You are warned that the session may have active users. Click **Yes** to expire the session. (You can check the time on the web session if you are concerned about expiring a session that has received recent user input.)

The session is expired.

Shutting Down a Server

Shutting down a server causes that server to stop receiving requests and halts the processing of any web sessions that were in progress. To shut down an active server:

1 In the Server Console, select the server.

2 Click the **Shutdown** button.

Note that the server does not check for running web sessions before it shuts down. Users that try to interact with those sessions will receive error messages indicating that the server is down. You may want to view traffic and check for active web sessions before shutting down the server.

Shutting down a server does not remove it from the server environment. It remains listed in the Server Console as inactive and can be restarted at any time.

Deleting a Server

Deleting a server shuts it down and removes it from the server environment. To delete a server:

1 In the Server Console, select the server.

2 Click the **Delete Server** button.

The server is removed from the Server Console and from the server environment.

3

Application Delivery

Once development of your Web application is complete, you need to package it for delivery and use in a production environment. This process is called *deploying* an application.

In the case of applications built using servlets or server pages, very few modifications in the server environment are required. In the case of applications built to use VisualWave, some additional steps are required.

A prebuilt deployment image is provided on the release media (**`runtime.im`**, located in the **`/web`** directory). To use this deployment image, your application must be extracted into parcels. You then start the deployment image along with your parcels. The pre-built deployment image differs from the development image in that it is more compact, providing the maximum performance in the final server installation.

This chapter describes:

- Loading Applications into the Server Environment
- Making an Application Available to a Server
- Managing Registries
- Making a Deployment Image
- Configuring the Application Server as an NT Service

Overview

At a high level, there are three steps in deploying a Web application with the Application Server. The specific steps vary slightly, depending upon whether you are working with an application developed using server pages and servlets, or an application that uses VisualWave.

In general:

1. Load your application into the server environment. This may involve first extracting the application from the development environment and savng it as a parcel.
2. Configure a server so that it can *launch* your application. In the case or Web Toolkit applications, this may involve setting up configuration files. In the case of VisualWave applications, this involves using tools provided by the Application Server to specify the path users will include in an URL to request your application.
3. Prepare a special deployment image for your applications.

These steps are described in more detail in this chapter. For applications developed using the Web Toolkit, you should also refer to the *Web Application Developer's Guide*.

Loading Applications into the Server Environment

Before deploying a Web application, it is generally extracted from the development environment and saved, either by separating the code into parcels or by filing-out the source code. Either extraction method can be used in order to prepare your application for loading into an Application Server image. However, to load an application into the deployment-only image, you must use parcels.

You can load parcels into the server environment at image startup or once the server environment is already running.

VisualWorks Application Server supports Smalltalk name spaces. When porting your application from an earlier version of VisualWorks, you might consider locating your application in a separate name space (see "Using Name Spaces" on page 3-11).

Additional details on extracting an application are provided in the *VisualWorks Application Developer's Guide*.

Loading Parcels at Startup

You can load parcels during startup of a Server deployment image by using one or both of two command-line options:

- **`-pcl parcelFileName(s)`** Allows you to specify the names of one or more parcel files to be loaded.
- **`-cnf parcelConfigFileName(s)`** Allows you to name one or more parcel configuration files. A parcel configuration file contains the full pathname of each of the parcel files to be loaded. Each pathname must be on its own line.

Parcel file and parcel configuration file options are added to the command line after the image name. For example, to load the Server Monitor from the **`WaveServerMonitor.pcl`** parcel in the Server release directory, you would execute a command such as:

UNIX

```
visual websrv.im -pcl WaveServerMonitor.pcl &
```

NT

```
visual c:\vw7\websrv.im -pcl
   c:\vw7\waveserver\WaveServerMonitor.pcl
```

Command-line loading of parcels is not available in a development image, but becomes available after the Runtime Packager utility has been used to create a deployment image (for details, see: “Making a Deployment Image” on page 3-17).

Neither the parcel file name nor the parcel configuration file name may contain wildcard characters. Both are resolved with respect to the current directory. You can include both options on the same command line.

Loading Parcels after Startup

There are several ways to load parcels into a running server environment:

Tool or Panel	Notes and Recommendations
Parcel Manager	Used to load and unload parcels.
Application Manager	Used to manage registries of *application paths*.
Configure Resolver panel for a RegisteredSessionLauncher	When you load a parcel, automatically creates a path for it and installs that path in the RegisteredSessionLauncher (see “Using a RegisteredSessionLauncher” on page 3-9).

Filing-In Application Code

If your application code is saved in Smalltalk file-out format (usually files with a **`.st`** extension), you can load (file-in) the code into a Server image. You cannot load these **`.st`** files into the deployment-only image; however, you can load parcels.

To file-in Smalltalk code:

1 Open the File Browser by clicking on the file cabinet icon or selecting **File → File Browser** in the Launcher window.

2 Locate your Smalltalk source files by browsing through the hierarchy of directories.

3 Select a file name, then click the <Operate> button and select **File in**.

The Application Server loads the source code into the current image. Repeat this step for all the source files needed for your application.

4 Save your image.

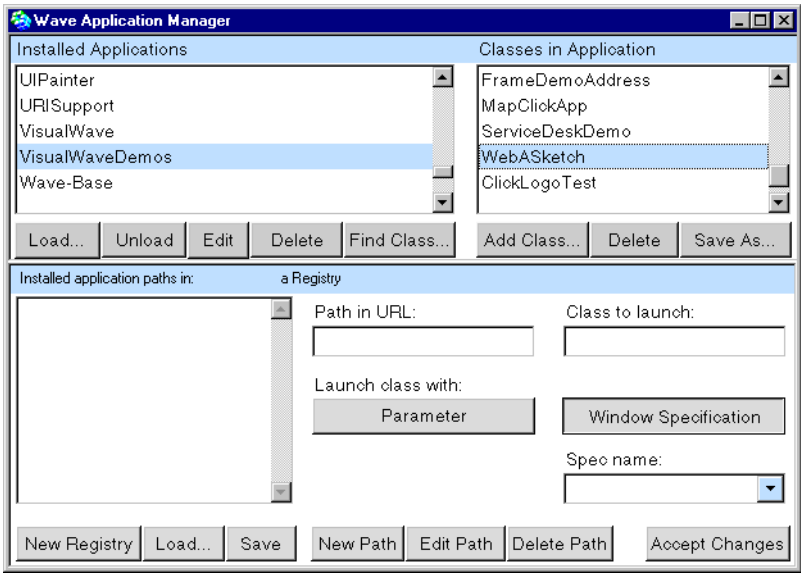
Using the Application Manager

For VisualWave applications, you may also use the Application Manager to load parcels:

1 In the Server Console, choose **Applications → Configure…** .

The top half of the Application Manager lists the parcels that are loaded in the server environment and the classes that are in the selected parcel.

The bottom half of the Manager enables you to create and modify registries of application paths. Registries of application paths are described in more detail in “Managing Registries” on page 3-12.



Application Manager

Initially, the Application Manager shows every parcel that has been loaded in the image.

2 To load a new parcel, click the **Load…** button in the top left quadrant of the Application Manager.

3 When prompted, enter the name of the parcel file to load and click **OK**.

The parcel is loaded into the server environment, with its name appearing in the **Installed Applications** list at the top of the Application Manager.

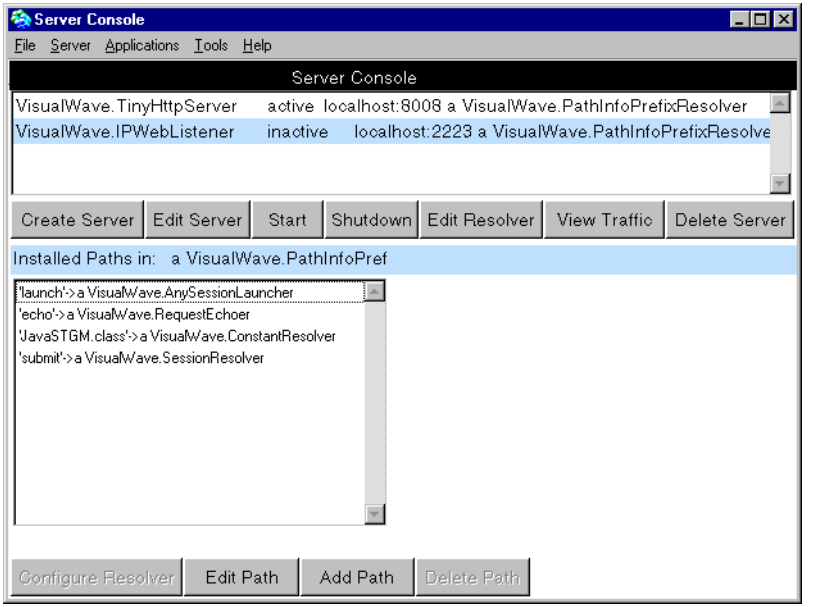
Using the Configure Resolver Panel

To load a parcel by using the Configure Resolver panel for a RegisteredSessionLauncher:

1 In the Server Console, select a server.

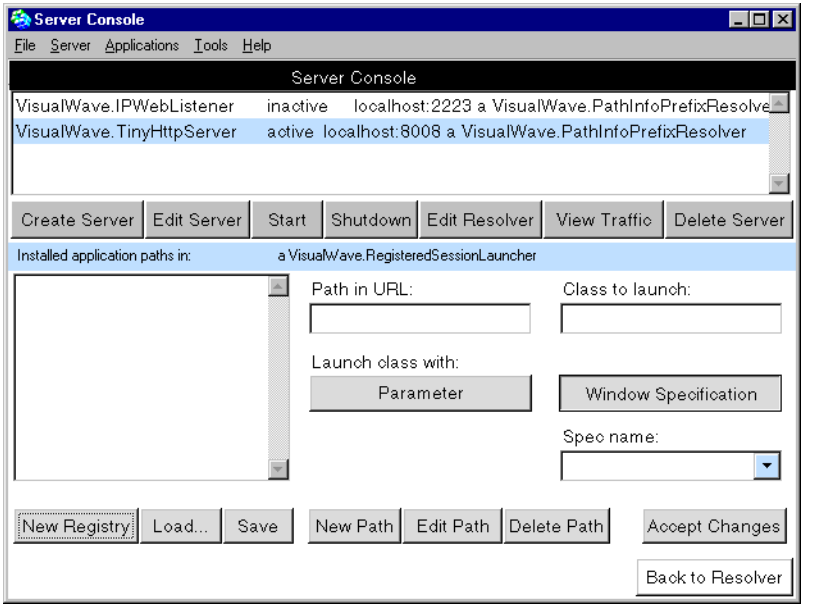
2 Click the **Edit Resolver** button.

The Edit Resolver panel lists the path prefixes that are recognized by the server when it attempts to resolve the requesting URL. The figure below shows the path prefixes that are set up when you specify **Use default resolvers** for the server.



Configuring a Resolver

3 In the **Installed Paths** list, select a path that is mapped to a RegisteredSessionLauncher and click the **Configure Resolver** button. The resolver panel appears in the Console window.



Showing Details about a Resolver

The Configure Resolver panel lists the application paths that are recognized by the RegisteredSessionLauncher when it attempts to resolve the requesting URL. Initially, this registry is empty.

4 Click the **Load...** button.

5 VisualWorks prompts you for the kind of item to load and the location from which to load it. To load a parcel, choose the **From Parcel** radio button and enter the filename for the parcel.

6 Click the **Load** button to continue.

VisualWorks loads the parcel into the environment.

When you load a parcel from the Configure Resolver panel, VisualWorks also performs the necessary steps to make the parcel's applications available to users. It automatically:

- Creates an application path for any applications (subclasses of ApplicationModel) that are in the parcel. The default path uses the class name as the path name.
- Installs the path(s) in the RegisteredSessionLauncher's registry.

Making an Application Available to a Server

When using VisualWave, the steps that you take to make your application available to a server depend on the kind of launching resolver that you want to use to start the application. There are two options, either using an AnySessionLauncher or a RegisteredSessionLauncher:

AnySessionLauncher:
This type of launching resolver can start *any* subclass of ApplicationModel. Applications do not have to be registered. AnySessionLaunchers are especially useful for testing and other internal uses.

RegisteredSessionLauncher:
This type of launching resolver can only start applications for which there are entries in the launcher's registry. You can set up multiple RegisteredSessionLaunchers, each of which provides access to a different application or group of applications. You can also set up different RegisteredSessionLaunchers to use SessionResolvers with different timeout policies. RegisteredSessionLaunchers are common for production use.

Using an AnySessionLauncher

If the main class for your application is a subclass of ApplicationModel, you can make it visible to a server simply by setting up a path prefix that maps to an AnySessionLauncher:

1 In the Server Console, select a server.

2 Click the **Edit Resolver** button.

3 Check the **Installed Paths** list to ensure that there is a path prefix that maps to an AnySessionLauncher.

If a path exists for an AnySessionLauncher, you can use that path. (In the Server development environment, servers set up to use default resolvers map the path prefix **launch** to an AnySessionLauncher.)

If a path does not exist for an AnySessionLauncher or if you want to create another path for an AnySessionLauncher (perhaps with a different path keyword), follow these steps:

a Click the **Add Path** button.

b From the **Resolver Type** menu, choose **AnySessionLauncher**.

c From the **Session Resolver** menu, choose the SessionResolver that you want to manage applications started by the

AnySessionLauncher. Different SessionResolvers may have different timeout policies.

d In the **Path** field, enter the string that you want users to include in the URL when requesting the launching of an application. You can use any string that is a valid segment of an URL.

e Click the **Accept** button.

Your application is available. To start your application, use an URL of the form:

http://hostname:port/launch/ClassName

where:

- The **hostname** and **port** specify the server from step 1.
- **launch** is the path prefix from step 3d.
- **ClassName** is the name of the main class for your application. That class must be a subclass of ApplicationModel.

Note: If your application is constructed using name spaces, you should specify a fully-qualified class name. For details, see "Using Name Spaces" on page 3-11.

Using a RegisteredSessionLauncher

A RegisteredSessionLauncher can start only applications that have entries in the launcher's application registry. The first three steps for setting up a RegisteredSessionLauncher are similar to those for setting up an AnySessionLauncher; the final steps, in which you set up the registry, are different:

1 In the Server Console, select a server.

2 Click the **Edit Resolver** button.

3 Check the **Installed Paths** list to ensure that there is a path prefix that maps to a RegisteredSessionLauncher.

If a path exists for a RegisteredSessionLauncher, you can use that path. (In the Server environment, servers set up to use default resolvers map the path prefix **launch** to a RegisteredSessionLauncher.)

If a path does not exist for a RegisteredSessionLauncher or if you want to create another RegisteredSessionLauncher (perhaps to handle another set of applications), follow these steps:

a Click the **Add Path** button.

b From the **Resolver Type** menu, choose **RegisteredSessionLauncher**.

c From the **Session Resolver** menu, choose the SessionResolver that you want to manage applications started by the RegisteredSessionLauncher. Different SessionResolvers may have different timeout policies.

d In the **Path** field, enter the string that you want users to include in the URL when requesting the launching of an application via this RegisteredSessionLauncher. You can use any string that is a valid segment of an URL.

e Click the **Accept** button.

4 Display the RegisteredSessionLauncher's registry:

a In the **Installed paths** list, select the path that maps to the RegisteredSessionLauncher from step 3.

b Click the **Configure Resolver** button.

5 Add an application path for your application. There are two ways to do this:

6 Create a new path.

7 Load a registry that includes a path for your application. You can load a registry from a parcel or from a file.

The following steps describe how to create a new path:

a Click the **New Path** button.

b In the **Path in URL** field, enter the string that you want users to enter in the URL — after the launch request — to request your application.

c In the **Class to Launch** field, enter the name of the main application model class for your application. This is the class to which the RegisteredSessionLauncher will send an `open` request.

If your application is located in a separate name space, you should specify a fully qualified class name in this field. For details, see "Using Name Spaces" on page 3-11.

d To open the class with a window specification, click the **Window Specification** button and choose a window specification resource from the **Spec name** menu.

e To initiate an arbitrary message-send when the path is requested, click the **Parameter** button and enter the message and the name of the class to which to send the message.

f Click the **Accept Changes** button.

(For the steps to load an existing registry, see "An Example: Click Logo Tester" on page 3-13.)

Your application is now available. To start your application, use an URL of the form:

http://hostname:port/launch/ApplicationPath

where:

- The **hostname** and **port** specify the server from step 1.
- **launch** is the path prefix from step 3d.
- **ApplicationPath** is the string from the **Path in URL** field that you entered in step 5b.

Note: If your application is constructed using name spaces, you should specify a fully-qualified class name.

Using Name Spaces

VisualWorks Application Server provides support for name spaces. If you choose to use name spaces when developing your application (strongly recommended), then you'll need to use fully-qualified names when loading it in the server image. You can specify fully-qualified class names in the Resolver Configuration and Registered Session Editors.

When launching your web application, you must use fully-qualified paths, e.g.:

http://hostname:port/launch/MyAppNameSpace.MyClassName

where:

- **MyAppNameSpace** identifies the name space which contains your application classes.
- **MyClassName** is the name of the main class for your application.

If you do not launch the application by using a fully-qualified name, the Application Server will perform an alphabetical search through the name space hierarchy and launch the first class with the name matching MyClassName. Proceeding alphabetically within Root.*, the search will thus

pass through Smalltalk.* and then VisualWave.*. If your development or server image contains multiple classes with the same name, you should use a fully qualified name to specify the proper class.

When porting an application to the Server environment, you can begin by using the name space Smalltalk.*. Ideally, you should create a new name space to contain your application. For a longer discussion of name spaces, see the *VisualWorks Application Developer's Guide*.

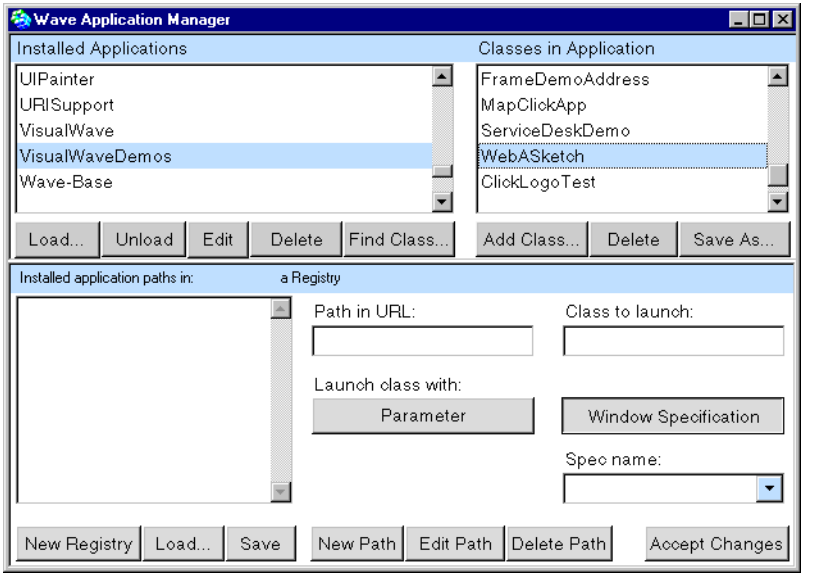
Managing Registries

The section titled "Using a RegisteredSessionLauncher" on page 3-9 described how to enter an application into the registry for a RegisteredSessionLauncher.

As an alternative to adding applications one-by-one to the current registry, you can create registries, save them as files or parcels, and then load them into RegisteredSessionLaunchers. Using registries makes it easier to configure additional RegisteredSessionLaunchers, additional servers, and additional server environments.

You use the Application Manager to create and save registries.

To open the Application Manager, choose **Applications → Configure…** in the Server Console.



Application Manager

You create registries by adding and editing paths. You then save the registry to a file or with a parcel.

Once a registry is saved, you can load it into the environment by clicking **Load…** in the Application Manager. You can load it into a RegisteredSessionLauncher by clicking **Load…** in the Configure Resolver panel for the launcher.

The following example provides detailed steps for setting up a registry, saving it, and loading it into a RegisteredSessionLauncher. For details, see "Application Manager" on page A-29.

An Example: Click Logo Tester

The Server development environment comes with several sample applications. These applications are delivered in a parcel file named **`VisualWaveDemos.pcl`** in the Server developer release directory **`wavedev`**. One of the example applications is a clickable image map.

This section walks you through the steps of loading the sample Click Map Tester application and making it available to users.

Load the Application Parcel

The first step to making the application available is loading it into the Server environment.

To load the Click Logo Test as a parcel:

1 In the Server Console, choose **Applications → Configure…**.

The top half of the Application Manager lists the parcels that are in the server environment and the classes that are in the selected parcel.

2 Click the **Load…** button in the top left quadrant of the Application Manager.

3 When prompted, select the name of the parcel file containing the test application, and click the **OK** button. The Click Logo Test is in the parcel file **`VisualWaveDemos.pcl`** in the Server developer release directory **`wavedev`**.

The contents of the parcel, including the ClickLogoTest class, are loaded into the server environment. The parcel's name appears in the **Installed Applications** list at the top of the Application Manager.

Set Up a Registry

The next step is to set up a path for the Click Logo Test in a registry of applications. VisualWorks automates the process if the application is in a parcel. If it is not, you need to specify the path as described in "Using an AnySessionLauncher" on page 3-8. Our example is in a parcel:

1 Click the **Load...** button in the bottom left quadrant of the Application Manager.

2 VisualWorks prompts you for the source from which to load a registry. Choose to load an initial registry from the **`VisualWaveDemos.pcl`** parcel:

   a Click the **Parcel** radio button.

   b In the list of parcels, select **ClickLogoTest**.

   c Click the **Load** button.

   VisualWorks checks to see if the parcel already contains a registry. If the parcel does not have a registry, VisualWorks searches the parcel for classes that are subclasses of ApplicationModel and sets up a registry that contains paths for those classes. In this case, there is one such class: ClickLogoTest. A **ClickLogoTest** path is added to the **Installed application paths** list and to the current registry.

Note: When VisualWorks creates a registry from a parcel, it does not save that registry back into the parcel automatically; the registry exists only in the context of the Application Manager until you explicitly save it.

---

3 To view the details on the Click Logo Test, select its entry in the **Installed application paths** list.

   Notice that the path created by the Server uses the application model class's name as the initial value for the **Path in URL**. This is useful for quick setup, but you may want users to enter something other than the class name when requesting the application—to hide your class names, to make the URL easier to type, or more descriptive.

4 Change the value for **Path in URL** to **LogoTest** and then click the **Accept Changes** button.

   Notice that in the **Installed application paths** list, the **ClickLogoTest** entry is changed to **LogoTest**, and in the **Class to launch** field, the class name has been fully qualified to indicate both the class and the name space (e.g., VisualWave.ClickLogoTest).

Although the registry could contain any number of paths, including paths to applications that are in other parcels, for this example, the registry will contain only the one **LogoTest** path.

5 Now that you are done creating the registry, you should save it. You can save it to either a file or to a parcel. For this example, save it to the ClickLogoTest parcel:

a Click the **Save** button.

The Application Server prompts for the location to which to save the registry.

b Choose the **Save in Parcel** radio button.

If the application is not in a parcel, you would save the registry in a file. You can then load the file again later.

c In the list of parcels, select **ClickLogoTest**.

d Click the **Save** button.

VisualWorks prompts you to save the parcel (now containing a registry) to a parcel file. If you do, loading the parcel from the file will also load the registry. For this example, saving the parcel to a file is not necessary.

The registry is now created and saved. However, the application is not yet accessible to web clients. The registry must first be installed in the RegisteredSessionLauncher for the server that receives web requests.

Set Up a Server

Create and start a server with the following characteristics:

Server Type	Smalltalk HTTP Server
Hostname	localhost
Port	8008
Virtual Directories	
Use default resolvers	checked
Start at system startup	not checked

For more information, see "Creating a Server" on page 2.

Set Up the Server's RegisteredSessionLauncher

Although the server is running and able to receive web requests, it is not yet able to start your application. To start your application, the server must have a launching resolver that knows how to find your application. To set up a RegisteredSessionLauncher with the registry that you created above, follow these steps:

1 In the Server Console, select the **Smalltalk HTTP Server**.

2 Click the **Edit Resolver** button.

3 Because you chose to set the server up to use default resolvers, the **Installed Paths** list includes a path that maps the **launch** keyword to a RegisteredSessionLauncher:

```
'launch'->a RegisteredSessionLauncher
```

4 Display the RegisteredSessionLauncher's registry:

   a In the **Installed paths** list, select the **launch** path.

   b Click the **Configure Resolver** button.

5 Load the registry that you created previously (following the steps described in "Set Up a Registry" on page 3-14):

   a Click the **Load…** button.

   The Application Server prompts you for the location from which to load the registry.

   b Select the **Registry** and **From Parcel** buttons, and select the **ClickLogoTest** parcel from the list.

   c Click the **Load** button to continue.

The Application Server loads the registry from the **ClickLogoTest** parcel and installs it in the RegisteredSessionLauncher. The **LogoTest** path can now be used to start your application.

Run the Application

The Click Logo Test is now available. To start it, open a web browser on the host machine and request the following URL:

http://localhost:8008/launch/LogoTest

Making a Deployment Image

If you have built a Web application using the Application Server development environment, simply saving this image is generally not adequate for deployment purposes.

Web applications may be deployed in one of three ways:

- Application code is saved as one or more parcels. These parcels are then loaded into the pre-built runtime image (**`runtime.im`**, located in the **`/web`** directory of the standard distribution). For details, see: "Loading Parcels at Startup" on page 3-3.
- The development image may be saved as a "headless" image, containing all the tools available in the development environment, but modified to run as a background process with no access to the host machine's display.
- Finally, you can use the Runtime Packager on your working image to "strip" all application development code. This creates a stand-alone image that can be more compact. The Application Server provides a simplified interface to the Runtime Packager.

Creating a Headless Image

When deployed in "headless" mode, a VisualWorks image runs as a background process that does not interact with the host machine's display.

To create a headless image:

1 Starting with the full VisualWorks development environment, make a backup copy of the current image.

2 To ensure that the server is activated on image startup, follow these steps:

   a Open the Server Console by clicking on the Server Console button in the Launcher window.

   b In the Server Console window, select the server you want to activate on startup, and click on the **Edit Server** button.

   c Select the **Start server on system start up** checkbox.

   d Click on the **Accept Changes** button.

3 When deploying to a specific host, you may set **Hostname** to that of the deployment server (optional).

4 In a Workspace, evaluate the following expressions:

```
WaveDeploymentNotifier installInServer
Transcript := WaveDeploymentNotifier
```

Since the Transcript window will be removed from a deployment image, the second expression redirects all Transcript messages to a special notifier.

5 To create the headless image, pick **Save Headless Development Image As...** from the **Web** menu in the Launcher window. A dialog box prompts for the name of the new image (e.g., **`headless`**). Enter the file name (do not include the file extension **`.im`**) and click on **OK**.

Using the Runtime Packager

The Runtime Packager can be used to strip non-essential parts of a development image and package it for deployment. When invoking the Runtime Packager from VisualWorks, headless operation is also enabled.

To create a deployment image using the Runtime Packager, follow these steps:

1 Starting with the full VisualWorks development environment, make a backup copy of the current image.

2 To ensure that the server is activated on activated on image startup:

a Open the Server Console by clicking on the Server Console button in the Launcher window.

b In the Server Console window, select the server you want to activate on startup, and click on the **Edit Server** button.

c Select the **Start server on system start up** checkbox.

d Click on the **Accept Changes** button.

3 In a Workspace, evaluate the following expressions:

```
WaveDeploymentNotifier installInServer
Transcript := WaveDeploymentNotifier
```

Since the Transcript window will be removed from a deployment image, the second expression redirects all Transcript messages to a special notifier.

4 Choose **Web →Save Production Runtime As...** in the Launcher window. A dialog box prompts for the name of the new image (e.g., **`runtime`**). Enter the file name (do not include the file extension **`.im`**) and click on **OK**.

A dialog box will show the name of the global configuration file used by the new production image. By default, this is **webtools.ini**. To restore this setting to the default, select **Web →Use Default Configuration**.

5 Save the stripped image to a new image name.

6 Once the new image has been saved, the Runtime Packager will automatically exit. Restart the new image to continue the procedure.

7 Restart the new image a second time to complete the packaging procedure.

The deployment image is now ready for use.

Optimizing for Deployment

When you use the Runtime Packager to create a development image, there are some additional settings that you may want to adjust for optimal memory performance and exception handling in the final deployment image.

Checking the Memory Policy

The behavior of the Smalltalk object memory is changed during the development process. The developer image uses the standard MemoryPolicy. Its counterpart in a server image is ServerMemoryPolicy. ServerMemoryPolicy has different low space threshold behavior for terminating processes and initiating garbage collections. To check whether an instance of MemoryPolicy or ServerMemoryPolicy is currently installed, print the result of the following Smalltalk expression:

```
ObjectMemory currentMemoryPolicy
```

If it is necessary to reinstall the ServerMemoryPolicy, evaluate the following expression using **Do It** (do this before using the Runtime Packager to produce the deployment image):

```
ServerMemoryPolicy reinstallFromSettings
```

When you apply any changes made in the Memory Configuration page of the Server Settings tool, the ServerMemoryPolicy is automatically reinstalled. If you want to operate with MemoryPolicy for development, you'll need to reinstall it after changing the Server settings.

Checking the Server Notifier Mechanism

During application development, you may wish to have exceptions appear as standard VisualWorks notifier dialogs. In a server environment, these are replaced by logged errors or messages in the Server Notifier window.

During development, the expression ProcessEnvironment isDeveloping answers true. This class message is used to specialize the behavior of a number of tools. For example, it determines whether the default launching resolver is an AnySessionLauncher (development) or RegisteredSessionLauncher (server) and whether or not the menu bar is shown in the Server Console.

When all the optional Server parcels are loaded, this method is modified to return false. When isDeveloping answers false, the Server Console's **Create Server** and **Edit Server** panels are altered so that the options for trapping errors and raising exceptions no longer appear. Also, a standard notifier won't be raised if the error was caused by a submit from the web. If you want to see regular notifiers and other standard development toolsets, you need to change ProcessEnvironment isDeveloping so that it returns true.

Before you strip the image for deployment, make sure that ProcessEnvironment isDeveloping returns false.

Configuring the Application Server as an NT Service

Windows NT, Windows 2000, and Windows XP allow you to configure the Application Server as a service, and to start it automatically on system boot. To deploy your Web application as a service, you must start with a (headless) production runtime image.

To save a development image as a production runtime image, use the Launcher window to select **Web → Save Production Runtime As…**.

Follow the three-step save process to build your production image.

In order to configure VisualWorks as a service you need the **`srvany.exe`** utility included in the Windows Service Pack for your operating system (e.g., for Windows NT, Service Pack 3 or later).

First, install the operating system specific Windows Service Pack; then follow these steps to install and configure your VisualWorks Application Server to start-up automatically when the machine restarts. (Refer to the Windows documentation for your operating system for additional information.)

1 Install your service.

Do not use the Wizard. Instead, use the command line to install the service. At a MS-DOS command prompt, type the following command:

```
<path>\INSTSRV.EXE <MyService> <path>\SRVANY.EXE
```

where:

`<path>` is the location of the Windows Resource Kit

(usually **`C:\Program Files\Resource Kit`**)

`<MyService>` is the name of the service you are creating

(e.g., **`VWebService`**)

This should result in the message **Create Service SUCCESS**.

2 Edit the Windows Registry to configure your service.

Caution: Editing the registry is always a risky operation, and if done incorrectly, can cause serious problems that may require you to reinstall your operating system.

You should follow all necessary precautions and always back up the registry before you edit it.

a Run the Registry Editor (**`regedt32.exe`**) and locate the following subkey:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<MyService>

b From the **Edit** menu, select **Add Key**, and enter the following:

Key Name : Parameters
Class : <leave blank>

c Click **OK** to create the new registry key.

d Select the **Parameters** key you just created.

e From the **Edit** menu, select **Add Value**, and enter the following:

Value Name : Application
Data Type : REG_SZ
String : <VisualWorks-CommandLine>

where:

<VisualWorks-CommandLine> is the complete set of command line arguments to execute your VisualWorks application, including the vm, image and all other command line parameters, e.g.:

c:\vw\bin\win\vwnt.exe myImage.im -pcl myParcel.pcl

Note that it is not necessary to fully qualify the image path or the parcel path if they are located in the AppDirectory defined below.

f Click **OK** to add the value.

g From the **Edit** menu, select **Add Value**, and enter the following:

Value Name : AppDirectory
Data Type : REG_SZ
String : <VisualWorks-AppDirectory>

where:

<VisualWorks-AppDirectory> is the current or "start in" directory for your application, which will normally contain your image, parcels and any configuration files, for example:

c:\vw\image

h Click **OK** to add the value.

i Close the Registry Editor.

3 Open the **Services** control panel.

4 Select your service and edit its Properties. Set the service to log on as **Local System account** (unless your application is required to log on with a specific account) and choose **Allow Service to Interact with Desktop**.

By default, the service you create with this procedure is configured to run automatically when the system is restarted. If you wish to change this setting to **Manual**, change the **Startup Type** in the Properties for your service.

You may now start the service from the **Services** control panel, or shut down and restart the machine. The VisualWorks Application Server should start up as expected.

4

Server Architecture

Architecturally, the VisualWorks Application Server is based upon a framework for presenting Smalltalk applications to clients on the Internet.

Serving applications to a client's web browser requires the cooperation of a variety of agents, both inside and outside of the VisualWorks Application Server environment.

This chapter provides an overview of:

- The external communication model between the user's web browser, a standard HTTP server, Common Gateway Interface (CGI) scripts, and the Application Server.
- How the Application Server processes a web request.
- How the Server resolves URLs to invoke applications.
- How the Server launches an application, establishes sessions with specific clients, and organizes transactions within sessions.

External Communication

Standard Interaction

On the World Wide Web, the standard pattern of interaction between a client and an HTTP server is as follows:

1. The user enters a Uniform Resource Locator (URL) in a web browser.

   The URL begins with a hostname and port designation (port number 80 is assumed if not specified). The rest of the URL is path information that may include standard directory information, but also may indicate that a Common Gateway Interface (CGI) script or other programmatic service is required. The last piece of the URL names the specific item being requested, such as a web page, a CGI script, or an application name.

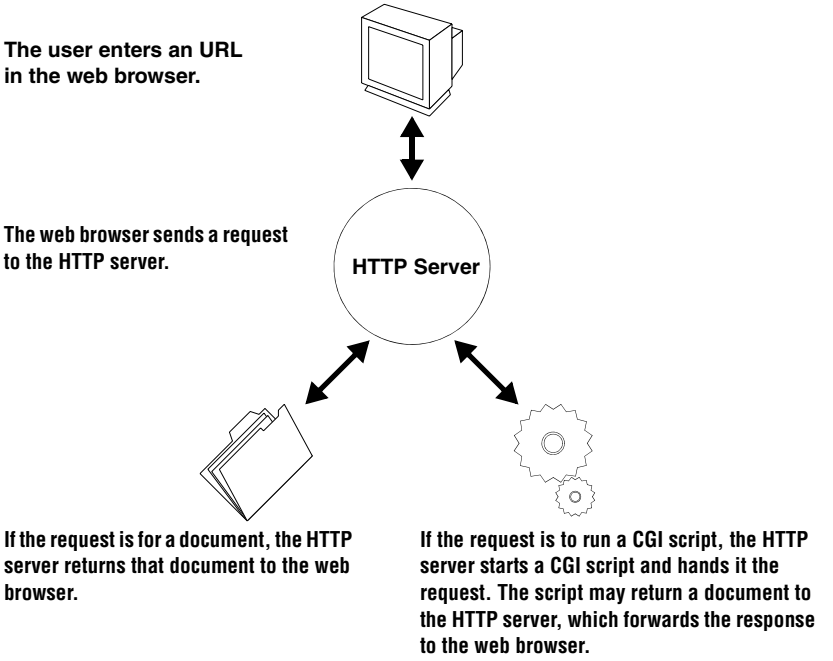
2. The web browser uses the hostname and port number to contact the HTTP server to which to send the request. Typically, this is accomplished by opening an IP/TCP stream over the Internet.

3. Once the HTTP stream has been established, the server uses the body of the URL to determine how to respond to the client's request.

4. If the request is for a document, the server returns that document to the web browser.

5. If the request specifies a CGI script to run, the request is handed off to that script. The CGI script processes the request and returns the result to the HTTP server, which forwards the result to the user's web browser.

The following diagram illustrates this HTTP transaction:



Interaction with the VisualWorks Application Server

The VisualWorks Application Server supports two types of servers:

Smalltalk HTTP Server

This server is used for development and testing of VisualWorks applications. It receives requests from and returns answers directly to the user's web browser, without an intermediate HTTP server. This is the Application Server's native server implementation.

External Web Server

This type of server receives requests from and returns answers to a standard HTTP server via a gateway or ISAPI. Interaction with HTTP servers requires the optional CGI or ISAPI components (for details, see "Using Front-End HTTP Servers" on page 5-1).

Interaction with the Smalltalk HTTP Server

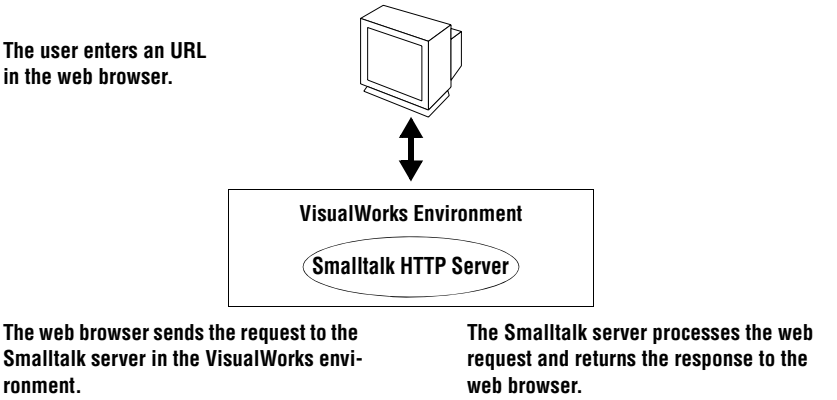
The Smalltalk HTTP Server is the native VisualWorks HTTP server. It is intended primarily to be used for development and testing purposes rather than as a production HTTP server.

This server receives requests directly from the client's Web browser and returns the appropriate response. Its answer can contain any HTML codes, including form definitions, and it can serve external files by MIME type. The response can also include references to external files that are available from other HTTP servers. The pattern of interaction between a client and a Smalltalk HTTP Server is as follows:

1 The user enters an URL in the Web browser. This URL specifies the host machine on which the VisualWorks Application Server is running and the port number on which a Smalltalk server in the environment is listening.

2 The Web browser uses the first part of the URL (the hostname and port number) to determine the HTTP server to which to send the request. In this case, the hostname and port number specify a particular server in the VisualWorks environment.

3 The Smalltalk server receives the request, processes it, and returns an answer directly to the user's Web browser.



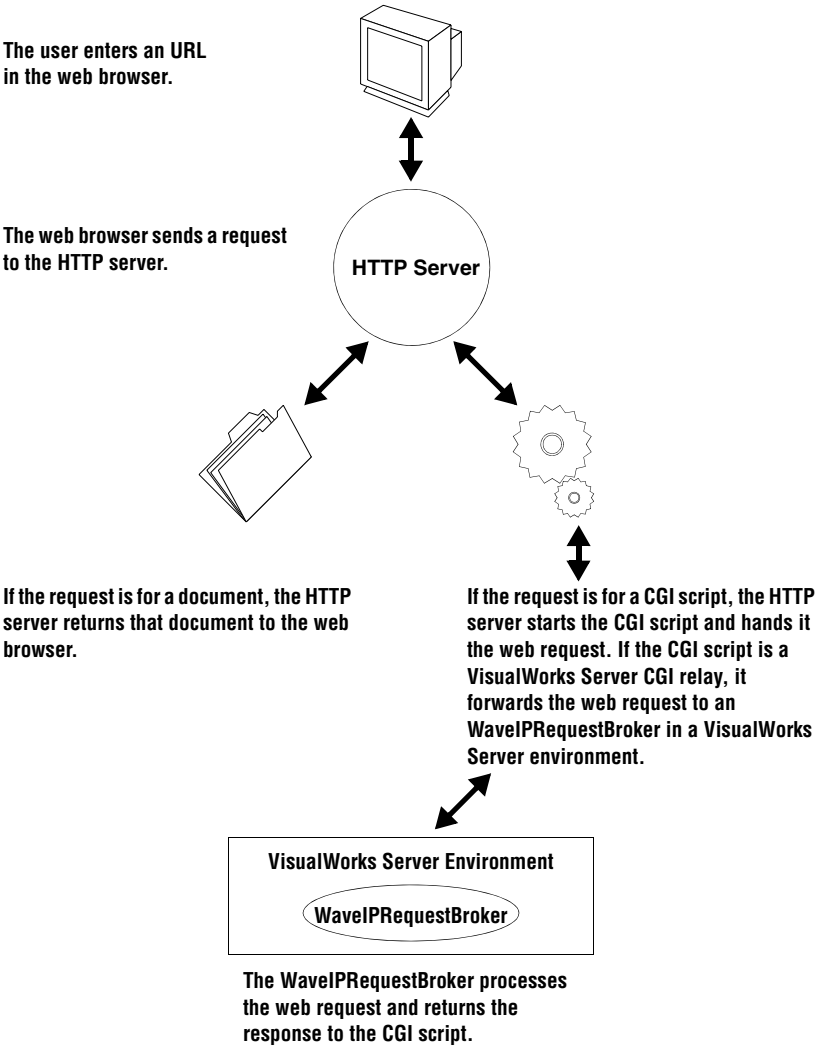
Interaction with an External Web Server

An External Web Server works in conjunction with a commercial HTTP server (e.g., Apache, IIS) via a CGI or ISAPI relay program.

Within the VisualWorks environment, this server is called an WaveIPRequestBroker. It is typically used for a production or release version of your VisualWorks application. In order to use a CGI relay program, you must choose the External Web Server.

The pattern of interaction between an end user and an External Web Server (WaveIPRequestBroker) via the CGI relay is as follows:

1. The user enters an URL in the Web browser. This URL specifies the host machine on which the Application Server is running and the port number on which an WaveIPRequestBroker in the VisualWorks environment is listening.
2. Using the first part of the URL (the hostname and port number) as a network address, the Web browser sends a request to a specific HTTP server. By convention, if no port number is specified, port 80 is assumed. In this case, the hostname and port number specify a standard HTTP server.
3. The HTTP server receives the request and processes the next part of the URL to determine if the request is for a document that it can serve up or for a CGI script to be run. If the request is for a CGI script to be run, the HTTP server starts up the CGI script and passes the web request to it.
4. The CGI script locates the Application Server process, and forwards the web request to an WaveIPRequestBroker.
5. The WaveIPRequestBroker receives the request, processes it, and returns the answer to the CGI relay.
6. The CGI relay returns the answer to the HTTP server.
7. The HTTP server returns the answer to the user's Web browser.


The user enters an URL in the web browser.
The web browser sends a request to the HTTP server.
HTTP Server
If the request is for a document, the HTTP server returns that document to the web browser.
If the request is for a CGI script, the HTTP server starts the CGI script and hands it the web request. If the CGI script is a VisualWorks Server CGI relay, it forwards the web request to an WaveIPRequestBroker in a VisualWorks Server environment.
VisualWorks Server Environment
WaveIPRequestBroker
The WaveIPRequestBroker processes the web request and returns the response to the CGI script.

Resolving Client Requests

The Application Server provides complete facilities for hosting Web applications. In order to direct incoming requests to your application, the server must first *resolve* the request.

Requests may be resolved to Smalltalk Server pages, servlets, or VisualWave applications. The Application Server supports the use of all three of these server technologies simultaneously.

Server pages are processed by a scripting engine native to the VisualWorks Application Server. Servlets and VisualWorks applications are executed as Smalltalk application code that is loaded into the server environment.

In the case of applications served using VisualWave, this procedure may involve *launching* a new instance of your application, thereby creating a new *session*. If your application has already been started, and if a client is in the middle of a transaction, the server must also keep track of the particular *phase* in the current transaction.

Regardless of the server technology, the Application Server provides a transparent mechanism to automatically keep track of all session information, resolving concurrent incoming client requests to the appropriate session/instance of your application.

Putting Objects Together to Resolve a Web Request

Inside the Application Server, a group of objects cooperate to answer a web request. The particular set of objects is determined in part by the URL used to initiate the web request and in part by how you have configured the resolvers.

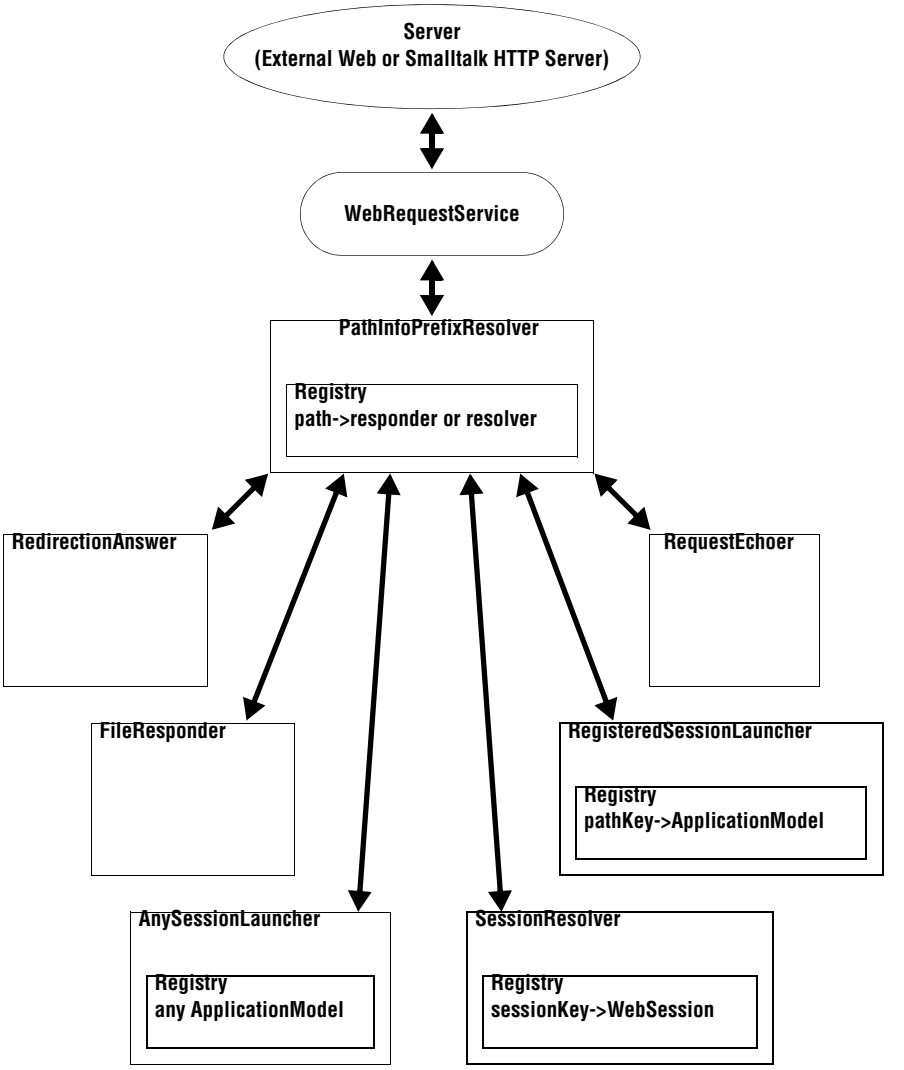
The process of answering a web request follows these steps:

1 The user makes a request of the Server. The URL determines the external communication path that the request follows to reach the Application Server environment (as described in the preceding section). The URL also determines the particular server to contact within the VisualWorks Application Server (either a Smalltalk HTTP Server or an External Web Server).

2 The server listens on a port, and when it receives an incoming HTTP stream, it passes the stream to a WebRequestService object.

3 The WebRequestService does all of the “serving.” It takes the stream of bits and composes a web request. Finally, it hands the web request to the server’s PathInfoPrefixResolver.

This process of resolving an incoming web request is illustrated in the following diagram:



4 The PathInfoPrefixResolver looks at the first segment of the path in the URL and checks its registry for a matching path key. The key

associates the path with one of six types of web responders and resolvers:

- RequestEchoer
- RedirectionAnswer
- FileResponder
- AnySessionLauncher
- RegisteredSessionLauncher
- SessionResolver

The PathInfoPrefixResolver then hands the web request to the appropriate responder or resolver.

5 The next step depends on the kind of responder or resolver:

- **RequestEchoer:** Returns the environment variables and other information.
- **RedirectionAnswer:** Returns a redirection instruction and alternate URL. Also returns optional message text to display in browsers that do not support redirection.
- **FileResponder:** Returns a requested external file.
- **AnySessionLauncher:** Starts the application named in the path of the URL. The application must be a subclass of ApplicationModel.
- **RegisteredSessionLauncher:** Starts the application referred to in the path of the URL. The application must have an entry in the RegisteredSessionLauncher's registry.
- **SessionResolver:** Hands the web request to an application that is already running in the web session. The SessionResolver checks hidden form data or cookies for a session key and then matches that session key to a web session.

Launching an Application

When there is a request to launch a new application, the launching resolver:

1 Checks the current memory load on the Application Server and checks the setting for Defer Launch Requests.

If the memory stress is at or above the Defer Launch Requests limit, the application is not launched and a Launch Refused alert condition is raised.

2 If the memory stress level is acceptable, the launching resolver finds the requested application:

- AnySessionLaunchers can find any subclass of ApplicationModel.
- RegisteredSessionLaunchers can find any class for which there is a path in the launcher's registry.

3 Finds the appropriate look policy. The look policy is determined by the User_Agent in the web request and the values for the User_Agent Patterns settings BasicHTMLLookPolicy and EnhancedHTMLLookPolicy (in the Settings "Look Policy"; for details, see "User_Agent Patterns" on page B-7). The default is EnhancedHTMLLookPolicy.

4 Creates a web session and installs the look policy in that session.

5 Tells its SessionResolver to register the web session.

6 Tells the web session to open the application model.

7 Returns a SessionResponder. The answer for the request is a web page generated by the application.

The URL's Role

As the preceding sections have mentioned, the URL that is used to initiate the web request in large part determines the specific agents — both outside and inside the Application Server environment — that cooperate to respond to the request.

From the client's perspective, the URL is parsed into a series of parts:

Transfer Protocol

The URL begins with the transfer protocol to use. For web requests, this protocol is always HTTP:

```
http://
```

HTTP Server

The next part of the URL determines the host machine and port number of the HTTP server to which the request is initially sent. The host machine may be specified either by machine name or by IP address. A port number may be included after a colon. If no port number is included, port 80 is assumed:

```
http://www.cincom.com/
http://localhost:8008/
http://127.0.0.1/
```

With VisualWorks Application Server, this may be a Smalltalk HTTP Server within the VisualWorks environment, or a commercial HTTP server that will forward the request to an IPWebListener in the Server environment.

CGI Script

Following the hostname and port number, the URL may include the location of a Common Gateway Interface (CGI) script. CGI scripts are used in conjunction with standard HTTP servers. When the HTTP server receives a request, it looks at the URL to determine if the request is for a CGI script. If it is, the HTTP server runs the specified script and passes it the web request.

HTTP servers support two methods of locating CGI scripts. The HTTP server may assume that any file in a particular directory (usually **`cgi-bin`**) is a CGI script or it can assume that any file with a particular extension (usually **`.cgi`**) is a CGI script.

```
http://www.cincom.com/cgi-bin/scriptName/
http://127.0.0.1/scriptName.cgi
```

VisualWorks Application Server provides CGI scripts and applications for relaying a web request to the Server environment. Refer to "Using Front-End HTTP Servers" on page 5-1 for more information.

Path Prefix

The remaining part of the URL is the *path*. The Application Server uses the first segment of this path to determine the kind of resolver or responder to handle the web request.

For example, consider these two sample URLs:

```
http://www.yoursite.com/cgi-bin/cgi2vw/echo
http://webhost:8008/launch/WebBrowser
```

In the first URL, the path is **echo**, and **echo** determines the kind of resolver used to handle the request. In the second URL, the path is **launch/WebBrowser**, and **launch** determines the kind of resolver used to handle the request

More specifically, the PathInfoPrefixResolver for a server matches the first segment to a path keyword in its registry. In the registry, the path keyword is mapped to a type of web responder or web resolver. In the examples above, the path keywords are **echo** and **launch**. The default resolvers associated with these keywords are a RequestEchoer and a RegisteredSessionLauncher, respectively.

Before forwarding the web request, the PathInfoPrefixResolver edits the path info in the web request so that it no longer contains the “used” first segment.

The PathInfoPrefixResolver hands the web request to the appropriate responder or resolver, which may in turn hand the request to other objects. Any remaining segments of the URL are used by the final handling object.

Additional Path Information (optional)

Any additional information included in the path is used by the resolver that received the web request from the PathInfoPrefixResolver. Launching resolvers use the additional path information to determine which application to start. For example, consider these two URLs:

```
http://www.yoursite.com/cgi-bin/scriptName/launch/WebBrowser
http://webhost:8008/launch/WebBrowser
```

In each URL, the additional path information **WebBrowser** specifies the class to be started. If the launching resolver is an AnySessionLauncher, the additional path information can be the name of any subclass of ApplicationModel. If the launching resolver is a RegisteredSessionLauncher, the additional path information should be a path keyword that is included in the registry for the launcher. In the registry that path keyword is mapped to the name of a subclass of ApplicationModel.

The additional path information can also identify a *name space*. If you choose to use name spaces while building your application (recommended), you can specify that the application be started with a fully-qualified path. For example:

```
http://webhost:8008/launch/MyAppNameSpace.WebBrowser
```

Refer to “Using Name Spaces” on page 3-11 for more information.

Inside the VisualWorks Application Server

In order to understand how communication between clients and your application is established, let's look in a bit more detail at the mechanisms used by the Application Server to resolve client requests.

Regardless of whether you are using the native Smalltalk HTTP Server or an External Web Server, the same strategy is used to resolve the URL and respond to web requests. At a high level, the process is simple: the server passes the web request to a WebRequestService. The request is put in a ProcessEnvironment so that it can be identified as a web interaction (as opposed to a screen interaction). The WebRequestService handles the request, invoking a resolver object which in turn uses the information in the URL to determine whether to respond to the request itself or forward the web request to another kind of resolver or responder. The forwarding continues until an object is found that can answer the web request.

A variety of objects participate in the process, including:

- WebRequests
- WebAnswers
- WebResponders, including:
    - FileResponders
    - RequestEchoers
    - RedirectionAnswers
- WebResolvers, including:
    - PathInfoPrefixResolvers
    - AnySessionLaunchers
    - RegisteredSessionLaunchers
    - SessionResolvers
    - ConstantResolvers

WebRequests

A web request is an object that encapsulates the information relayed into the server environment. Web requests are instances of class WebRequest and are created by reading formatted CGI or ISAPI information over a stream or socket, or by being handed an HTTP request, including information about the originating HTTP server and communications stream.

A web request is also capable of generating an answer. When passed a web answer object via the answerWith: method, a web request writes that answer, with appropriate headers, to its responseStream and then closes that stream.

WebAnswers

A web answer is an object that represents the reply to a web request. A web answer is usually an instance of class WebAnswer or one of its subclasses. A web answer can write itself out to a stream in the fashion required by the CGI interface between HTTP servers and external programs.

Numerous objects in the system can be coerced into becoming a web answer object by being sent the message asWebAnswer.

WebResponders

A web responder is any object that can take a web request and offer a meaningful reply. Class WebResponder is in fact an abstract superclass that defines the behavior and protocol for concrete objects that can respond to WebRequest objects. Web answers are web responders because they are themselves meaningful replies to a web request (WebAnswer is actually a subclass of WebResponder.) Web resolvers are also web responders because, after they find the object that can respond to a web request, they can ask that object to respond.

Web responders are usually, but do not need to be, instances of a subclass of the abstract superclass WebResponder. Web responders must implement an answerFor: method that accepts a web request and returns an object which is suitable as a web answer.

Three web responders are of particular interest:

- FileResponder
- RequestEchoer
- RedirectionAnswer

FileResponder

A FileResponder returns a file external to the Application Server's environment. Many common MIME file types are supported, including graphics files, audio files, JavaScript, and others, as defined in FileResponder class>>initializeDefaultTypes.

RequestEchoer

A RequestEchoer simply returns the web request. More specifically, it returns an HTML document that describes the web request received and other environment information.

The default set of resolvers includes the URL path **echo** and maps it to a RequestEchoer. The **echo** path is useful for testing the server setup.

RedirectionAnswer

A RedirectionAnswer returns an HTTP header that causes the web browser to fetch an alternate URL. In case the browser does not properly support this header, a bit of explanatory text and an explicit link to the alternate URL can also be sent.

WebResolvers

A web resolver finds an object that can respond to a web request. That object may be another web resolver. As mentioned above, a web resolver is a kind of WebResponder (in fact, a subclass). A web resolver object is primarily responsible for launching a new instance or session of your application.

Web resolvers are instances of a subclass of the WebResolver class. Web resolvers must implement a resolve: method and return an object that can serve as a web responder.

There are five primary types of web resolvers:

- PathInfoPrefixResolver
- AnySessionLauncher
- RegisteredSessionLauncher
- SessionResolver
- ConstantResolver

PathInfoPrefix Resolver

When the server first receives a web request, it forwards the request to its service, which hands the request to a PathInfoPrefixResolver. The PathInfoPrefixResolver's sole responsibility is to forward the web request to another object. A PathInfoPrefixResolver can forward the web request to:

- A web responder:
  - RequestEchoer
  - RedirectionAnswer
  - FileResponder
- Another web resolver:
  - AnySessionLauncher
  - RegisteredSessionLauncher
  - SessionResolver

To determine the responder or resolver to which to forward the request, the PathInfoPrefixResolver looks at the path information from the URL. In particular, it looks at the first segment after the hostname and port and (optional) CGI relay location. For example, in the following sample URLs, the PathInfoPrefixResolver looks at the keyword **launch**:

```
http://webhost:8008/launch/WebBrowser
http://www.asite.com/cgi-bin/scriptName/launch/WebBrowser
http://www.asite.com/scriptName.cgi/launch/WebBrowser
```

The PathInfoPrefixResolver has a *registry* that maps path keywords to other types of resolvers and responders.

When you create a server, you can specify that you want its PathInfoPrefixResolver to have an initial registry that includes default resolvers. (See "Creating a Server" on page 2.) The default registry includes four path keywords:

launch
: Forwards the request to an AnySessionLauncher in the Application Server (developer environment) or to a RegisteredSessionLauncher in the server environment. The **launch** keyword is used to start a new instance of an application.

submit
: Forwards the request to a SessionResolver. The **submit** keyword is used to connect with an already-running instance of an application.

echo
Forwards the request to a RequestEchoer, which returns the web request.

JavaSTGM.class
Forwards the request to a JavaSTGMRenderer to render Smalltalk graphics as Java graphics.

You can also create and edit the registry manually using a Smalltalk inspector. When you create your own path, you can map it to any of the five types of responders and resolvers listed above. When you edit an existing path, you can change the name of the path, but you cannot change the type of responder or resolver.

Launching Resolvers

Launching resolvers start (or *launch*) new instances of applications in order to find the object to respond to a web request. They launch the applications within SessionResolvers, which manage the application until it terminates.

To determine the application to launch, a launching resolver looks at the path information from the URL. In particular, it looks at the *second* segment after the hostname and port and (optional) CGI relay location. For example, in the following URLs, the launching resolver looks at the keyword **MyApp**:

```
http://webhost:8008/launch/MyApp
http://www.asite.com/cgi-bin/scriptName/launch/MyApp
http://www.asite.com/scriptName.cgi/launch/MyApp
```

Note that the **launch** segment is not used by the launching resolver; it is used by the PathInfoPrefixResolver to cause it to forward the request to the launching resolver.

The launching resolver has a registry that maps path keywords to the names of ApplicationModel classes. There are two types of launching resolvers:

- AnySessionLauncher
- RegisteredSessionLauncher

These two launching resolvers differ with respect to the restrictions imposed upon what they allow web clients to launch:

AnySessionLauncher
An AnySessionLauncher uses all subclasses of ApplicationModel as its registry of available applications, with the class names as the keys. There are *no restrictions* on the application models that can be

opened; any application model class in the server environment can be opened. For that reason, AnySessionLaunchers are most often used in situations in which security and environment integrity is not at risk.

In the development environment, the default **launch** path prefix is mapped to an AnySessionLauncher.

RegisteredSessionLauncher

A RegisteredSessionLauncher starts only those applications that have been explicitly registered for availability. Availability is limited to the RegisteredSessionLauncher in whose registry there is a path for the application.

In the registry, path keywords are mapped to the names of subclasses of ApplicationModel. The registry may also include a modifier:

- For application models, this modifier is by default a window specification. You can open the same class with different specifications based on a different path keyword.
- Optionally, the modifier can be a message that is sent when the path is requested. The modifier is an instance of MessageChannel and can take up to one argument. If the specified message selector takes an argument, the web request is passed in for reference. The result of this message send becomes an argument to the method ApplicationModel>>open:withPolicy:inSession:. You should override this method to handle your argument appropriately.

In the server environment, the default **launch** path is mapped to a RegisteredSessionLauncher. Initially, the registry for the **launch** RegisteredSessionLauncher contains no paths. You can create a registry from scratch, load one from a parcel, or load one from a file.

Session Resolvers

SessionResolvers provide a place to keep launched web sessions, where a *web session* can be thought of as a manager for a running instance of an application. A web session is created when an application is launched from a web browser.

SessionResolvers keep track of web sessions and the timeout policy for expiring inactive sessions. Each SessionResolver has a registry that maps session key numbers to web sessions. The session key uniquely identifies the web session and can be found in the form data of the web request.

The default set of resolvers includes the path **submit** and maps it to a SessionResolver. The default resolvers also include a launching resolver (with the path **launch**) that launches its applications into the **submit** SessionResolver.

Constant Resolvers

Constant resolvers are not available as a user selectable resolver. The only constant resolver present in the VisualWorks Application Server is the **JavaSTGM.class** resolver. Additional constant resolvers can be implemented as instances of ConstantResolver.

5

Using Front-End HTTP Servers

When a Web application is deployed, the VisualWorks Application Server is usually configured as a separate process that runs behind a front-end HTTP server. This arrangement can provide superior performance over the Smalltalk HTTP Server that is used during development. For maximum reliability and security, the front-end server and VisualWorks may also be hosted on two separate machines.

Requests from the front-end server are passed to the VisualWorks environment using either a *proxy* or a special *gateway application*. This chapter discusses the use of both, and explores some of the performance tradeoffs.

VisualWorks Application Server supports several different gateway applications: CGI, Perl, and ISAPI. A CGI relay can be used with almost any HTTP server, though it is the least efficient gateway because it needs to fork a process for each Web request. When using the Apache web server, you can also configure the server without a gateway application.

This chapter covers:

- Configuring VisualWorks with a Front-End Server
- Configuring VisualWorks to use an Apache Proxy
- Configuring a CGI Relay
- Configuring a Perl Relay
- Configuring an ISAPI Relay
- Defining an Alias using a Hostmap file
- Specifying a Gateway Error Message

Configuring VisualWorks with a Front-End Server

When running the VisualWorks Application Server with a front-end server such as Apache or IIS, a number of configuration options are available.

In general, Apache is much simpler to configure and maintain, has fewer limitations, and is thus the recommended platform for VisualWorks server applications. When using Apache, a proxy may be used to complete the installation by simply modifying a single configuration file. For details, see: "Configuring VisualWorks to use an Apache Proxy" on page 5-2.

Generally, a gateway application is used. Version 7 of the VisualWorks Application Server supports the following gateway configurations:

Gateway	Server	Supported Operating Systems
CGI	Various	Linux, Sun, HP, AIX, Windows
Perl	Various	Linux, Sun, HP, AIX, Windows
ISAPI	IIS 4 & 5	Windows 98, NT 4.0 (with Service Pack 6), ME (with Personal Web Server), 2000, and XP.

The following pages provide detailed instructions for installation and configuration of each type of relay.

Three basic steps are required to make the Application Server communicate with a front-end HTTP server:

1. Create an External Web server within the Application Server environment (for step-by-step details, see "Servers" on page 2-1).
2. Install the relay, creating an alias and then editing the relay configuration file.
3. Configure the external server to invoke the appropriate relay.

Configuring VisualWorks to use an Apache Proxy

Using a proxy with Apache is one of the simplest ways to install and configure the VisualWorks Application Server. No gateway is required. You simply edit the Apache configuration file and restart the server.

For example, using Apache 2.0 on Redhat Fedora 4:

1 Install VisualWorks, e.g. in **`/usr/local/vw7.5`**.

2 Set up the execution path and environment variables for VisualWorks.

a Edit **.bash_profile** (located in your login directory), to include the following:

```
PATH=$PATH:$HOME/bin:/usr/local/vw7.5/bin/linux86
VISUALWORKS=/usr/local/vw7.5
export PATH VISUALWORKS
```

b To put these changes into effect, return to the shell and execute:

```
source .bash_profile
```

c To check that the variable has been set, try:

```
echo $VISUALWORKS
```

3 Login as **root**, and start the Apache server:

```
/etc/init.d/httpd start
```

Before proceeding, you might want to confirm that Apache is active at the desired public URL.

4 To set up the proxy, add the following to the end of **/etc/httpd/conf/httpd.conf**:

```
NameVirtualHost 10.9.8.7:80
<VirtualHost *>
</VirtualHost>

<VirtualHost 10.9.8.7:80>
    ServerName www.myDomain.com
    ServerAlias myDomain.com *.myDomain.com
    ProxyPass / http://www.myDomain.com:8008/
    ProxyPassReverse / http://www.myDomain.com:8008/
</VirtualHost>
```

Note: in the NameVirtualHost and VirtualHost directives, use the IP address of your server in place of 10.9.8.7.

5 In the shell, restart the HTTP server to make these changes take effect:

```
/etc/init.d/httpd restart
```

6 Connect to the VisualWorks /web directory (**/usr/local/vw7.5/web**) and start the pre-packaged runtime image as follows:

```
./runtime.im &
```

7 Check that you can see the VWAS Welcome Page using a browser:

http://www.myDomain.com/configure

At this point, the configuration is complete.

If you wish additional security, or to simplify the setup of a firewall, it is also possible to configure the VisualWorks Application Server to listen only on the loopback interface 127.0.0.1.

> **Note:** the pre-packaged server runtime.im is not configured to listen on the loopback address 127.0.0.1, and cannot be used for the following configuration.

To configure Apache to communicate via the loopback address, perform the first four steps outlined above, and then continue as follows:

1 Modify the VirtualHost directive in **`/etc/httpd/conf/httpd.conf`** like this:

```
<VirtualHost 10.9.8.7:80>
    ServerName www.myDomain.com
    ServerAlias myDomain.com *.myDomain.com
    ProxyPass / http://127.0.0.1:8008/
    ProxyPassReverse / http://127.0.0.1:8008/
</VirtualHost>
```

For this configuration, only the ProxyPass and ProxyPassReverse directives are changed. In this way, the Apache server faces the world, and the VisualWorks server is in a private space.

Again, in the VirtualHost directive, use the IP address of your server machine in place of 10.9.8.7.

2 In the shell, restart the HTTP server to make these changes take effect:

```
/etc/init.d/httpd restart
```

3 To configure the Application Server to listen on the loopback address, open the Server Console and create a new server (for details, see "Creating a Server" on page 2-2), using **127.0.0.1** as the **Hostname** and **8008** as the **port number**.

4 Untick the **Bind to all interfaces** checkbox in the Server Console.

5 Click the **Create** or **Create and Start** button.

6 Once the Application Server is running (you may need to create a deployment image; for details, refer to "Making a Deployment Image" on page 3-17), check that you can see the VWAS Welcome Page from a browser:

http://www.myDomain.com/configure

Configuring a CGI Relay

Platform-specific CGI relays are provided on the VisualWorks distribution medium in the directory named **/waveserver/waverelays/cgi2vw**.

Installing a Relay

HTTP servers generally expect to be able to execute a gateway application as a stand-alone process. Because it is not efficient to start and stop a VisualWorks image for each request, the Application Server instead uses a small application to collect the request information, relay it to a server in the VisualWorks environment, and then return the result as a web page to the HTTP server.

Each supported platform requires a specially-targeted gateway application. These platform-specific files are provided on the VisualWorks distribution medium in the directory named **/waveserver/waverelays**.

An initialization file for each gateway can be found in the directory **/waveserver/waverelays/VisualWave**.

Preparing the CGI Relay

When the CGI relay is invoked, it contacts a server process that is specified by a hostname and port number. Because the only information that can be passed easily to a CGI program is the name by which it is invoked, VisualWorks uses the name of the relay application to convey the hostname and port number information.

VisualWorks Application Server supports two conventions for naming the CGI relay:

- You can name the CGI relay application to be the Application Server's hostname and port number.
- You can use an alias for the CGI relay and map that alias to the Application Server's hostname and port number in a **hostmap** file.

For new installations, the former strategy is recommended. Support for the **hostmap** file is provided mainly for backward compatibility. For details, see "Defining an Alias using a Hostmap file" on page 5-29.

Using the Hostname and Port Number

The simplest way to set up the CGI relay to connect to a specific server is to make a copy of the CGI relay file, and name it as follows:

```
hostname@portNumber
```

For example, assume that you have created an External Web Server named “tyler” on port 2223 in the VisualWorks environment. To redirect requests to this server, make a copy of the CGI named **tyler@2223** on UNIX systems, or **tyler@2223.exe** on Windows systems.

Configuring the CGI Initialization File

Each CGI gateway reads a set of configuration parameters from a special initialization file. The gateway expects to find the INI file in a directory called **VisualWave** inside a directory specified using the environment variable **SystemRoot**.

To set the location of the INI file:

- For MS-Windows, the **SystemRoot** variable is set by default to the Windows install directory; e.g. for Windows NT this would be **C:\WINNT\VisualWave**.
- Under UNIX, the **SystemRoot** variable is not normally defined. If you define **SystemRoot=/etc** then the gateway will expect to find the INI file in **/etc/VisualWave**. If the variable **SystemRoot** is not defined, the gateway starts looking from the root directory, e.g., the INI file should be in **/VisualWave**.

To configure the INI file:

1 Copy the INI file **/waverelays/VisualWave/cgi2vw.ini** to the appropriate directory (e.g. **/etc/VisualWave**).

2 Under UNIX, ensure that the ownership of the INI file is correct:

```
chown root.apache cgi2vw.ini
```

3 Modify the **[localhost]** section of the INI file:

   a Modify the name of the section, changing **localhost** to a name corresponding to the gateway you're using (e.g. **cgi2vw**).

   b Specify the appropriate **host** and **port** definitions.

   c Set **logFileName** to the fully-qualified log file name.

4 Save your modifications to the INI file.

5 Under UNIX, ensure that permissions on the directory containing the INI file allow reading by the group that is running the relay:

```
chmod +r /etc/VisualWave
```

Testing the CGI Relay

To test your relay, invoke it with the **-d** command line flag:

```
tyler@2223 -d

rose -d
```

If the relay is able to connect to the IPWebListener server, the server returns a set of environment variables. If it does not connect, an error message is displayed.

Common errors include failure to resolve the hostname, which could indicate that you need to specify a fully qualified name, and "connection refused," which probably indicates a server configuration error.

Note: If the relay cannot locate the INI file, it exits with a segmentation fault. If you observe this behavior, check to make sure that the **`SystemRoot`** environment variable is specifed correctly. This behavior will be ameliorated in a future release of VisualWorks.

Preparing the HTTP Server

You may need to configure the HTTP server to run the CGI relay. Typically, commercial HTTP servers support two methods for this:

- **Directory:** A specific directory (usually **`/cgi-bin`** or, on IIS, **`inetpub\scripts`**) is designated as one that contains gateway applications. Any file found in this directory is assumed to be a CGI program or script.
- **Filename:** A specific filename extension (usually **`.cgi)`** is designated as identifying CGI applications, anywhere in the directory hierarchy.

The documentation for your HTTP server provides instructions for setting these options.

Using a cgi-bin Directory

If you've configured a CGI directory for your HTTP server, move the renamed copy of the CGI relay script to that directory.

Now test the configuration. For example, suppose that:

- the HTTP server is running on the default port on a machine named "victoria"
- the CGI directory is **`/cgi-bin`**, and
- the CGI relay script name is **`rose`** which the **`hostmap`** file maps to host machine "tyler" and port number 2223

In your web browser, enter the following URL:

http://victoria/cgi-bin/rose/echo

The Application Server should return an echo of the environment variables in the Web request.

Using a .CGI Extension

Certain HTTP servers allow you to associate a file type with a particular gateway, e.g., on IIS, you may specify a set of mappings to gateways for a given directory. Thus, for example, the site http://victoria/example.ssp could be associated with VisualWorks.

If you've configured your HTTP server to use a specific extension (by convention, **`.cgi`**) for CGI programs, then rename the CGI relay file to have that extension. Formerly, UNIX servers required use of the **`.cgi`** extension, though this is now only a convention.

Note: On Windows, the extension must be **.exe**.

You may test an installed CGI by echoing the environment variables sent from the Web browser.

For example, suppose that:

- the HTTP server is running on the default port on a machine named "victoria"
- the script is in the directory **demo** in the server root directory, and
- the CGI relay script name is either **rose.cgi** or **tyler@2223.cgi**, depending on the naming convention

Enter the appropriate URL in your browser:

http://victoria/demo/tyler@2223.cgi/echo
http://victoria/demo/rose.cgi/echo

The Application Server should return an echo of the environment variables in the Web request.

Configuring a CGI for Authorization

When using a front-end server, you must configure the CGI to pass authorization tokens. For Apache servers, see "Configuring Apache to pass HTTP Authentication Requests" on page 5-12.

To make a CGI work with authorization under MS-Windows, you must disable integrated Windows authentication, and also change the properties of the IIS **scripts** directory (for details on the former, see "Configuring Password Authentication Under Windows" on page 5-25).

1 Using the IIS Manager tool, open the **Properties** tab for the **scripts** directory.

2 Click on the **Custom Errors** tab, pick **401** and change it from a filename to **Default**.

The CGI gateway should now work with authentication.

Sample Installation: Linux/Apache

To install the gateway for use with Apache 1.3 running under RedHat 7.1:

1 Login as **root**.

2 Copy the Linux CGI gateway (**/waveserver/waverelays/cgi/linux86/cgi2vw**) to the **/var/www/cgi-bin** directory.

3 Rename the gateway to be **myhostname@2223**.

4 Change the permissions of the gateway to be executable by the **apache** group (the group that Apache is running as) using:

```
chown root.apache myhostname@2223
chmod g+x myhostname@2223
```

5 Create the directory **/etc/VisualWave** and copy the default file **/waveserver/waverelays/VisualWave/cgi2vw.ini** there.

6 Ensure that the ownership of the INI file is correct:

```
chown root.apache cgi2vw.ini
```

7 Make both the directory and the file readable by the **apache** group.

```
chown -R root.apache /etc/VisualWave
chmod -R g+r /etc/VisualWave
```

8 Open **/etc/httpd/conf/httpd.conf** using a text editor.

9 Add the following to **httpd.conf**:

```
SetEnv SystemRoot /etc
```

This allows the gateway to find the INI file.

10 Save the change to **httpd.conf**.

11 Start the HTTP server with:

```
/etc/init.d/httpd start
```

Configuring a Perl Relay

The perl gateway can be run either as a separate process that relays CGI information from the front-end server to VisualWorks or, when using Apache servers, the script can be run directly inside the Apache process space using **mod_perl**.

When run as a CGI script, the perl gateway is likely to be slower, but it is easier to configure and debug. When run using **mod_perl**, it may actually be significantly faster than the compiled CGI, depending upon the server configuration and load.

The perl gateway differs from the others in that it reads neither the .INI file for configuration or the **hostmap** file. Since the script is directly editable, all relevant parameters are embedded directly.

The gateway is provided on the VisualWorks distribution medium in the directory named **/waveserver/waverelays/perl**.

Using the Perl Gateway as a CGI with Apache

To use the perl gateway, first copy the script to the /cgi-bin directory (**/waveserver/waverelays/perl/visualworks.pl**) to the **/var/www/cgi-bin** directory.

When using the perl gateway to invoke your web application, there are several ways to present the application in the URL.

The easiest way is to refer directly to the script in the URL, e.g:

http://host/cgi-bin/visualworks.pl/your/path/to/file.ssp

This URL would serve the server page indicated by the path your/path/to/file.ssp.

Alternately, you can use an alias to map all requests to a particular URL to the gateway file. For example, if you used the location myApp, then any URL of the form:

http://host/smalltalk/your/path/to/file.ssp

would serve the server page indicated by the path your/path/to/file.ssp.

To configure this alias, place the script file **visualworks.pl** in the directory **/var/www/cgi-bin/visualworks.pl**, and use the ScriptAlias directive in **httpd.conf** to indicate that it is a script to be executed:

ScriptAlias /smalltalk /var/www/cgi-bin/visualworks.pl

Configuring Apache to pass HTTP Authentication Requests

By default, Apache **mod_cgi** does not pass HTTP Authentication headers to CGI scripts. To enable this behavior, we need to use **mod_rewrite** to manually copy these headers into the request's environment so they are passed to the scripts.

There are two different ways to achieve this: using the file **.htaccess**, or by using the server's **httpd.conf**.

Passing Authentication Requests Using .htaccess

Create a file named **.htaccess** in your **/cgi-bin** directory which contains the CGI script (in this example **visualworks.pl**). The file **.htaccess** should contain:

```
Options +FollowSymLinks
<File visualworks.pl>
    RewriteEngine on
    RewriteCond %{HTTP:Authorization} ^(.*)
    RewriteRule .* - [E=HTTP_AUTHORIZATION:%1]
</File>
```

Passing Authentication Requests Using the Server's httpd.conf

By editing the file **httpd.conf** (or **Vhosts.conf**), there are several different ways to configure an Apache server to pass authentication requests.

First, you may edit the <Directory ..> section for the directory that contains **visualworks.pl** and insert the same directives into that section:

```
<Directory /myAppDirectory>
    Options +FollowSymLinks
    RewriteEngine on
    RewriteCond %{HTTP:Authorization} ^(.*)
    RewriteRule .* - [E=HTTP_AUTHORIZATION:%1]
</Directory>
```

Alternately, you may edit the <VirtualHost ..> section of the config file:

```
<VirtualHost 192.168.x.y>
  ServerName myServer.com
  ServerAlias myServer.net
  CustomLog   /var/www/myServer/log/access_log combined
  ErrorLog    /var/www/myServer/log/error_log
  RewriteEngine on
  ScriptAlias / /var/www/myServer/cgi-bin/visualworks.pl/
  RewriteCond %{HTTP:Authorization} ^(.*)
  RewriteRule .* - [E=HTTP_AUTHORIZATION:%1]
</VirtualHost>
```

You may also use a <Directory ..> section within the <VirtualHost ..> section of the config file:

```
<VirtualHost 192.168.x.y>
    ServerName www.myServer.com
    DocumentRoot /var/www/myServer/html
    CustomLog   /var/www/myServer/log/access_log combined
    ErrorLog    /var/www/myServer/log/error_log
    ScriptAlias /app /var/www/myServer/cgi-bin/visualworks.pl
    <Directory /var/www/myServer/cgi-bin/>
        Options +FollowSymLinks
        RewriteEngine on
        RewriteCond %{HTTP:Authorization} ^(.*)
        RewriteRule .* - [E=HTTP_AUTHORIZATION:%1]
    </Directory>
</VirtualHost>
```

To use the <Location ..> directive to redirect requests to a specific URL (e.g., http://www.myServer.com /mySite):

```
<VirtualHost 64.208.230.40>
    ServerName www.myServer.com
    DocumentRoot /var/www/myServer/html
    CustomLog   /var/www/myServer/log/access_log combined
    ErrorLog    /var/www/myServer/log/error_log
    Alias /cgi-bin/ /var/www/myServer/cgi-bin/
    <Directory /var/www/myServer/cgi-bin/>
        Options +ExecCGI +FollowSymLinks
        SetHandler cgi-script
        RewriteEngine on
        RewriteCond %{HTTP:Authorization} ^(.*)
        RewriteRule .* - [E=HTTP_AUTHORIZATION:%1]
    </Directory>
    <Location /mySite>
        Action cgi2vw-script /cgi-bin/visualworks.pl
        SetHandler cgi2vw-script
    </Location>
</VirtualHost>
```

Using the Perl Gateway with mod_perl

There are many different ways to configure the Apache to run a perl gateway with **`mod_perl`**. The following discussion shows a basic configuration.

Many installations use two daemons, **`httpd`** and **`httpd-perl`**. The latter is basically the **`httpd`** binary, statically linked with **`libperl.so`**. The majority of package-based installations are of this type.

Assuming that the Apache server's data files are located in **/var/www**, there is typically a directory **/var/www/perl** containing perl scripts. In the server config file (**httpd.conf**), all requests for either http://server/perl/script.pl or http://server/cgi-perl/script.pl are passed to the **httpd-perl** daemon (typically running on port 8200). The configuration file for this daemon (**httpd-perl.conf**) defines aliases for the **/perl** and **/cgi-perl** locations, both pointing to **/var/www/perl**.

By default, the configuration does not invoke scripts using path arguments (e.g., http://server/perl/script.pl/arg/subarg), which is a requirement for the **visualworks.pl** script.

To use the **visualworks.pl** gateway, change the default, assuming that the perl gateway is located in **/var/www/perl/visualworks.pl**, and that the desired query URL is: http://server/myscript/info.ssp, place the following configuration directives in the **Vhosts.conf** file:

```
#Enable mod_rewrite
RewriteEngine on

#For queries sent to the httpd server, pass to httpd-perl server
<IfModule !mod_perl.c>
    RewriteRule ^(/myscript.*)$ http://%{HTTP_HOST}:8200$1 [P]
</IfModule>

#Configuration used by httpd-perl server
<IfModule mod_perl.c>
    # Configure /myscript to represent the Perl Gateway
    Alias /myscript /var/www/perl/visualworks.pl
    # mod_cgi will not pass the HTTP Authorization header
    # in the environment.  This can be fixed by tailoring
    # the Perl Gateway to use more Apache::Registry behavior.
    # As a quick fix, one that can be used for all CGI scripts,
    # we do the following:
    <Location /myscript>
        RewriteCond %{HTTP:Authorization} ^(.*)
        RewriteRule .* - [E=HTTP_AUTHORIZATION:%1]
    </Location>

    # Finally, configure the Perl Gateway to run under
    # Apache::Registry
    <Files visualworks.pl>
        SetHandler perl-script
        PerlHandler Apache::Registry
        PerlSendHeader On
    </Files>
</IfModule>
```

This configuration also works on a server that dynamically loads **mod_perl**. In this case, the <IfModule !mod_perl.c>...</IfModule> section is not used and may be deleted from the configuration file.

For details on configuring **mod_perl**, see the Apache documentation:

http://perl.apache.org/docs.

Using the Perl Gateway with IIS

When using IIS, you can configure the server to accept a direct URL as with a UNIX server, but this requires a virtual directory. For details on creating virtual directories under IIS, see "Creating IIS Virtual Directories" on page 5-20.

Briefly, to configure the IIS Server to map all files in a virtual directory:

1. Open the IIS configuration tool, and get the properties for the desired directory (creating it first, if necessary). For example, we might use a directory named smalltalk.
2. From the **Virtual Directory** tab, press the **Configuration** button. A dialog appears that includes **Application Mappings**.
3. In the **Application Mappings** dialog, map the extension "*" to, e.g.

   c:\perl\bin\perl.exe c:\inetpub\scripts\visualworks.pl

   Alternately, to map only certain files in the directory, map only those extensions required (e.g. **.ssp**, for Smalltalk Server Pages).

Next, the VisualWorks Application Server must be configured to receive the requests from IIS:

1. Open the Application Server Console (select **Server Console** from the **Web** menu from the Launcher window).
2. In the Server Console, select the target WaveIPRequestBroker, and select **Edit Server** (for details on creating an WaveIPRequestBroker, see "Creating a Server" on page 2-2).
3. Enter the name of virtual directory defined in IIS (e.g. "smalltalk") in the **Virtual Directories** input field.

This configuration allows a URL of the form:

http://host/smalltalk/your/path/to/file.ssp

to invoke the server page indicated by the path your/path/to/file.ssp.

Compiling the VisualWorks Perl Script

The perl script may be compiled to an executable, using the Perl-to-C compiler. This would result in some speed improvement, and also make it possible to use the gateway as a direct URL within IIS. However, this option would not allow the use of **`mod_perl`** under Apache, and so would likely have slower performance on Apache servers.

Known Limitations in IIS

At this time it is unclear whether IIS supports HTTP POST requests through the CGI if the 'Transfer-Encoding' HTTP header of the request is set to 'chunked'. The current Perl gateway script can not determine when it has read the complete POST request body.

Configuring an ISAPI Relay

The ISAPI relay is an extension designed for use with Microsoft IIS 4.0 and later. Installing the relay involves making sure it is located on the correct directory path and configuring the Application Server for its use.

The ISAPI relay enables the front-end HTTP server to pass incoming service requests directly to a special DLL, instead of invoking the gateway as a separate process (i.e., as a CGI). By running the gateway in the same thread of execution as the HTTP server, the overhead of process-switching during request servicing is minimized.

Selecting an ISAPI Library

Two DLLs for Windows are provided with the VisualWorks distribution:

```
/waveserver/waverelays/isapi/nt/isapi2vw.dll
/waveserver/waverelays/isapi/nt/isapi2vw-debug.dll
```

In a production environment, always use the **`isapi2vw.dll`** relay.

The debug version should only be used for testing an IIS installation. For details, see “Using the ISAPI Debugging Relay” on page 5-26.

Preparing the ISAPI Library

The ISAPI relay, when invoked, determines which VisualWorks server to contact. This is specified via the server’s hostname and port number. To configure the relay for passing requests to the server, the relay file must be named appropriately.

The ISAPI relay does not use an INI file. The principal configuration element is the name of the DLL file. When used with IIS, the VisualWorks Application Server allows three different conventions for naming the relay:

1. ISAPI file named with the server’s hostname and port number.
2. Using an alias for the ISAPI relay to map the Application Server’s hostname and port number in a **`hostmap`** file.
3. Using a virtual directory in IIS.

Generally, the first option is the simplest, but it carries some security risks and forces the name of the DLL to appear in the URL. In a production environment, the second or third options are recommended. For greatest control over the appearance of the URL presented to the client, the third option (using a virtual directory) is best.

Details on these three approaches to configuration are explained below.

Using the Hostname and Port Number

The simplest way to set up the ISAPI relay to connect to a specific server is to make a copy of the ISAPI DLL, and name it as follows:

```
hostname@portNumber.dll
```

For example, assume that you have configured a WaveIPRequestBroker server named “tyler” on port 2223 in the VisualWorks environment. To direct requests to this server, make a copy of the ISAPI DLL provided with the VisualWorks standard distribution, and name it **`tyler@2223.dll`**.

In this configuration, a URL to reach the WaveIPRequestBroker would be:

http://www.mydomain.com/scripts/tyler@2223.dll/MyApplication

To exercise more control over the appearance of the URL, you may alternately configure the DLL using an alias or a virtual directory.

Using an Alias

To use an alias for an External Web Server, you must create a **`hostmap`** file.

With an alias defined in the **`hostmap`** file, make a copy of the ISAPI DLL, and rename it to the alias. For example, if you have defined an alias rose, name the copy of the DLL **`rose.dll`**.

In this configuration, a URL to reach the WaveIPRequestBroker would be:

http://www.mydomain.com/scripts/rose.dll/MyApplication

By using an entry in the **`hostmap`** file, you can keep the hostname and port out of the URL space. To exercise maximum control over the URL space, you must use an IIS virtual directory.

For details, see “Defining an Alias using a Hostmap file” on page 5-29.

Using a Virtual Directory

To map requests to the WaveIPRequestBroker using a virtual directory, you must create and configure a virtual directory for the front-end HTTP server (in this case, IIS).

To prepare the ISAPI DLL for use with an IIS virtual directory, you have the option to use a **`hostmap`** file, or not. The simplest way to set up the ISAPI relay for a virtual directory is to make a copy of the DLL file, and name it as follows:

```
hostname@portNumber.dll
```

The DLL can be mapped directly via the virtual directory's application mapping properties. Alternately, you can use an alias defined in a **hostmap** file, in which case the DLL file does not need to be renamed.

Using a virtual directory, it is possible to conceal the name of the DLL or hostmap alias from clients — only the name of the virtual directory need appear in the URL.

For example, using a virtual directory named flower, a URL to reach the WaveIPRequestBroker would be:

http://www.mydomain.com/flower/MyApplication

For details, see "Creating IIS Virtual Directories" on page 5-20.

Installing the ISAPI Library

Once the DLL has been given the appropriate name, it must be made available to the IIS installation:

1 Copy the file **isapi2vw.dll** to the appropriate directory.

By default, this is generally **C:\Inetpub\Scripts**, though any directory available to IIS may be used. As a rule, it should be on the same physical drive as the **%SystemRoot%** where IIS is installed.

By default, the environment variable **SystemRoot** is defined as the Windows install directory (e.g., **C:\WinNT**).

2 Set the directory containing **isapi2vw.dll** to allow full execute permission. Note that an IIS virtual directory which maps to the gateway does not require any special permissions.

For basic operation, these steps are sufficient to install the ISAPI library.

IIS provides a number of options that control the operation of the ISAPI gateway (e.g., the gateway can be isolated from the server process, its execution priority can be changed, etc.).

To confirm that the basic installation is correct, you may perform a simple echo test with the ISAPI relay. For details, see "Testing the ISAPI Relay in Stand-Alone Mode" on page 5-26.

Creating IIS Virtual Directories

In IIS, you may use a *virtual directory* to map the URL space to a specific physical path on the server machine. A virtual directory associates an IIS *alias* with the physical path. Note that this alias is distinct from that used in the **`hostmap`** file (described previously in this chapter).

The VisualWorks Application Server may receive requests that IIS has mapped using a virtual directory. In this case, IIS maps the alias of the virtual directory as it appears in the URL to the ISAPI DLL located in a physical directory, preserving the remainder of the URL path, and any query parameters attached to the URL. When a server is configured in this way, clients will only see the alias of the virtual directory.

To use an IIS virtual directory with the VisualWorks Application Server, you must first create and configure it using the IIS tools, and then declare the virtual directory to VisualWorks using the Server Console.

Creating and Configuring a Virtual Directory in IIS

To create a virtual directory, you must use the **Internet Services Manager**. Administrative access to the server is required.

1 In IIS 4.0, the Internet Services Manager may be opened via **Start → Programs → Windows NT 4.0 Option Pack → Microsoft Internet Information Server → Internet Service Manager**. Note: do not use **Internet Service Manager (HTML)**.

In IIS 5.0, the Internet Services Manager may be opened via **Start → Programs → Administrative Tools → Internet Service Manager**.

Under Windows XP, the IIS Management Console may be opened via **Start → Settings → Administrative Tools → Internet Infrmation Services.**

2 Select **New** from the menu bar to open the **New Virtual Directory Wizard**.

When using the IIS Management Console under Windows XP, open the collapsed tree in the left-hand view of the console window, select Default Web Site, and the right-click to select **New → Virtual Directory…**

The wizard enables you to define an alias, a phsyical directory, and provides various options for the **Default Web Site**.

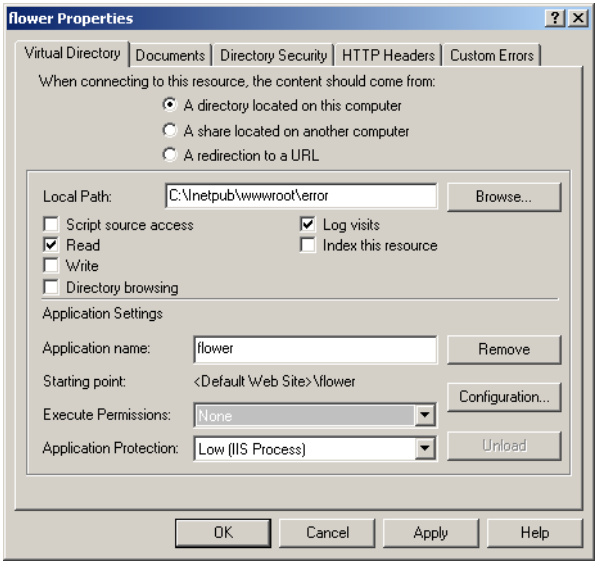
3 For the **Default Web Site**, set the **Alias** to be the name you wish to appear in the URL path.

I.e., for http://www.mydomain.com/flower/MyApplication, enter **flower**.

Set the **Directory**. Note that this is a default, but is optional. You may set it to the IIS error directory, which is usually **C:\Inetpub\wwwroot\error**. This contains a default html file for reporting an error.

4 Set **Access Permissions** to **Read** access.

5 Right click to select the new virtual directory, and open its **Properties**.



6 In the **Application Settings** region of the Properties window, click **Create** to set the application name to your virtual directory name.

7 Set **Execute Permissions** to be **None**.

8 Set **Application Protection** to be **Medium**.

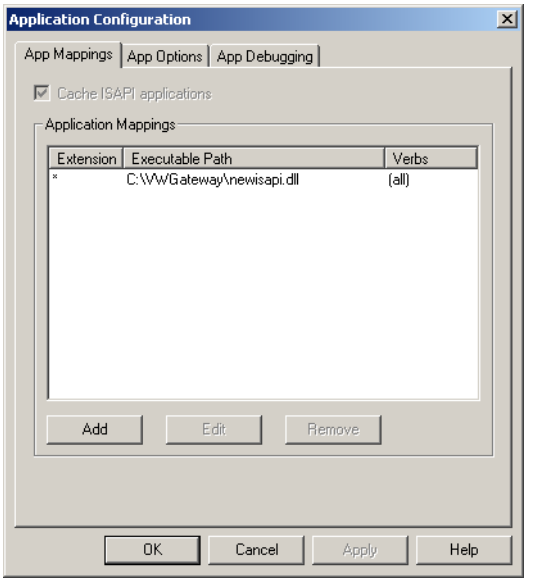
Depending upon your configuration, you may choose **Medium** or **Low**.

When using the debugging DLL for tracing, you must use **Low**. Otherwise, the choice depends upon a safety/performance tradeoff.

Choosing **Low (IIS Process)** directs IIS to run the relay in the same process as Windows Web services. This involves less overhead, but a severe error in your process will halt IIS. Choosing **Medium** directs IIS to run the gateway in an isolated pooled process.

Note: To select **Low (IIS Process)**, you must first create an application by clicking the **Create** button and letting the application take its name from the directory name (see previous step). Unless you create an application for the directory, the protection level can only be **Medium**.

9 Click **Configuration** to edit the application mappings.



10 Select the **App Mappings** tab, and remove all of the default mappings. This step is tedious, but necessary.

11 Add a new mapping, and configure it as:

Extension	*****
Executable Path	**(fully qualified path name of the ISAPI DLL)**
Verbs	**(All)**
Check that file exists	**No**

Note: Note: to enter the ***** for **Extension**, you must type **.*** (period follwed by star). Also, note that in IIS 4.0, **Verbs** is called **Exclusions**.

12 Apply these changes in the Application Configuration window by clicking **OK** to return to the **Properties** window.

13 Apply the changes in the **Properties** window.

Configuration of the virtual directory is now complete.

> **Note:** In order for an IIS virtual directory to become available, it is often necessary to stop and restart IIS. If the ISAPI DLL has already been loaded, IIS must be restarted. If you wish to move or rename the DLL, Internet Services must be restarted to release file locks.

Configuring a Virtual Directory in IIS 4.0

IIS 4.0 (only available in the NT 4.0 Option Pack, separately downloadable from Microsoft) does not create the correct syntax in the MetaBase for the ScriptMaps entry on the virtual directory when you use * as the file extension. It uses .* (as with .asp or .shtml), when it should be using simply *.

As a workaround, use Microsoft's MetaEdit tool (downloadable from their site) to manually change the ScriptMaps entry for your Virtual Directory which points to your ISAPI DLL.

1 After installing the Microsoft tool, select **Start → Programs** and enter **MetaEdit** as the application to run.

2 In the Metabase Editor, select **LM → W3SVC → 1 → Root → <virtual-directory-name>**

3 Click on **ScriptMaps**, and change the entry from (whatever your directory is called)

.*,C:\VWGateway\isapi2vw.dll,1

to

***,C:\VWGateway\isapi2vw.dll,1**

All you are doing is removing the period before the asterisk.

4 Close the editor.

In summary, the following steps have been found to work most reliably with IIS:

1 Open the IIS Manager.

2 Stop Internet Services on the machine.

3 Make the configuration changes (create or modify a virtual directory).

4 Close the IIS Manager.

5 Open the IIS Manager again.

6 Restart Internet Services.

7 Close the IIS Manager.

8 Open the IIS Manager.

9 Inspect your virtual directory to ensure the application mapping is correct.

10 Close the IIS Manager.

Specifying Virtual Directories to a VisualWorks Server

When using virtual directories defined by a front-end HTTP server, it is necessary to declare them using the VisualWorks Server Console.

When creating a server, insert the names of any virtual directories into the **Virtual Directories** input field. The input field can accept multiple names, delimited by semicolons.

If your application starts an WaveIPRequestBroker programmatically, you may alternately add a list of virtual directory names using the accessor method virtualDirectories:.

Configuring Password Authentication Under Windows

To have password checking done by your VisualWorks application rather than using the Windows domain, you must explicitly enable password authentication. By default, IIS 5.0 performs its own authentication, overriding all password settings made using the VisualWorks Application Server administrator's Web interface.

To use the Application Server's authentication mechanism, the Windows Integrated Authentication feature must be disabled. This can be done either on a single ISAPI DLL or on an entire directory.

To enable VisualWorks authentication (under Windows 2000):

1 Open the Computer Management window (from the **Start → Programs → Administrative Tools** menu).

2 Define a user to assign appropriate permissions to the ISAPI DLL. By default, this user will be in the IUSER group.

3 Close the Computer Management window and open the Internet Services Manager.

In IIS 4.0, the Internet Services Manager may be opened via **Start → Programs → Windows NT 4.0 Option Pack → Microsoft Internet Information Server → Internet Service Manager**. Note: do not use **Internet Service Manager (HTML)**.

In IIS 5.0, the Internet Services Manager may be opened via **Start → Programs → Administrative Tools → Internet Service Manager**.

Under Windows XP, the IIS Management Console may be opened via **Start → Settings → Administrative Tools → Internet Infrmation Services.**

4 Using the IIS Manager, locate the virtual directory containing the **`isapi2vw.dll`** file installed in the previous section (e.g. **`isapirel`**), select the directory, and then open its **Properties**.

5 Click on the **Directory Security** tab in the properties window.

6 **Edit** the values for **Anonymous Access and authentication control**.

Note: Basic Authentication passwords are sent as plain text. For this reason it is advisable to ensure that passwords are only sent after an HTTPS connection has been established. For information on setting up an HTTPS secure connection, refer to the IIS documentation.

7 In the **Authentication Methods** dialog, select **Anonymous access**.

8 Deselect **Basic authentication**.

9 Deselect **Integrated Windows authentication**.

10 Close the **Authentication Methods** window by clicking **OK**.

11 Close the directory **Properties** window.

The configuration is now complete.

Testing the ISAPI Relay in Stand-Alone Mode

For testing purposes, the ISAPI debug relay provides an echo feature that may be used without connecting the VisualWorks Application Server. This feature allows you to test for the correct installation of the ISAPI relay, and any virtual directories you may have defined.

The echo test is invoked using the query parameter EchoIsapi2vw.

For example, with IIS running on machine localhost at port 8008, and with a virtual directory called vwtest, you can test for an echo as follows:

http://localhost:8008/vwtest?EchoIsapi2vw

To test a **POST** request, this query parameter can also be added to the action value on a form. When the form is sent via Submit, the ISAPI relay echos the request, including its form data, back to the client.

For an example of the feature, see the file **isapi-test-form.html**, included in the **\waveserver\waverelays\isapi** directory.

Note that the echo test does not verify that the **hostmap** file is set up correctly for the VisualWorks Application Server. However, it does confirm that the relay DLL can access the **hostmap** file, and find its alias there.

Using the ISAPI Debugging Relay

The ISAPI debug extension is a Visual C++/MFC application compiled with the standard debug options, plus an internal VisualWorks option which allows the DLL to record trace entries in a log file.

The DLL keeps a log file that is common to all threads running on the Application Server. Logging is thread safe, with each record containing a unique thread process identifier for the logging activity.

The log file name is created from the unique application name, e.g.:

vw-<appFileName>.log

The log file is written in the directory **%SystemRoot%\VisualWave**, which must exist and have write permission. If any failure occurs trying to create or write the file, the relay functions but no logging is performed.

Note: The ISAPI debug extension should only be used to test installation on an IIS server. Some performance degradation occurs, and the log file grows forever until disk space is exhausted.

When using the debug relay, you must ensure that it is associated with a virtual directory configured with **Application Protection** set to **Low**. For details, see "Creating and Configuring a Virtual Directory in IIS" on page 5-20.

Configuring a NSAPI Relay

Netscape servers support functional extensions through the NSAPI. VisualWorks Application Server supports platform-specific NSAPI relay programs that allow a direct interface to the Netscape server.

The NSAPI relay is used to allow the front-end server to pass incoming service requests directly to a special DLL, instead of invoking the gateway as a separate process. By running the gateway in the same thread of execution as the front-end HTTP server, the overhead of process-switching during request servicing is minimized.

Platform-specific library files are provided on the VisualWorks distribution medium in the directory named **`/waveserver/waverelays/nsapi`**.

Note: Effective in release 7.1 of the VisualWorks Application Server, support for NSAPI relays has been removed from the product.

The existing NSAPI code for HP, MS-Windows, and Sun platforms is still supported through version 7.0, and is henceforth available from Cincom by individual request. For details, please contact Cincom Smalltalk support <supportweb@cincom.com>.

Defining an Alias using a Hostmap file

External relay programs may use an alias to identify the VisualWorks server to which they pass requests. For security, an alias is generally a safer way to specify the server than by naming the DLL directly with the server's hostname and port.

Aliases are defined in a **hostmap** file. If you choose to set up aliases, you must create this file — it is not created by VisualWorks.

The file should be in plain text format consisting of name/value pairs in the form:

```
alias: hostname@portNumber
```

The **hostname** and **portNumber** parts are the hostname and port number used to identify the server. The **alias** is any string of printable characters, except white space. The **hostmap** file also can contain blank lines and comments, which are any lines beginning with a pound sign (**#**).

The hostname may be fully qualified (e.g., **tyler.mydomain.com**) or not (**tyler**), as required by the Domain Name Server which resolves the name or your network. Alternately, an IP Address may be used (the most efficient, since it does not require a DNS lookup to resolve).

To use the alias defined in the hostmap file, make a copy of the gateway, and rename it to match **alias**, as it appears in the file.

Creating a Hostmap file under UNIX

On UNIX systems, the **hostmap** file is located here:

```
/etc/VisualWave/hostmap
```

When using the CGI relay under UNIX, the **hostmap** file should contain entries in the following form:

```
rose: tyler@2223
```

or

```
rose: 123.456.54.32@2223
```

In this case, the relay file would be named **rose**, and users would enter the string **rose** as part of the URL path used to connect to the WaveIPRequestBroker host **tyler** on port **2223**.

Creating a Hostmap file under MS-Windows

On Windows, the **hostmap** file is located in the **VisualWave** subdirectory of the Windows system root directory, e.g.:

```
%SystemRoot%\VisualWave\hostmap
```

When using a **hostmap** file under Windows, the exact form of each alias definition depends upon the type of gateway being used, e.g.:

Alias		Gateway
rose.exe:	tyler@2223	CGI
rose.dll: or rose:	tyler@2223 tyler@2223	ISAPI

Note that when using the ISAPI DLL on Windows, the extension **.dll** is not required in the alias.

If you are using the **hostmap** file to test both ISAPI and CGI relays on a single Windows machine, you need two distinct file names for the ISAPI library and the CGI executable. For example, **rose.exe** and **rose.dll** are not different, whereas **rose1.dll** and **rose2.exe** are.

Again, the hostname may be replaced with an IP address.

Specifying a Gateway Error Message

The gateway application can also be configured to return a special error message when the Application Server is unreachable. The error message may be a string specified in the gateway's INI file, or the contents of a separate file. The error message may be formatted as valid HTML.

To enable this option using a string in the INI file:

1 In the **[General]** section of the gateway's INI file, set the value of **printUserErrorMessage** to **1**.

2 Specify the error string using **userErrorMessage**.

3 Set the value of **errorHTMLMessageMAXSize**.

4 Save your modifications to the INI file.

To specify the error message contained in a separate file:

1 In the **[General]** section of the gateway's INI file, set the value of **useErrorHTMLFile** to **1**.

2 Specify the file name using **errorHTMLfile**.

3 Set the value of **errorHTMLMessageMAXSize**.

4 Save your modifications to the INI file.

6

Using Multiple HTTP Servers

VisualWorks Application Server provides a distributed architecture for scaling your application across multiple server machines. A scalable architecture enables you to process a higher volume of transactions by using a pool of dedicated application servers.

With each server machine processing a number of transactions simultaneously, a distributed architecture can provide an order of magnitude increase in system performance. It also provides better overall reliability through fault tolerance.

The core technology in VisualWorks's scalable architecture is a load balancing scheme that dynamically distributes client demand across a group of interconnected application servers. The Application Server's load balancer provides two different policies for distributing demand: a round-robin policy that can be used with non-VisualWorks servers, and a least-busy policy that provides higher performance.

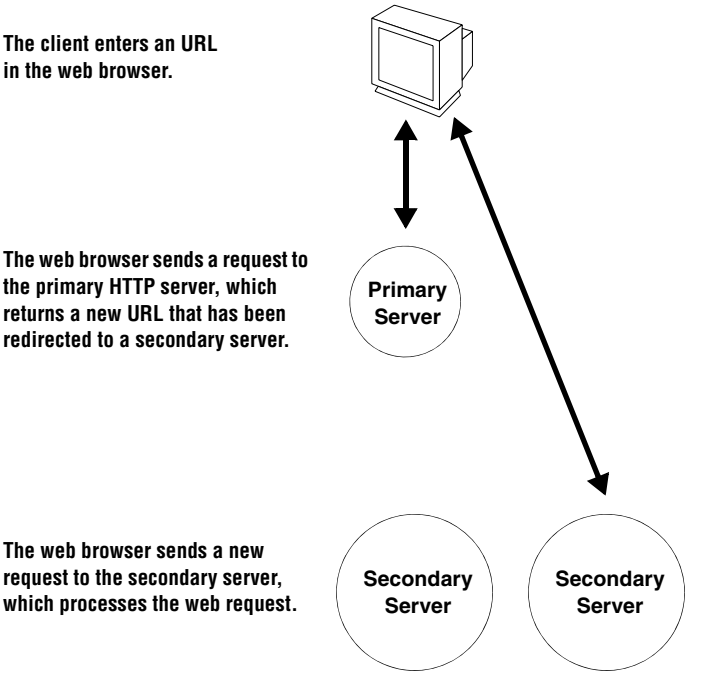
The intercommunication mechanism used for the high performance load balancing policy requires the addition of the VisualWorks OpenTalk. For an overview of OpenTalk, see the first chapters of the *OpenTalk Developer's Guide*.

This chapter describes:

- Multiple Server Architecture
- Load Balancing Policies
- Defining Servers
- Performance Monitoring and Remote Administration

Multiple Server Architecture

In the scalable architecture used by VisualWorks Application Server, one host machine is designated as the *primary server*. This server receives incoming requests from clients and then redirects them to a group of *secondary servers* (also referred to as *application servers*). The primary server handles all redirection and load balancing, while the secondary servers actually run the application to process transactions.



Interaction between the client and server thus happens in two steps: first, the client sends a web request to the primary server. The primary server uses a preset load balancing policy to redirect the request to a particular secondary server. Once the request has been redirected to a secondary server, it is processed by the application running on that server until its conclusion.

The redirection is accomplished when the primary server sends a modified URL back to the client machine (in effect, only the network address of the URL is modified). The client then closes its HTTP connection with the primary server and uses the re-directed request to open a new HTTP stream with the application server.

The primary server maintains a list of application servers, and can be configured to dynamically update the list to reflect the availability of individual hosts. The list of application servers can include any host, non-VisualWorks servers, or servers responding to different ports on the primary host. In most configurations, however, the primary host is dedicated to the activities of load balancing, administration, and monitoring.

Load Balancing Policies

VisualWorks Application Server provides two basic policies for redirection and load-balancing:

- Round-robin policy
- Least-busy policy

The two policies differ in terms of performance, robustness, and complexity.

The round-robin redirection policy works by cycling repeatedly through a predefined list of secondary servers. Each request received by the primary server is redirected to the next secondary server on the list. This policy has the virtue of being very simple to configure, and it can be used with both VisualWorks and non-VisualWorks servers.

Under the least-busy policy, the primary server monitors the availability and load reported by each secondary server so that it may redirect requests to the appropriate server. To implement the load-reporting mechanism, this configuration requires the addition of VisualWorks OpenTalk.

The least-busy policy has the virtue of providing more consistent performance as well as increased fault tolerance. The reporting mechanism enables the primary server to automatically cease redirecting requests to secondary servers which are off-line or otherwise unavailable.

It is possible to use the round-robin policy in conjunction with load reporting, so that the primary server cycles through a list of secondary servers in round-robin fashion, but only redirects requests to those servers operating below a preset load level.

Redirecting Client Requests

In order to redirect incoming requests to a particular secondary server, the primary server must resolve the request and pass it to a special redirection service. The redirection and load balancing is implemented within the standard VisualWorks Application Server framework (for details, see: "Resolving Client Requests" on page 4-7). The redirection process follows these steps:

1. The primary server listens for a client's request.
2. When a client request is received, an HTTP stream is established and passed to a WebRequestService object.
3. The WebRequestService does all of the "serving," composing a web request object. The request object is passed to the path associated with the application (a PathInfoPrefixResolver).
4. The path is associated with a special type of resolver called a RedirectionLoadBalancer, which performs the actual redirection by returning a modified URL to the client. The modified URL contains the address of a secondary server.
5. The client closes the HTTP stream to the primary server, and then opens a new stream to the secondary server.

Different paths may be used on the primary and secondary servers, because on the secondary servers, the path is resolved to an application, whereas on the primary server, it is resolved to a single RedirectionLoadBalancer.

Configuring Servers

The multiple-server architecture used by VisualWorks requires a special configuration for the primary server. If you are using a simple redirection scheme without reporting, or if you are using non-VisualWorks servers, only the primary server need be configured for load balancing. To use the load reporting mechanism, the secondary servers require additional components as well. The configuration procedures depend upon whether or not you choose to use load reporting.

Configuring the Primary Server

The VisualWorks load balancer is provided as a set of parcels on the release medium. During installation, these parcels are typically copied into the **`waveserver`** subdirectory of the VisualWorks installation.

To use the load balancer:

1 Using the Parcel Manager (choose **System → Parcel Manager** in the Launcher window), load the parcel WaveLoad-Server.

2 Open the Server Console window by selecting **Web → Server Console** in the Launcher window. If a Server Console was already open, close it and open a new one.

3 Click on the **Create Server** button, set the parameters for a new WaveHTTPRequestBroker, and then **Create** it (do not start it yet).

4 Select the server in the list and click on **Edit Resolver**.

5 To add a new resolver object, click on **Add Path**.

6 Two fields should appear in the right-hand side of the resolver panel. Select **RedirectionLoadBalancer** from the drop-down **Resolver Type** menu.

7 Enter a path identifier (for this example, use **test**) and click on the **Accept** button.

The load balancer has now been installed, but no servers have yet been defined.

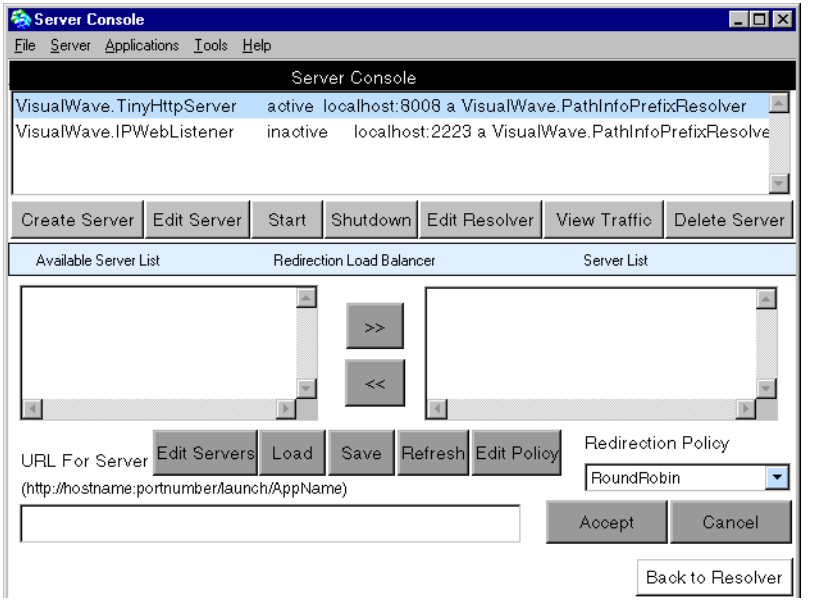
Defining Servers

To define a list of servers for use with the RedirectionLoadBalancer, you must use a special tool called the Virtual Server Editor. This tool enables you to build, load, and save lists of server bindings for both VisualWorks and non-VisualWorks application servers. The latter are known as *virtual servers*. A virtual server is essentially a named alias for the non-VisualWorks server. Secondary servers (both VisualWorks and virtual) do not need to be available when you define them.

To open the Virtual Server Editor:

1 Open the Server Console, select the primary server in the list and click on **Edit Resolver**.

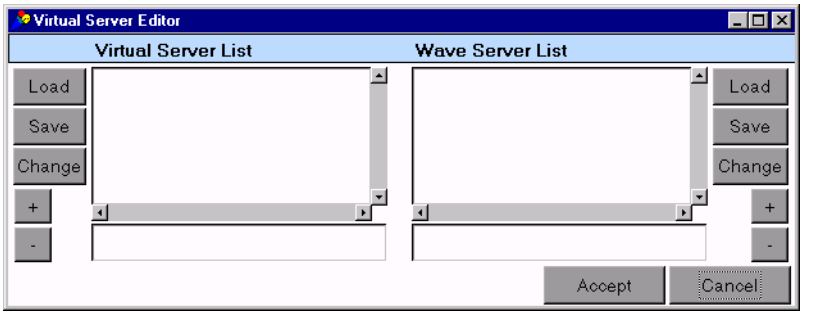
2 Select the new RedirectionLoadBalancer in the list of paths, and click on **Configure Resolver**. The load balancer panel now appears in the bottom part of the Server Console:



3 Open the Server Editor by clicking on the **Edit Servers** button.

Using the Virtual Server Editor

The Server Editor tool is divided into two sections: use the controls on the left-hand side of the window to create and maintain a list of virtual application servers (either non-VisualWorks servers, or VisualWorks servers which do not report to the primary server).



Virtual Server Editor

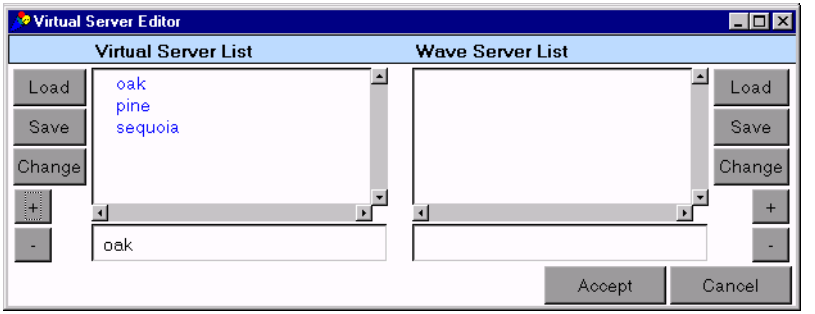
Use the list on the right-hand side to create servers which you intend to monitor with the reporting mechanism.

To create an example configuration that does not use reporting:

1 Enter a server name in the input field below the left-hand list.

2 Click on the **+** button to add the server name to the list.

3 Repeat step 2 for each server you wish to include in the group (in this example: **oak**, **pine**, and **sequoia**).



4 When you are finished, **Accept** the list of servers to close the window and return to the Server Console.

A list of “virtual” secondary servers has now been associated with the load balancer.

Defining Servers for Redirection with Reporting

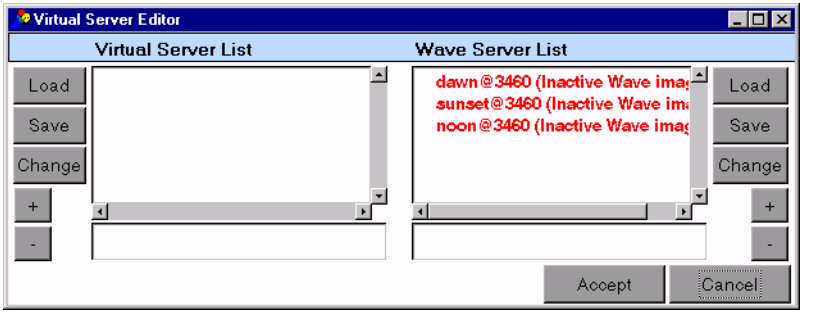
The Virtual Server Editor is also used to define servers that will be monitored by the load balancer, but the procedure is slightly different:

1 From the load balancer panel in the Server Console, open the Server Editor by clicking on the **Edit Servers** button.

2 Enter a server name in the input field below the left-hand list.

3 Click on the **+** button to add the server name to the list.

4 Repeat step 2 for each server you wish to include in the group (in this example: **dawn**, **sunset**, and **noon**). Use the syntax **hostname@port**, where **hostname** is the DNS name of the secondary server and **port** is the port number (you should use the default value: 4849).



5 When you are finished, **Accept** the list of servers to close the window and return to the Server Console.

Each server is displayed in red text and marked as **Inactive** until an OpenTalk connection has been established with the secondary server image (the steps for configuring OpenTalk are discussed later in this chapter). Inactive servers will not be used by the load balancer.

Associating Servers with URLs

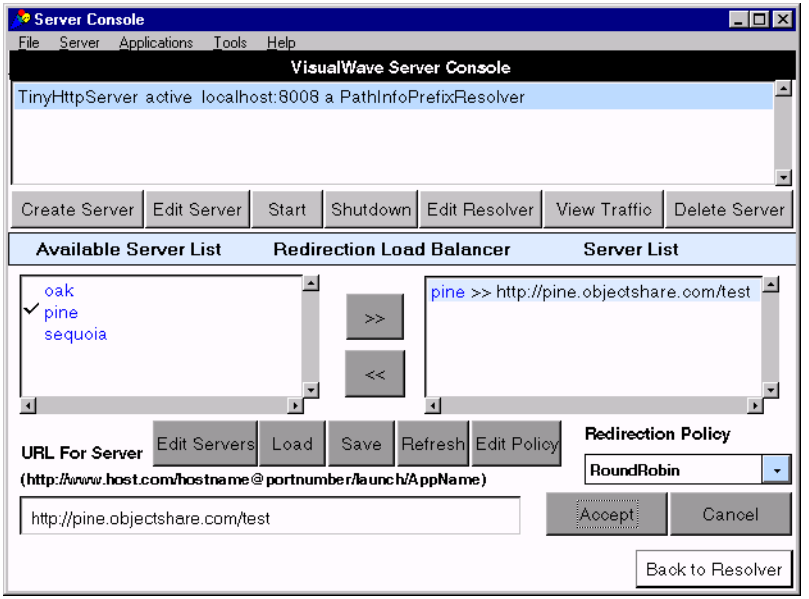
To complete the configuration, you must associate an URL with each secondary server. Use the resolver configuration panel to associate server aliases with specific URLs.

To configure redirection to a non-VisualWorks (virtual) server, or a VisualWorks server which has *not* been set up for load reporting:

1 Select the server you wish to associate with an URL, and move it to the active server (right-hand) list by clicking on the >> button (hold down the <Shift> key to select multiple servers).

Only servers in the right-hand **Server List** will be used for load balancing.

2 To establish an association, select a server from the right-hand list, enter a URL in the input field located in the lower-left corner of the panel, and then hit <Tab> or <Return>
(in this example: http://pine.objectshare.com/test).



Console Window showing Application Servers

Although the spelling of the host address of an URL is not case-sensitive, the spelling of the remainder of the path (for example, launch/Examples.CheckbookInterface) is case-sensitive.

Note: When entering a port number in the URL that is answered during redirection, make sure that the ports used for OpenTalk communication (i.e., 4849) and HTTP (i.e., 8008) are distinct.

3 Repeat step 2 for each server.

4 Select a **Redirection Policy** from the drop-down menu (in this case, use **Round Robin**).

5 **Accept** the redirection configuration, and then click on **Back To Resolver** to return to the resolver panel.

The resolver has been configured and the server can now be started.

The procedure for configuring redirection to a VisualWorks server which *has* been set up for load reporting is essentially the same: after defining servers to use reporting in the Virtual Server Editor, associate an URL with each application server and **Accept** the configuration. The primary difference in the resolver configuration panel is that you should set the **Redirection Policy** to be **Least Busy** before clicking **Accept**.

In fact, either redirection policy can be used when secondary servers are configured to do load reporting. The choice depends upon the target performance for the cluster of servers as a whole.

Once you have finished configuring the list of available servers, you can save and later re-load the entire configuration from a file by using the **Save** and **Load** buttons.

Configuring for Least-Busy Load Balancing

Using VisualWorks Application Server with least-busy load balancing requires installing several additional components in both the primary and application server images. These components implement the reporting mechanism via a client/server model, with the secondary servers acting as clients. Thus, the component configuration differs slightly for the primary and secondary servers:

Components Required by Primary Server:

- VisualWorks OpenTalk
- WaveLoad-Server parcel

Components Required by Each Secondary Server:

- VisualWorks OpenTalk
- WaveLoad-Client parcel

The following discussion provides a walk-through of the component installation. It is easiest to set up the primary server image before the secondary servers, although you can perform the installation in either order. Typically, a single deployment image is used on all application servers.

Configuring the Primary Server

To configure the primary server:

1 Using the Parcel Manager (choose **System → Parcel Manager** in the Launcher window), load the parcel WaveLoad-Server.

2 The WaveLoad-Server parcel will automatically load OpenTalk.

3 Open the Server Console window and **Create** a Smalltalk HTTP Server.

4 Select the Smalltalk HTTP Server and create a new resolver object (click on **Edit Resolver**, and then **Add Path**), setting the **Resolver Type** to be

RedirectionLoadBalancer. The **Path** you select here will be the path that clients must invoke to reach the redirection service.

5 Configure the list of secondary servers as described in the preceding sections. In the Virtual Server Editor, make sure to create servers that are configured for load reporting (use the controls and input field on the right-hand side of the window).

6 Associate URLs with the secondary servers (as described in the previous section).

7 Start the Smalltalk HTTP Server.

The primary server is now running and ready to redirect incoming requests to application servers. The Server Console will continue to indicate that the secondary servers are **Inactive** until you start them. Note that there may be a latency of a minute or two before the display is refreshed to indicate that the application servers are running. To manually refresh the status of the application servers listed in the Server Console, use the **Refresh** button.

Configuring Secondary Servers

To configure a secondary server, you will normally begin with the VisualWaveServer parcel and your VisualWorks application already loaded in the application server image.

1 Load the client component (choose **Tools → Load Parcel Named…** in the Launcher window, and enter WaveLoad-Client).

2 The WaveLoad-Client component will automatically load OpenTalk.

3 When the WaveLoad-Client component has loaded, a Client Startup tool appears that allows you to start the communication link with the primary server.

4 In the Client Startup Tool, enter the DNS name or IP address of the primary server. The **Port** should be set to the default value of 4849.

5 In the **Client Startup** tool, click on the **Start** button.

The client application server should now establish communication with the primary server.

Once OpenTalk has established communication with the primary server, you can continue the installation of your VisualWorks application. Install your application following the normal procedure (see “Loading Applications into the Server Environment” on page 3-2).

Setting Policy Restrictions

When the application servers are configured to report their load to the primary server, it is possible to set an upper limit to the number of sessions active on each secondary server. The load balancer on the primary server will skip over any application servers that have the maximum number of active sessions and issue a **System Busy** error if there are no available secondary servers.

To set a limit, do the following:

1 On the primary server, select the Smalltalk HTTP Server being used to do redirection, and click on **Edit Resolver**.

2 In the resolver registry, select the path that is mapped to the RedirectionLoadBalancer (in the example, this will be **test**) and click on **Configure Resolver**.

3 In the resolver configuration panel, select the server you wish to configure (right-hand list) and click on **Edit Policy**.

4 Set the maximum number of sessions in the input field labelled **Governor Settings**, enable the governor by clicking in the accompanying check-box, and then click on **Accept**.

Performance Monitoring and Remote Administration

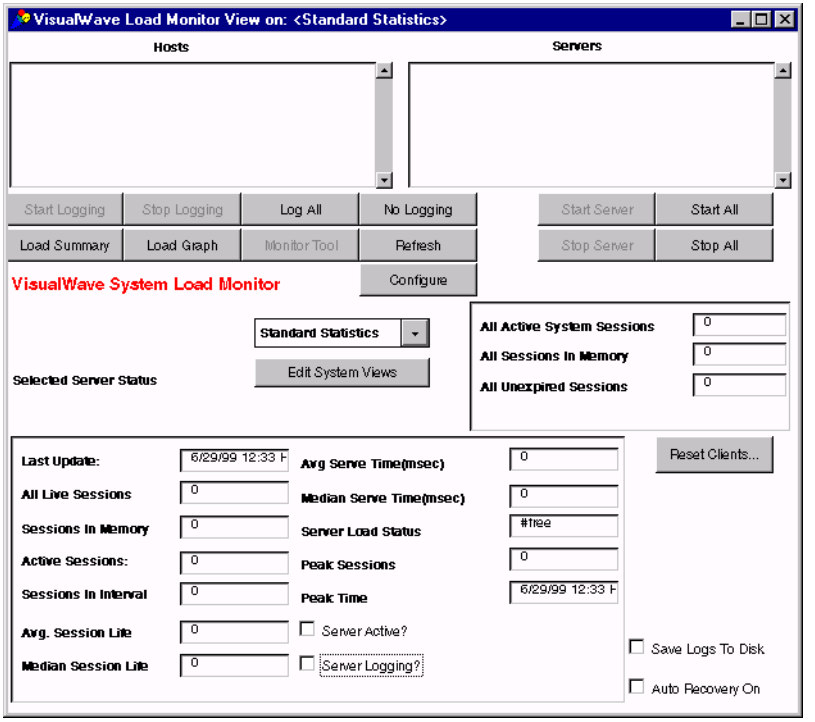
VisualWorks Application Server provides a mechanism for the remote administration of all secondary servers in a server farm. A single tool is provided for monitoring and administrative tasks. Application servers that have been configured for load reporting periodically send statistics to the primary server. Using the Load Monitor Tool on the primary server, it is possible to view the following information about each host in the server farm:

- Server status
- Average, peak, and total number of sessions
- Number of users served during logging period
- Average and median session lifetime
- Graph of session load over time

Load Monitor Tool

Use the Load Monitor tool to collect performance statistics and to control the activity of secondary servers.

To open the Load Monitor, choose **Tools → Load Monitor** from the Server Console window.



Load Monitor Tool

Use this tool to monitor and control the state of all VisualWorks servers registered with the primary server. To display information about VisualWorks servers that are active on a particular host, select its DNS name from the left-hand list. The list of servers will be displayed in the right-hand list, with the statistics for the selected server in the lower portion of the view.

To enable or disable logging on the selected host, click on **Start Logging** or **Stop Logging**. Logging may be enabled or disabled on all registered servers by clicking on **Log All** or **No Logging**.

To open a window containing a summary report on activity across all servers on a specific host, click on **System Summary**.

To start or stop a specific server, select its name in the right-hand view, and click **Start Server** or **Stop Server**.

For additional details about the Load Monitor tool, including a summary of the statistics reported by the secondary servers, see “Load Monitor Tool” on page A-36.

The check box **Is Server Logging** indicates whether or not the server is currently logging. Use **Save Logs to Disk** to record incoming statistical information in a file on the primary server (the file will be created in the current directory).

Enable **Auto Recovery On** to set the selected image to act as a backup to the primary server. If the primary server becomes unavailable, the backup server will assume control when the clients eventually reconnect as they time out.

A

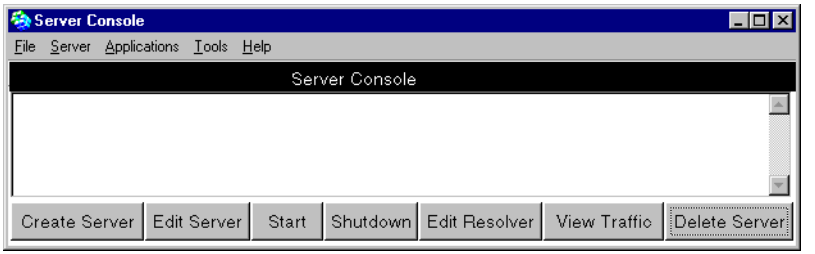
Tools Reference

This appendix contains a complete description of each of the tools in the VisualWorks Application Server environment. It includes navigation tips and descriptions of all panels, options, and commands for these tools:

- Server Console
- Application Manager
- Server Notifier
- Server Settings Tool
- Load Monitor Tool
- Bookmark Manager

Server Console

Basic Console



Basic Server Console Window

Use the Server Console to monitor and control the VisualWorks Application Server, including both the servers running within it, and the applications served from it.

The basic Server Console includes three areas, described below:

- Main Menu Bar
- Servers List
- Button Bar

Main Menu Bar

The Server Console main menu bar contains these menus:

File
: Commands that control the entire server environment, allowing you to save it, change settings, and exit.

Server
: Commands that control individual servers. These commands are also available as buttons at the bottom of the collapsed Server Console.

Applications
: Commands for loading applications and making them available to users.

Tools
: Commands for launching tools.

Help
: Lists online help items.

Servers List

The main part of the Server Console lists the individual servers that exist within the environment. For each server, the list includes:

Server type
: Either TinyHttpServer or IPWebListener.

State
: Either active (started) or inactive (shutdown).

Hostname and port number
: Network address of the server.

Auto start
: Included only if the **Start server at system start up** option is chosen for the server.

Resolver Type

All servers have an associated PathInfoPrefixResolver, which knows how to take the first segment of the path in the URL and match it to an action.

Button Bar

The button bar at the bottom of the collapsed Server Console provides operations you can perform on individual servers. With the exception of **View Traffic**, all of these commands are also available from the **Server** menu.

Create Server

Expand the Console to include the Create Server panel. From there you can create and start new servers. The **Server Type** and **Use default resolvers** values cannot be reset after the server is created.

Edit Server

Expand the Console to display the Edit Server panel. From there you can change characteristics of the selected server, such as its hostname, port, and whether or not it starts at system startup. If no server is selected, this button has no effect.

Start

Make the selected server active and available to accept web requests. If no server is selected or if the selected server is already started, this button has no effect.

Shutdown

Make the selected server inactive and no longer available to accept web requests. Web sessions in progress are expired. If no server is selected or if the selected server is inactive, this button has no effect.

Edit Resolver

Expand the Console to display the Edit Resolver panel. From there you can change the paths recognized by the selected server's PathInfoPrefixResolver and the mapping of those paths to other web responders and resolvers. If no server is selected, this button has no effect.

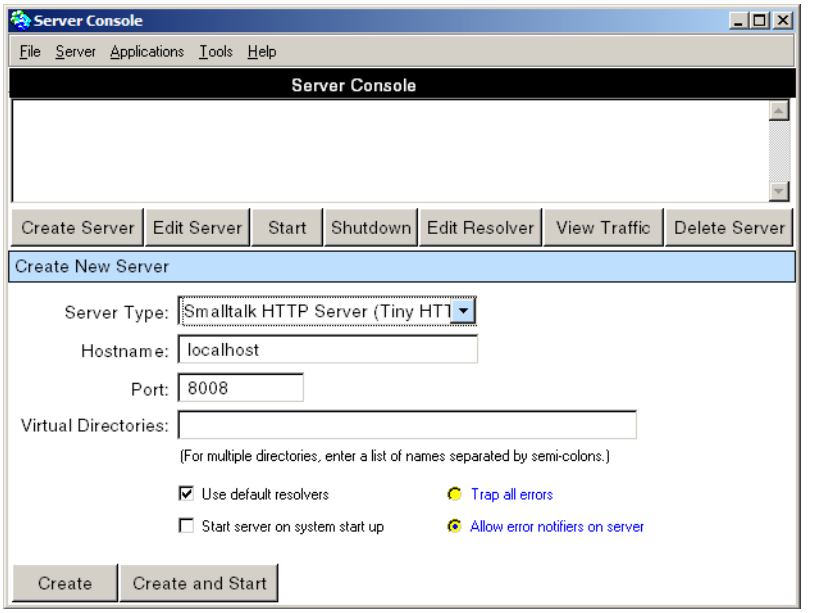
View Traffic

Expand the Console to display the View Traffic panel. From there you can see a list of the web requests received by the selected server, the entire details of a selected web request, and the sessions in which those requests were handled. If no server is selected, this button has no effect.

Delete Server

Shut down the selected server and delete it from the environment. Web sessions managed by that server are expired. If no server is selected, this button has no effect.

Extended Console: Create Server



Extended Server Console Window

Navigation

To display the Create Server panel, click the **Create Server** button in the Server Console. If this panel is already displayed, clicking **Create Server** resets the fields to their default values.

Options

Server Type

There are four types of servers:

Smalltalk HTTP Server accepts requests directly from web browsers. It does not require a standard HTTP server, and is useful for testing the server environment and applications.

External Web Server (WaveIPRequestBroker) accepts requests from a front-end HTTP server. It is recommended for deploying commercial applications and serving large numbers of users.

Hostname

Name of the machine on which the VisualWorks Application Server is running. The default, **localhost**, can be used when the web browser (for Smalltalk HTTP Servers) or HTTP server (for External Web Servers) is on the same machine as the server environment.

Port

Port on which the server listens. The defaults are:

Smalltalk HTTP Server: 8008

External Web Server: 2223

A VisualWorks server can use any port. Note that port 80 is the default value assumed when the URL does not include a port number. Note, also, that on most systems, low numbered ports (including port 80) require special system privileges. You can avoid this by choosing a higher numbered port. See your system's networking documentation for specific requirements.

Use default resolvers

When checked (the default), the server is set up to recognize and resolve three path prefixes in an URL:

launch: For starting applications. In the URL, **launch** must be followed by an application name or alias. The server hands the web request to a launching resolver, which starts the requested application. The kind of launching resolver depends on the VisualWorks product:

AnySessionLauncher: can run any application in the environment.

RegisteredSessionLauncher: can run only applications named in its registry.

echo: For checking the server and browser setup. The server hands the web request to a RequestEchoer, which returns a list of the environment variables that make up the user's web request. No further processing of the request is done.

submit: For continuing interaction with a running application. The server hands the web request to a SessionResolver, which finds the appropriate session (and application) to handle the input.

JavaSTGM.class: Used to send Smalltalk graphics as Java-rendered graphics.

Start server on system start up

When chosen, the server will restart automatically whenever the server environment is started. By default, the server is shut down when the server environment is exited and the server remains shut down until you restart it.

Commands

Create

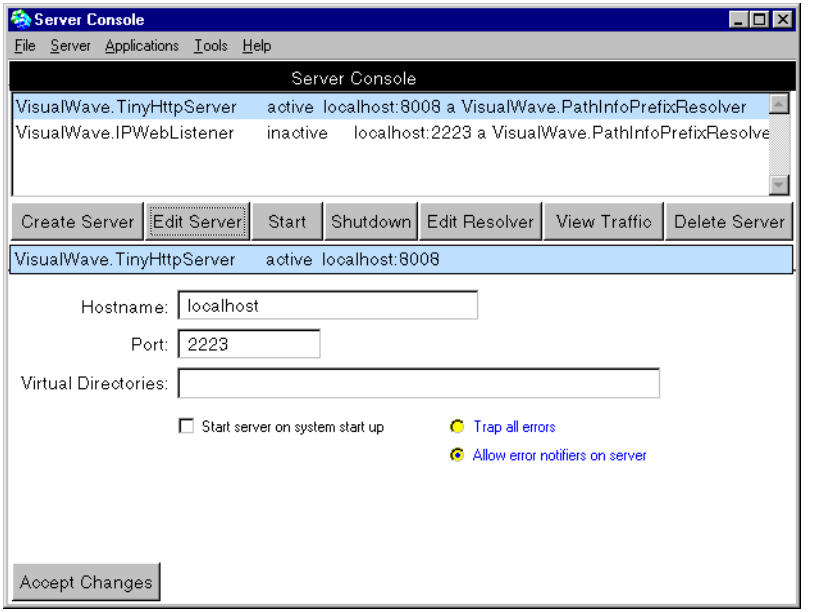
Create a server with the characteristics specified above. The server created is inactive and unable to process web requests.

Create and Start

Create a server with the characteristics specified above *and* start it. The server is active and able to process web requests.

The server is not created until you click the **Create** or **Create and Start** button. Any other action that causes the Server Console to dismiss the Create Server panel, cancels creation of the server.

Extended Console: Edit Server



Editing Server Characteristics

To display the Edit Server panel, click the **Edit Server** button in the Server Console. If the Edit Server panel is displayed and you select another server in the Servers List, the panel updates to display information about the newly-selected server.

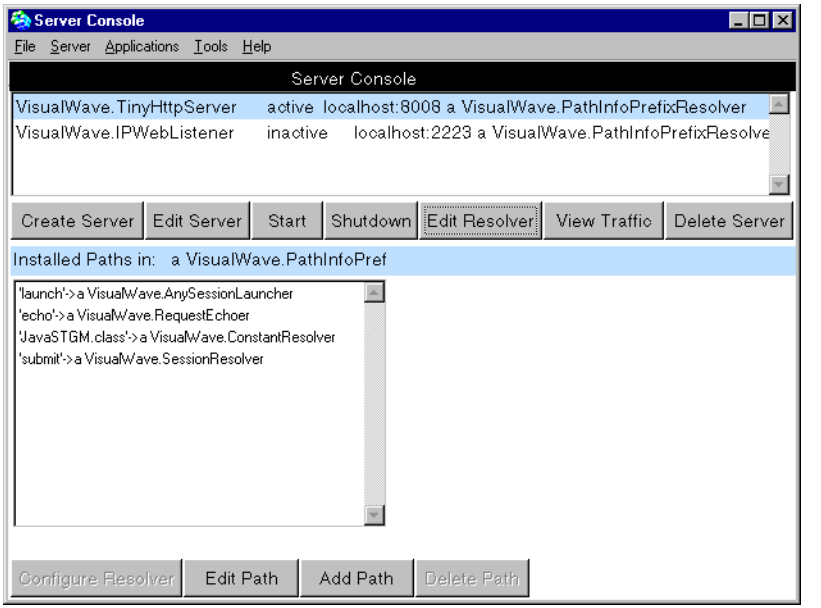
The fields on the Edit Server panel match those on the Create Server panel, with two exceptions:

- You cannot change the type of the server.
- You cannot choose to use or not use default resolvers.

The choice to use default resolvers and the selection of this type can only be made when you create a server.

Changes take effect when you click the **Accept Changes** button. If you close the Edit Server panel without accepting changes, your changes are lost.

Extended Console: Edit Resolver



Editing a Resolver

The Edit Resolver panel allows you to configure the PathInfoPrefixResolver for a particular server. The server's PathInfoPrefixResolver is the object that reads the first path segment in the URL and hands the web request to the appropriate object for further resolving or responding.

Navigation

When you select a server and click the **Edit Resolver** button, the Server Console expands to include the Edit Resolver panel. When you select another server in the servers list, the Edit Resolver panel updates to display information about the resolver for the newly-selected server.

To return to the Edit Resolver panel from a Configure Resolver panel, click the **Back to Resolver** button in the bottom right corner.

Contents

Installed Paths: Displays the registry for the selected server's PathInfoPrefixResolver. The registry of the PathInfoPrefixResolver maps string path fragments (on the left of the arrow) to objects that can further resolve or respond to the web request (on the right). When the user

makes a web request, the first segment of the path in the URL is matched against the paths in the registry. For example, assume that the user requests this URL:

```
http://www.yoursite.com:8008/launch/WebBrowser
```

And assume that the server listening on port 8008 has the following mapping in its PathInfoPrefixResolver's registry:

```
'launch'->a RegisteredSessionLauncher
```

The PathInfoPrefixResolver looks at the first segment of the URL after the hostname and port number, and finds the string *launch*. It then looks in its registry for a path launch and finds launch mapped to a RegisteredSessionLauncher. Based on that mapping, the PathInfoPrefixResolver then hands the web request to a RegisteredSessionLauncher.

Commands

Configure Resolver

Allows you to configure a resolver that is mapped to a path in the **Installed Paths** list. The following four resolvers can be configured: RegisteredSessionLauncher, SessionResolver, RedirectionAnswer, or FileResponder.

When you select a path that is mapped to a resolver of one of those kinds, the Server Console replaces the Edit Resolver panel for the PathInfoPrefixResolver with a panel for editing the kind of resolver you selected. For details on these panels and the configuration options for their associated resolvers, see "Extended Console: Configure Resolver" on page A-13.

Add Path

Allows you to add a new path to the registry shown in the **Installed Paths** list and map it to one of several kinds of responders and resolvers. The **Add Path** command displays additional information and options on the right side of the Edit Resolver panel. Those panels and options are described below.

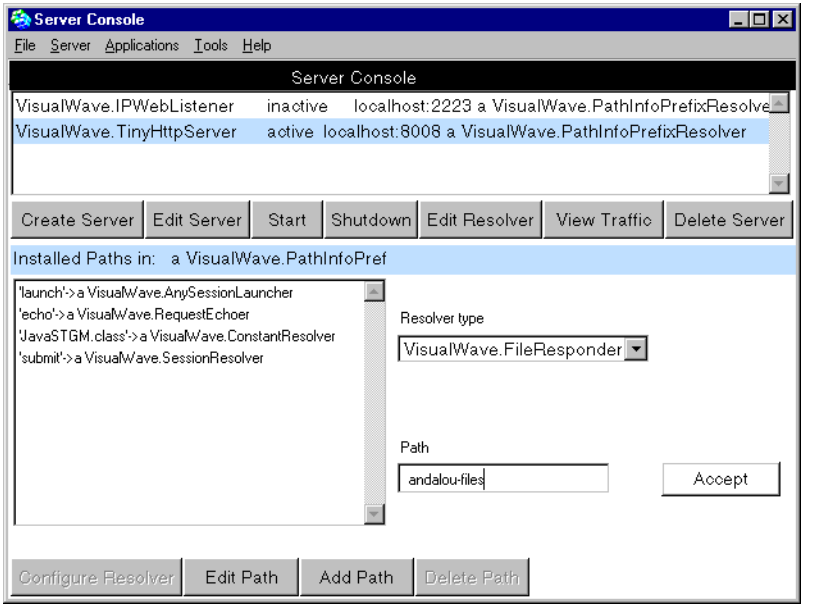
Edit Path

Allows you to edit the path selected in the **Installed Paths** list and, for paths mapped to launching resolvers, change the corresponding session resolver. The **Edit Path** command displays additional information and options on the right side of the Edit Resolver panel. Those panels and options are described below.

Delete Path

Delete the selected path selected in the **Installed Paths** list.

Extended Console: Add Path



Extended Server Console Window

The Add Path extensions to the Edit Resolver panel allow you to create a new mapping and add it to the registry for the selected server's PathInfoPrefixResolver.

Navigation

To display the Add Path panel from anywhere:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **Edit Resolver** button.

4 Click the **Add Path** button.

Options

The exact fields displayed depend on the **Resolver type** that you choose.

Resolver Type

RequestEchoer: Returns the web request exactly as it was received, listing the environment variables and their values and the Application Server that responded to the request. RequestEchoers are useful for testing a server.

RedirectionAnswer: Redirects the user to another URL. The URL can be another web site or another server or application within the VisualWorks environment.

RegisteredSessionLauncher: Opens the application specified in the URL, provided that the application is registered with the RegisteredSessionLauncher.

AnySessionLauncher: Opens the application specified in the URL. The application name must be the name of an application model class. It can be any application model in the server environment.

SessionResolver: Passes the web request (usually a continuation of an interaction in progress) to an existing session. The session is determined by checking the form data in the web request for a session key and matching that key to a session.

FileResponder: Returns a file to the requestor. This can be used for returning a static HTML file, a graphic, or any other file.

SessionResolver

If the **Resolver Type** is RegisteredSessionLauncher or AnySessionLauncher, allows you to specify the SessionResolver into which applications are launched and by which they are managed. You can launch all applications with the same SessionResolver or you can specify a different SessionResolver for each launching resolver. Using different SessionResolvers is useful when you want to use different timeout policies.

Path

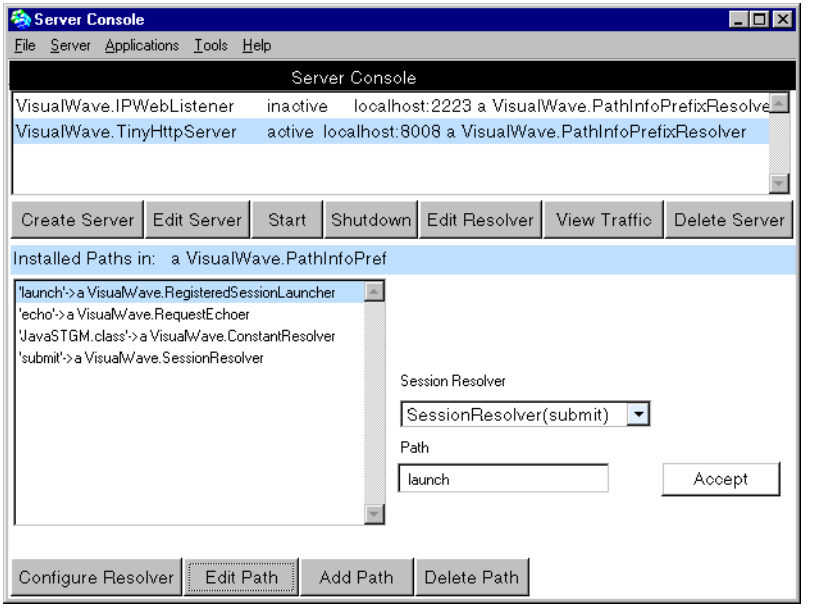
The string to map to the **Resolver Type**. The Path appears in the requesting URL. It can be any string that is valid as a single segment in an URL. It cannot contain a slash or question mark.

Commands

Accept

Create the path mapping; adds it to the **Installed Paths** list and the PathInfoPrefixResolver's registry. If you leave the Add Path panel without accepting, your specifications are dismissed and the path mapping is not created.

Extended Console: Edit Path



Editing a Path

The Edit Path extensions to the Edit Resolver panel allow you to edit an existing mapping in the registry for the selected server's PathInfoPrefixResolver.

Navigation

To display the Edit Path panel from anywhere:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **Edit Resolver** button.

4 In the **Installed Paths** list, select a path mapping.

5 Click the **Edit Path** button.

Options

The exact fields displayed depend on the type of responder resolver in the mapping that you select.

SessionResolver

If the resolver is a RegisteredSessionLauncher or an AnySessionLauncher, allows you to change the SessionResolver into which applications are launched and by which they are managed. You can launch all applications with the same SessionResolver or you can specify a different SessionResolver for each launching resolver. Using different SessionResolvers is useful when you want to use different timeout policies.

Path

The string to map to the responder or resolver. The Path appears in the requesting URL. It can be any string that is valid as a single segment in an URL. It cannot contain a slash or question mark.

Commands

Accept

Change the path mapping in the **Installed Paths** list and the PathInfoPrefixResolver's registry. If you leave the Edit Path panel without accepting, your specifications are dismissed and the path mapping is not changed.

Note: You cannot change the type of responder or resolver that is referenced in the mapping. If you want to change the type of resolver, you must delete and recreate (add) the path.

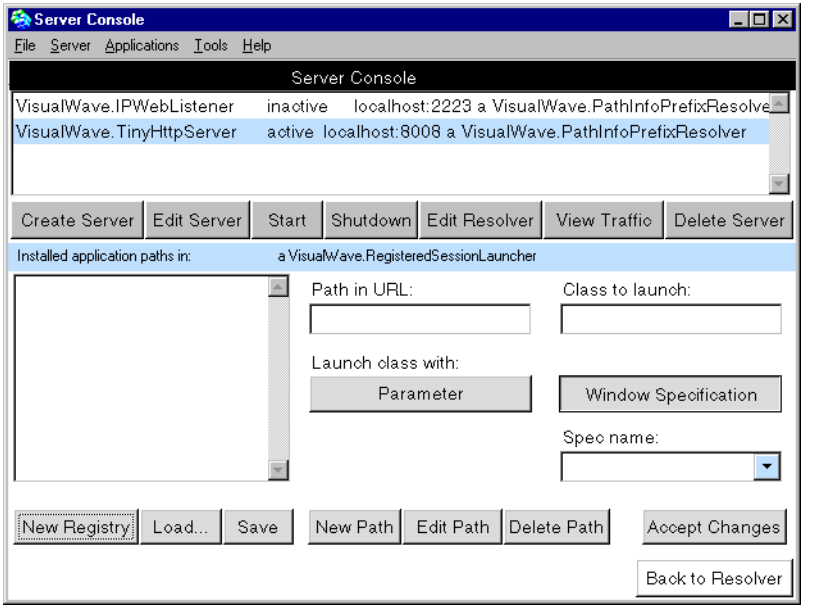
Extended Console: Configure Resolver

The Configure Resolver panels allows you to configure a resolver that is mapped to a path in the **Installed Paths** list. The following kinds of resolvers can be configured:

- RedirectionAnswer
- RegisteredSessionLauncher
- SessionResolver
- FileResponder

The contents of the Configure Resolver panel depend on the type of resolver you are configuring.

Configure RegisteredSessionLauncher



Configuring a RegisteredSessionLauncher

Navigation

To display the Configure Resolver panel for a RegisteredSessionLauncher:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **Edit Resolver** button.

4 In the **Installed Paths** list, select a path that is mapped to a RegisteredSessionLauncher.

5 Click the **Configure Resolver** button.

Options

Installed Application Paths

Displays a list of the paths in the registry for the RegisteredSessionLauncher. The RegisteredSessionLauncher attempts to match the segment of the URL to a path in this list. Each path is mapped to the name of an application model class to open in response to the web request.

Path in URL

The string displayed in the **Installed Application Paths** list and that appears in the requesting URL. The string can be any valid segment of an URL. It cannot contain a slash or a question mark.

Class to Launch

The name of the application model class that is to be opened when the path is found in the requesting URL.

Launch class with

Allows you to specify to open the class with a parameter or a window specification:

Parameter: When clicked, displays two fields in which you can specify a message send that executes when the path is requested:

Class to send message to: The name of the class to which to send the message.

Message name: The message to send to the class. The modifier is an instance of MessageChannel and can take up to one argument. If the specified message selector takes an argument, the web request is passed in for reference. The result of this message send becomes an argument to the method ApplicationModel>>open:withPolicy:inSession:. You should override this method to handle your argument appropriately.

Window Specification: When clicked, displays a menu button from which you can choose the window specification to use when opening the class.

Spec name: Contains the names of all of the window specifications for the class named in the **Class to Launch** field. Choose the window specification to use. The default is windowSpec.

Commands for Managing Registries

New Registry

Creates a new, empty registry. A dialog allows you to choose the initial contents of the new registry:

- Load a registry from a file
- Load a registry from a parcel that is in the environment. If the parcel does not contain a registry, the system creates one that includes the application model classes.
- Create an empty registry.

Load

Load a registry or a parcel. A dialog allows you to choose whether to load a registry or a parcel and allows you to choose the source:

- Registries can be loaded from files or from parcels that are already in the environment
- Parcels can be loaded from parcel files

Save

Save the registry. A dialog allows you to choose whether to save the registry to a file by itself or with a parcel to a parcel file.

Commands for Editing Application Paths

New Path

Clear the path options so that you can create a new path. The new path you specify is created when you click the **Accept Changes** button.

Edit Path

Enables editing of the options for the path that is selected in the **Installed Application Paths** list. As a shortcut, you can also enable editing simply by moving the keyboard focus to any of the options. Changes take effect when you click the **Accept Changes** button.

Delete Path

Deletes the path that is selected in the **Installed Application Paths** list. **Delete Path** takes effect immediately; you do not have to accept the change.

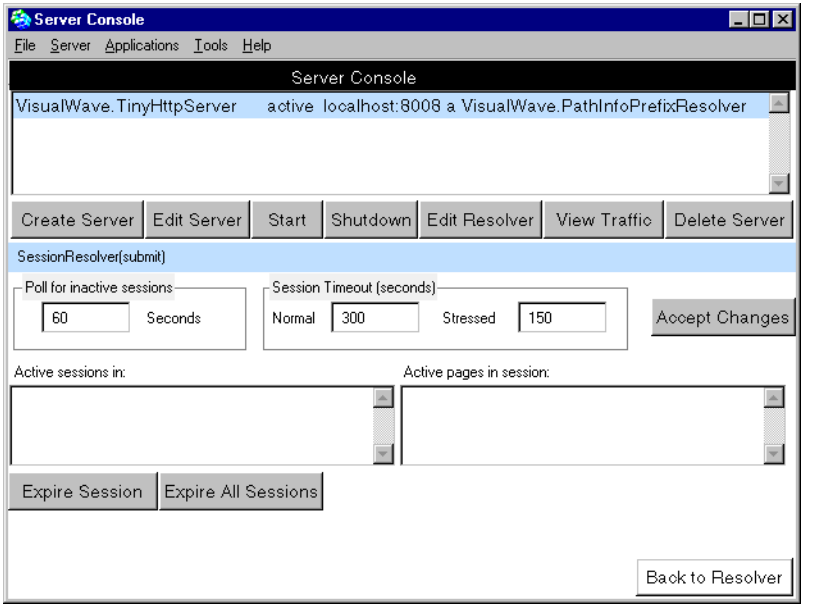
Accept Changes

Causes the current changes to take effect. Used in combination with **New Path** or **Edit Path**.

Back to Resolver

Returns to the Edit Resolver panel, which displays the installed paths for the server's PathInfoPrefixResolver. See "Extended Console: Edit Resolver" on page A-8.

Configure SessionResolver



Configuring a SessionResolver

Navigation

To display the Configure Resolver panel for a SessionResolver:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **Edit Resolver** button.

4 In the **Installed Paths** list, select a path that is mapped to a SessionResolver.

5 Click the **Configure Resolver** button.

Options

Poll for inactive sessions

In part, determines how often the SessionResolver checks for inactive sessions. When a launch request is received, the SessionResolver checks to see how much time has passed since it last checked for inactive sessions. If the amount of time is more than the amount in this field, the SessionResolver expires inactive sessions. Note that if a launch request does not occur, the SessionResolver does not check for inactive sessions.

Changes take effect when you click the **Accept Changes** button. Values are rounded to the nearest integer.

Session Timeout

Determines how long a session must go without interaction from the user before it is considered "inactive" and eligible for expiration. You can set two timeout values:

Normal: Determines the timeout period when the server environment is under normal memory use conditions.

Stressed: Determines the timeout period when the server environment is under stressed memory conditions. You can set the memory amount that is considered "stressed." See "Aggressive Session Expiration" on page B-9.

Active Sessions

Lists the active sessions that are under the control of this SessionResolver. For each session, the list displays the session key (in parentheses) and the last time the session received input from the user.

Active Pages in Session

Lists the pages that are still in use by the selected session (in the **Active Sessions** list). For each page, the list includes the kind of page key (in parentheses) and the page name (equivalent to the window title). The user can interact with pages on this list. The user cannot interact with pages that are not on this list, even if those pages are cached in his/her web browser. Attempts to interact with stale pages redirect the user to active pages.

Commands

Accept Changes

Cause changes to take effect. Used in conjunction with the **Poll for Inactive Sessions** and **Session Timeout** fields.

Expire Session

Expire the selected session. Expiring a session removes it from the list, closes applications running within it, and ends user interaction. If the user attempts to interact with an expired session, a Session Expired alert condition results. You can customize the response to that alert condition. See "Alert Condition Options" on page B-3.

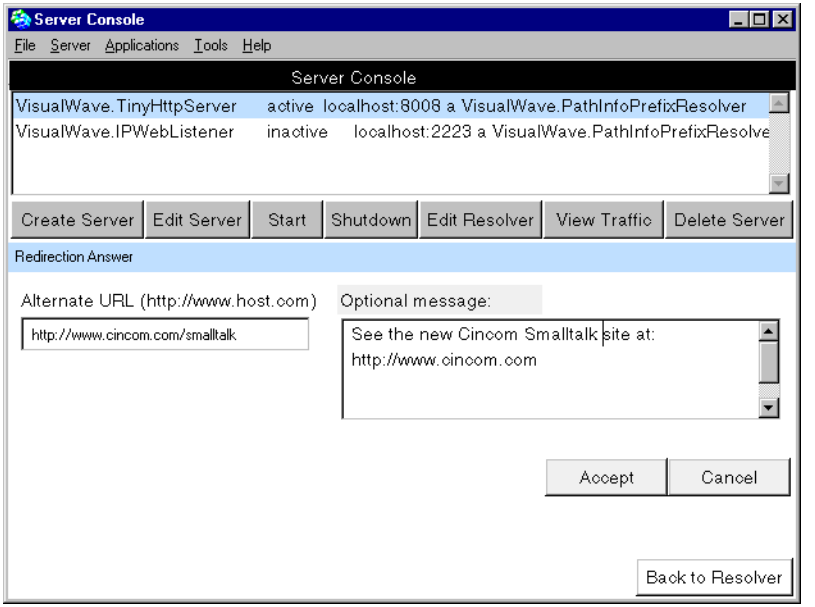
Expire All Sessions

Expire all sessions for the current SessionResolver. The results are similar to **Expire Session**, except that they affect all sessions.

Back to Resolver

Return to the Edit Resolver panel, which displays the installed paths for the server's PathInfoPrefixResolver. See "Extended Console: Edit Resolver" on page A-8.

Configure RedirectionAnswer



Configuring a RedirectionAnswer

Navigation

To display the Configure Resolver panel for a RedirectionAnswer:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **Edit Resolver** button.

4 In the **Installed Paths** list, select a path that is mapped to a **RedirectionAnswer**.

5 Click the **Configure Resolver** button.

Options

Alternate URL

The URL to send to the user's web browser with the redirection instruction. The URL can redirect the user's browser to another web site or to another server or application in the VisualWorks environment. Every RedirectionAnswer must have an **Alternate URL** value. The value must be a complete URL, beginning with **http://**, even if the redirection is to another server application in the VisualWorks environment.

Optional message

For web browsers that don't support redirection, this message is displayed. The optional message can be any value. Typically, the optional message contains text and a link that redirects the user to another URL.

Commands

Accept

Cause changes to take effect.

Cancel

Revert to the last accepted values.

Back to Resolver

Return to the Edit Resolver panel, which displays the installed paths for the server's PathInfoPrefixResolver. See "Extended Console: Edit Resolver" on page A-8.

Configure FileResponder

Navigation

To display the Configure Resolver panel for a RedirectionAnswer:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **Edit Resolver** button.

4 In the **Installed Paths** list, select a path that is mapped to a **FileResponder**.

5 Click the **Configure Resolver** button.

Options

Default directory

Specify the default root file directory for the FileResponder path name. Files will be contained either in this directory or in subdirectories, with file path references specified relative to the default directory.

Commands

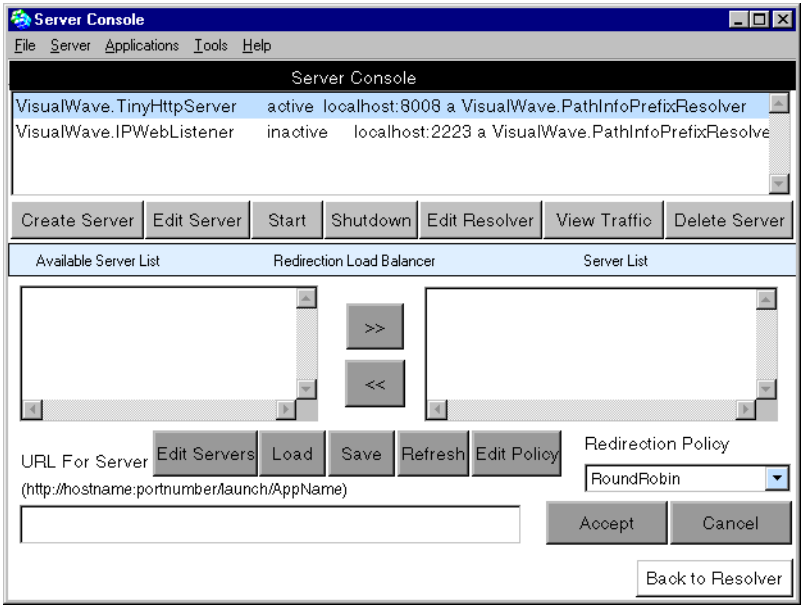
Accept

Cause changes to take effect.

Cancel

Revert to the last accepted values.

Configure RedirectionLoadBalancer



Configuring a RedirectionLoadBalancer

Navigation

To display the Configure Resolver panel for a RedirectionLoadBalancer:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **Edit Resolver** button.

4 In the **Installed Paths** list, select a path that is mapped to a **RedirectionLoadBalancer**.

5 Click the **Configure Resolver** button.

The two list views in the Configure RedirectionLoadBalancer panel display the group of secondary servers associated with this primary server. Available servers are displayed in green text, while servers that are unavailable are displayed in red text. Only the servers displayed in the right-hand list will be considered during the redirection process.

Options

Redirection Policy

The redirection policy to use with this load balancer. Choose from **Round-Robin** or **Least Busy**. Either policy may be used when the secondary servers are running VisualWorks. When using the load balancer with non-VisualWorks servers, only **Round-Robin** can be used.

URL for Server

Enter the URL to be associated with the secondary server that is selected in the right-hand list (servers currently queued), and type **`<Return>`** to accept the value.

Commands

>>

Add the selected server (on the left-hand list) to the list of servers receiving incoming requests.

<<

Remove the selected server (on the right-hand list) from the list of servers receiving incoming requests.

Edit Servers

Open a Virtual Server Editor on the list of available servers.

Load

Prompt to load a previously-saved server configuration file from the host computer's file system.

Save

Prompt to save a server configuration file in the host computer's file system.

Refresh

Update the status of secondary servers, showing which are available and which are unavailable for use with the least busy redirection policy.

Edit Policy

Open an editable properties view on the currently selected server. Use this feature to set a limit on the number of active sessions permitted on any particular server.

Accept

Cause changes to take effect.

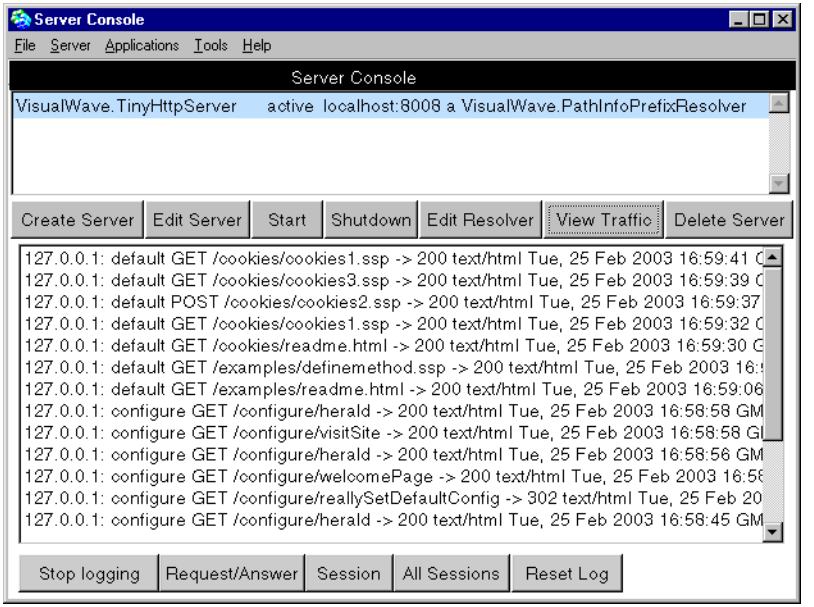
Cancel

Revert to the last accepted values.

Back to Resolver

Return to the Edit Resolver panel, which displays the installed paths for the server's PathInfoPrefixResolver.

Extended Console: View Traffic



Viewing Server Traffic

Navigation

To display the View Traffic panel:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **View Traffic** button.

Contents

The View Traffic panel displays a list of web hits responded to by the selected server. For each web hit, the list includes information about the web request, and information about the web answer.

Web request information (starting at left):

- Remote machine name
- Method used to return the form data (**`GET`** or **`POST`**)
- The path from the URL (includes all segments after the hostname and port number)

Web answer information (starting at the ->):

- The error code, if an error occurred, or nil if the web request was successful.
- The mime type of the web answer
- The time of the web answer

Commands

Stop Logging and **Resume Logging**

Stop and resume logging of web hits for the selected server. When logging is stopped, the View Traffic panel is not updated as additional web requests are handled. When logging is resumed, logging begins with the next web request received. Web requests received while logging was turned off are not displayed.

Request/Answer

Display details about the selected web hit, including the complete web request, the web answer, and the time it took to handle the web request. For more information about the Request/Answer panel, see “Extended Console: View Request/Answer” on page A-25.

Session

Display details about the session that handled the selected web hit. **Session** is applicable only for requests that were handled by SessionResolvers (submits). For more information about the Session panel, see “Extended Console: View Session” on page A-27.

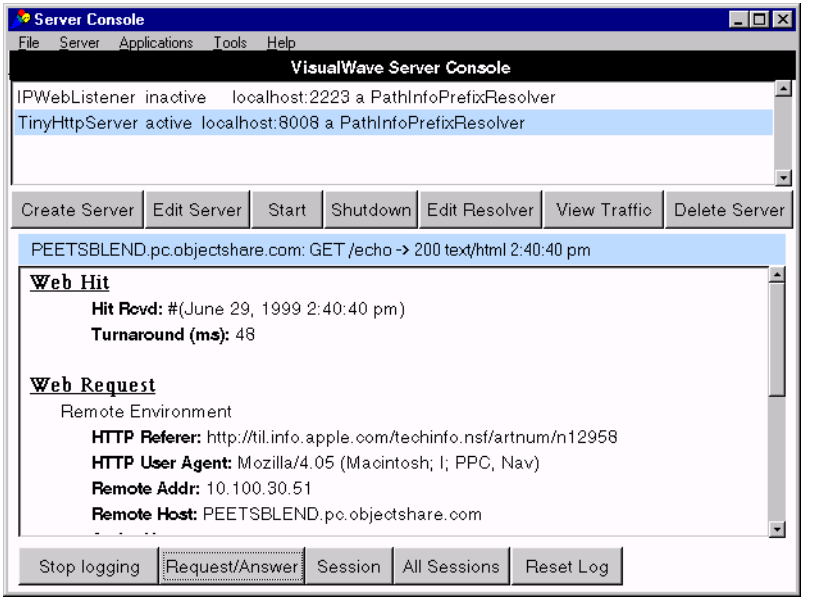
All Sessions

Display details about all sessions for the selected server. For more information about the All Sessions panel, see “Extended Console: View All Sessions” on page A-28.

Reset Log

Clear the View Traffic log. Logging continues with the next web request.

Extended Console: View Request/Answer



Viewing Web Hit Statistics

Navigation

To display the View Request/Answer panel:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **View Traffic** button.

4 Select a web hit from the traffic log.

5 Click the **Request/Answer** button.

Contents

The View Request/Answer panel displays details about the selected web hit, including the complete web request, the web answer, and the time it took to handle the web request.

Web Hit:

Includes the time that the web request was received and the time it took to process the web request and return an answer.

Web Request:

Includes the environment variables that make up the web request and their values. For web hits that are continuations of an existing user interaction (submits), the form data includes the session key and page (inline) key.

Web Answer:

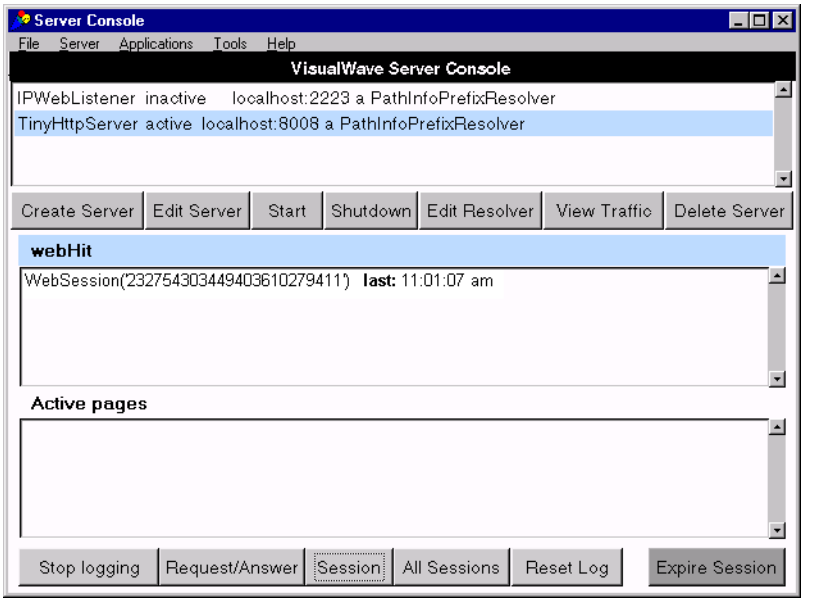
Information about the response returned by the server to the user's browser:

Status: The status code of the response. An empty string indicates that the request was successfully answered. An error code number indicates that there was a problem answering the request.

Reason: An empty string indicates that the request was successfully answered. If there was an error, a short explanation appears here.

Media: The kind of object that was returned. For launches, this is the name of the class that was opened.

Extended Console: View Session



Viewing Session Information

Navigation

To display the View Session panel:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **View Traffic** button.

4 Select a web hit from the traffic log.

5 Click the **Session** button.

Contents

The View Session panel displays details about the session that handled the selected web hit. **Session** is applicable only for requests that were handled by SessionResolvers (submits).

The View Session panel has two lists:

- The top list displays a single entry for the session, including the session key (in parentheses), the last time input was received from the user, and the referring HTTP server (if any).

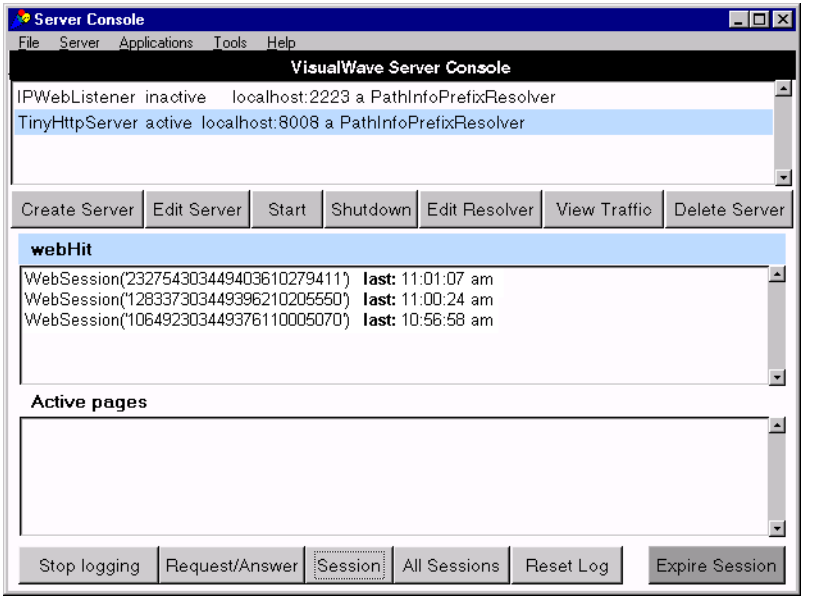
- The bottom list displays all of the pages that are still in use by the selected session. For each page, the list includes the page key (in parentheses) and the page name (equivalent to the window title). The user can interact with pages in this list. Attempts to interact with stale pages redirect the user to active pages.

Commands

Expire Session

Expire the selected session. Expiring a session removes it from the list, closes applications running within it, and ends user interaction. If the user attempts to interact with an expired session, a **Session Expired** alert condition results. You can customize the response to that alert condition. See "Alert Condition Options" on page B-3.

Extended Console: View All Sessions



Viewing All Sessions

Navigation

To display the View All Sessions panel:

1 Display the Server Console.

2 Select a server in the server list.

3 Click the **View Traffic** button.

4 Select any web hit, except "echo."

5 Click the **All Sessions** button.

Contents

The View All Sessions panel displays details about all the sessions for the selected server. This panel has two lists:

- The top list displays an entry for each session. Each entry includes the session key (in parentheses), the last time input was received from the user, and the referring HTTP server (if any).
- The bottom list displays all of the pages that are still in use by the selected session. For each page, the list includes the WebPageController, the page key (in parentheses), and the page name (equivalent to the window title). The user can interact with pages in this list. Attempts to interact with stale pages redirect the user to active pages.

Commands

Expire Session

Expire the selected session. Expiring a session removes it from the list, closes applications running within it, and ends user interaction. If the user attempts to interact with an expired session, a Session Expired alert condition results. You can customize the response to that alert condition. See "Alert Condition Options" on page B-3.

Application Manager

Use the Application Manager enables you to parcels and application registries.

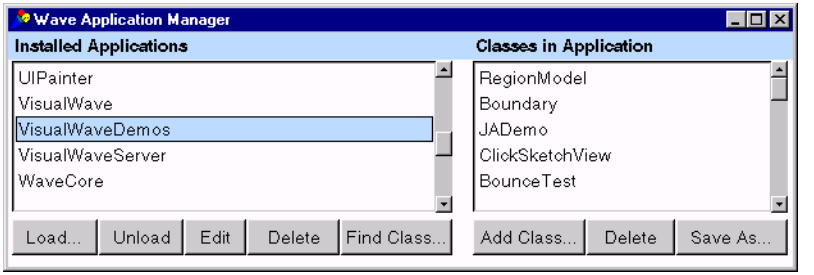
The Application Manager is similar to the Parcel List in that it displays a list of the parcels that are in the environment and enables you to load, delete, and save parcels. The Application Manager is similar to the Parcel Browser in that it displays a list of classes in the selected parcel and lets you add classes to the parcel. The Application Manager differs from both in that it allows you to unload a parcel's classes and it allows you to create and save an application registry with a parcel.

The Application Manager takes two primary forms:

- A basic Application Manager for managing parcels
- An extended Application Manager for managing parcels and their application registries

These two forms are described below.

Application Manager: Load



Application Manager Window

Navigation

To display the Application Manager in its basic form for loading parcels:

1 Display the Server Console.

2 From the **Applications** menu, choose **Load…**

Contents

The main part of the Application Manager is a list of the parcels that are currently loaded into the server environment.

Commands

Load

Load a parcel from a parcel file into the server environment. A dialog prompts you for the parcel file name. If the parcel file contains an application registry, that registry is also loaded.

Unload

Remove the selected parcel *and* its contents from the environment. Registries that are installed in RegisteredSessionLaunchers are not unloaded.

Edit

In the server environment, extend the Application Manager to include the list of classes that are in the selected parcel. In a server-enabled development environment, opens the Parcel Browser on the selected parcel.

Delete

Remove the selected parcel from the system. The parcel's contents (classes and methods) remain in the system.

Find Class

Search all of the parcels in the environment for the class that you specify. Wildcard characters are allowed in the search string:

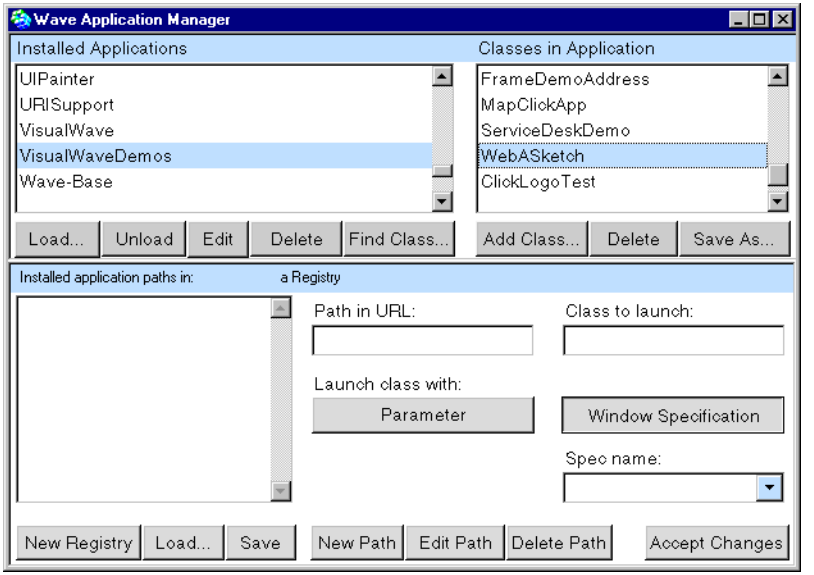
matches a single character

* matches one or more characters

If the class exists in more than one parcel, you are prompted to choose from a list of parcels that contain the class. When a matching class is found, the Application Manager expands to the right, displaying all of the classes in the same parcel as the selected class. See "Application Manager: Configure" on page A-31 for information about the buttons below the **Classes in Application** list.

The **Find Class** command does *not* check non-parceled classes in the system.

Application Manager: Configure



Extended Application Manager Window

Navigation

To display the Application Manager in its extended form for loading parcels and configuring application registries:

1 Display the Server Console.

2 From the **Applications** menu, choose **Configure…**

Contents

The top right corner of the extended Application Manager contains a list of the classes that are in the selected parcel.

The bottom of the extended Application Manager is identical to the Configure Resolver panel for a RegisteredSessionLauncher. Note that the **Installed Application Paths** list displays the registry that you are configuring; it is *not* dependent on the **Installed Applications** list above. The Installed Applications are displayed only for reference while creating registries.

Commands

Add Class

Prompt for a class name and adds that class to the selected parcel. The class name can be any class in the environment.

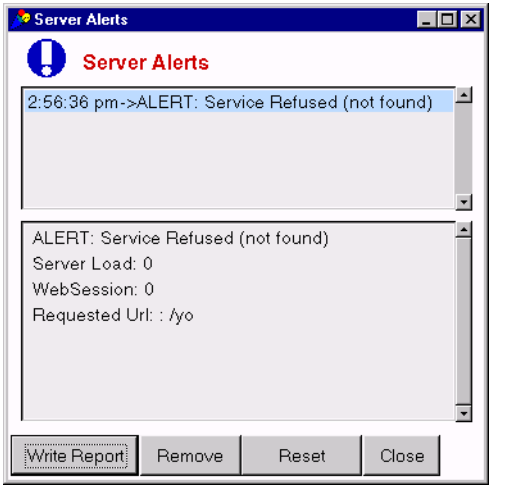
Delete

Delete the selected class from the selected parcel and, optionally, deletes the class from the system.

Save As

Save the parcel and its contents to a file. Saving a parcel does not save a registry with it. To save a registry and a parcel together, use the **Save** button at the bottom of the panel and save the registry to a parcel.

Server Notifier



Server Notifier

In the server environment, the Server Notifier replaces all other notifiers.

Alerts can also be logged directly to a file. See the **Log Alerts**, **Log to File**, and **Log File Name** options discussed in "Server Environment Settings" on page B-1.

Navigation

The Server Notifier is controlled by the Display Alerts option in the Server Settings tool. When Display Alerts is **On**, the Server Notifier displays on the server environment's host machine whenever any type of alert condition is encountered.

Contents

The Server Notifier is divided into two subviews:

- The Alerts List (top subview) contains an entry for each alert that has occurred since the Server Notifier was last reset. All types of alert conditions are included: Application Error, Service Not Found, Launch Refused, Submit Refused, Session Completed, and Session Expired.
- The Alert Detail (bottom subview) provides information about the alert that is selected in the Alert List. It includes information such as the kind of alert, the session key, and the requested URL. The requested URL includes all of the URL except the hostname and port.

Commands

Write Report

Write a report of all the alerts. For each alert, the report includes the alert details. The report matches the format of the log file.

Remove

Remove the selected alert from the Alerts List. Removing alerts is useful when you want to write a report that includes only a subset of the alerts.

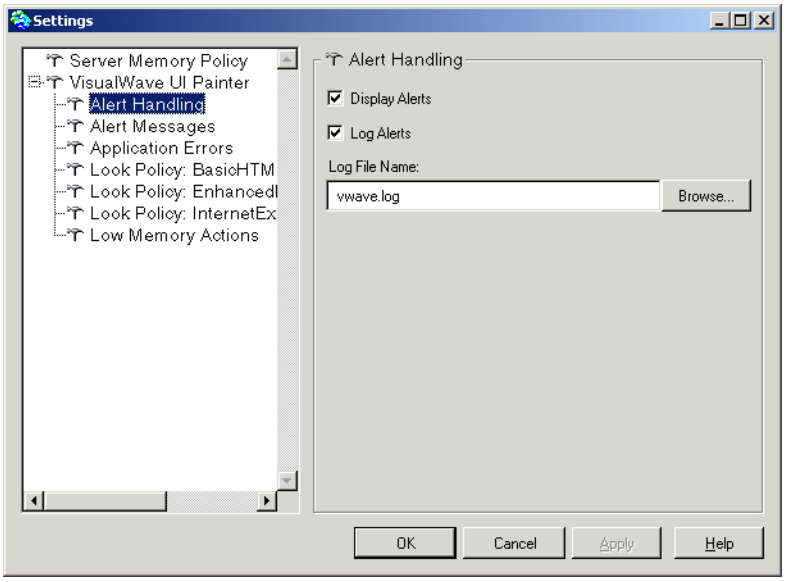
Reset

Remove all alerts from the list in the Server Notifier.

Close

Close the Server Notifier. Closing the Server Notifier does not clear the alerts. When the next alert occurs, the Server Notifier reopens itself and includes both the old and new alerts.

Server Settings Tool



Server Settings Tool

The Server Settings tool enables you to set a variety of options for the VisualWorks Application Server, including:

- How to handle alert and error conditions
- The web browsers recognized for each look policy
- The maximum amount of memory to be used
- The actions to be taken when memory is stressed and the levels at which memory is considered stressed

Values that you set in the Server Settings tool override those saved in the server image, set by a configuration file, or entered on the command line at startup (see "Command Line" on page 1-4). Unless the settings are saved in the image or the configuration file, they are lost when the environment exits.

Navigation

To open the Server Settings tool, choose **Web → Open VisualWave Settings** in the Launcher window.

Options

The Server Settings tool is a notebook with a menu in the upper right corner that you use to choose the “page.” Initially, the **Alert Conditions** page is displayed. The pages roughly correspond to sections in the server configuration file.

Server Settings sections and options are described in “Server Environment Settings” on page B-1.

Commands

The following commands are common to all Server Settings pages:

OK

Cause the changes on the current page to take effect, and close the Settings Tool. All changes take effect immediately except Memory Sizes, which require that you save and restart the image.

Cancel

Cause any changes to be dismissed and the values reverted to the last applied values.

Apply

Cause the changes on the current page to take effect. All changes take effect immediately except Memory Sizes, which require that you save and restart the image.

Using the <Option> menu in the left-hand side of the window, you may also load and/or save settings to a file:

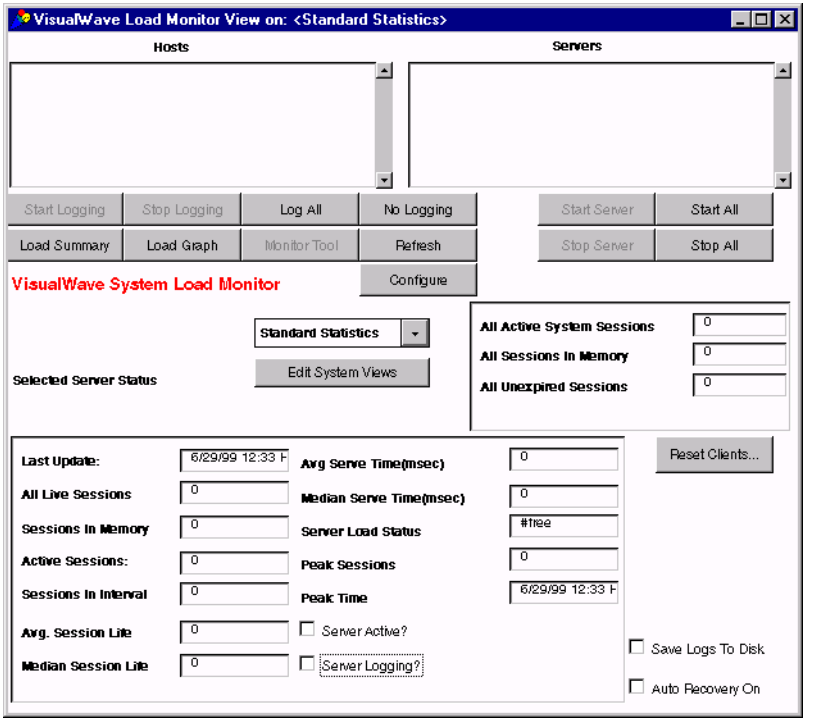
Save…

Write the entire collection of settings to a configuration file. You are prompted for a filename.

Load…

Read all of the settings from the configuration file that you specify.

Load Monitor Tool



Load Monitor Tool

The Load Monitor tool enables you to profile and control a group of hosts that have been configured in a server farm.

A list of hosts registered as secondary servers is displayed in the left-hand view. Selecting an individual host in the left-hand view displays all the servers running on that host in the right-hand view. Selecting a server displays more detailed statistics.

Navigation

To open the Load Monitor tool, choose **Tools→Load Monitor** in the Server Console.

Options

The Load Monitor tool contains a subcanvas which you can customize to display profile information of specific interest. Initially, the **Standard Statistics** page is displayed. Click on **Edit System Views** to open a tool for adding new subcanvas objects.

Save Logs To Disk
Record incoming statistical information in a file on the primary server (the file will be created in the current directory).

Auto Recovery On
Set the selected image to act as a backup to the primary server.

Is This Server Logging
Toggle logging on the selected secondary server.

Commands

Start Logging
If logging on the selected secondary server is offline, start it.

Stop Logging
If logging on the selected secondary server is running, stop it.

Log All
Enable logging on all secondary servers.

No Logging
Disable logging on all secondary servers.

System Summary
Open a window with a textual summary of activity on all hosts in the server farm.

Graph Summary
Open a window with a graphical summary of activity on all hosts in the server farm.

Monitor Tool
Open a Server Monitor tool for the selected secondary server.

Refresh
Refresh the current view to reflect the latest reported information from the server farm.

Configure
Open a Server Configuration tool for the selected secondary server.

Start Server
Start the selected (in the right-hand list) VisualWorks Server.

Stop Server
Stop the selected (in the right-hand list) VisualWorks server.

Start All
Start all VisualWorks servers on the selected (in the left-hand list) host.

Stop All

Stop all VisualWorks servers on the selected (in the left-hand list) host.

Edit System Views

Open a tool for installing custom subcanvas in the Load Monitor Tool.

Reset Clients…

If the primary server has become unavailable and been restarted, clicking on this button will notify all hosts in the server farm that a new primary server is available.

Statistics

The statistics reported by the secondary servers are summarized in the following table:

Name	Value
Last Update	Time of last update from server.
Live Sessions	Count of all sessions on the host that are unexpired.
Sessions in memory	Count of all sessions that exist in the host, expired or not.
Active Sessions	Count of all session active for reporting purposes.
Sessions in Interval	How many sessions have there been in the current sliding window.
Average Session Life	Mathematical average of session lifetime.
Median Session Life	Mathematical median session lifetime.
Average Time to serve	Mathematical average to serve a request from VisualWorks.
Median Time to serve	Mathematical median to serve a request from VisualWorks.
Server Load Status	Current system load level.
Peak Sessions	Highest number of sessions concurrently in the server.
Peak Session Time	When peak sessions occurred.
Server Active recently	Has the server served a request recently (see configuration settings).

Bookmark Manager

You can use the Bookmark Manager in the Application Server environment, just like you do in the VisualWorks Development environment.

To open the Bookmark Manager in the server environment, select **Tools → Bookmark Manager** in the Server Console.

- For more information about the Bookmark Manager, refer to the online *Web GUI Developer's Guide*.

B

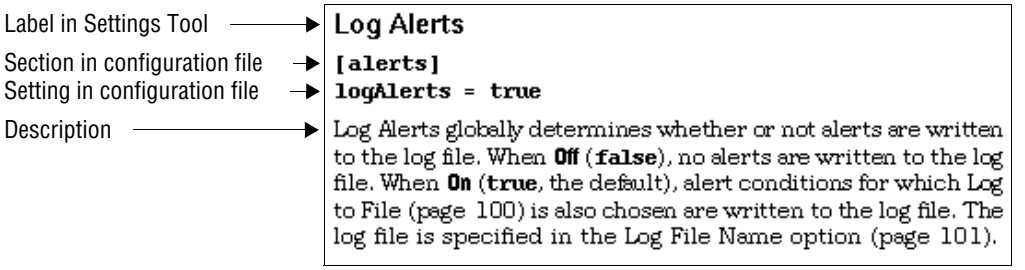
Server Environment Settings

This appendix describes the configuration options available for the VisualWorks Application Server. For each option, there is a system default stored in the server image. You can override the system default by setting options:

- In a server configuration file
- On the command line at startup
- In the Server Settings tool

Options are loaded from the sources above in the order listed — if an option is set in the Server Settings tool, it overrides the setting on the command line and in a configuration file. (For more information about these options, see "Configuring Environment Settings" on page 1-3.)

In this appendix, options are grouped by the "page" on which they appear in the Server Settings tool. The few options that are available only in the configuration file are listed at the end of the appendix. Each option is listed by its label in the Server Settings tool (when applicable). Syntax for the configuration file options appears below the option's heading in the **`courier`** typeface. A list of command-line shortcuts appears at the end of the appendix.



Alert Conditions

In general, an alert condition is any event that keeps the application user from getting the response he or she expected. VisualWave Server's response to various types of alert conditions can be configured in the Server Settings tool on the Alert Conditions and Application Error pages. Alert conditions appear in the **`[alerts]`** section of the server configuration file (for details on the location of this file, see "Configuration File" on page 1-3.)

Types of Alert Conditions

There are six types of alert conditions:

- Application Error
- Service Not Found
- Launch Refused
- Submit Refused
- Session Completed
- Session Expired

Application Error

An application error is any event that would have brought up a notifier in a screen-oriented application. For example, errors in application logic or execution are application errors.

Service Not Found

This alert occurs when the path specified in the URL cannot be resolved. It may occur for several reasons:

- The path prefix is not defined in the PathInfoPrefixResolver's list of paths and corresponding resolvers.
- The path prefix corresponds to a RegisteredSessionLauncher that does not include the path suffix in its registry of applications.
- The application specified does not exist in the server environment.

Launch Refused

This alert occurs when memory stress conditions cause the VisualWave Server environment to reject new web requests. Users that attempt to initiate new interactions with web applications are refused. The stress level at which launches are refused is determined by the **Defer Launch Requests** setting (see "Defer Launch Requests" on page B-10).

Submit Refused

This alert occurs when memory stress conditions cause the VisualWave Server environment to cease application interactions that are already in progress. The stress level at which submits are refused is determined by **Defer Submits** setting (see "Defer Submits" on page B-11).

Session Completed

This alert occurs when the application user terminates the application using whatever mechanism is provided by the application. Unlike other alerts, Session Completed is controlled by the user and is considered a normal part of application execution. It is included as an alert so that you can choose to respond to it with a message and/or log it in the standard log file.

Session Expired

This alert occurs when the session in which the interaction was handled has been inactive for some period of time and has been expired. The amount of time before an inactive session is expired is determined by the session resolver's **Poll for Inactive Sessions** and the **Session Timeout** values. These values are set in the Server Console.

Alert Condition Options

Log Alerts

```
[alerts]
logAlerts = true
```

Log Alerts globally determines whether or not alerts are written to the log file. When **Off** (**`false`**), no alerts are written to the log file. When **On** (**`true`**, the default), alert conditions for which Log to File (page 4) is also chosen are written to the log file. The log file is specified in the **Log File Name** option (see "Log File Name" on page B-5).

Display Alerts

```
[alerts]
displayAlerts = true
```

Display Alerts determines whether or not alerts are displayed in the Server Notifier window. When **Off** (**`false`**), no alerts are displayed in the Server Notifier. When **On** (**`true`**, the default), all types of alert conditions are displayed in the Server Notifier window.

Alert Type

In the Server Settings tool, this menu button determines the alert type for which the **Web Browser Message** and **Log to File** settings are displayed. (See page 2 for a description of the various Alert types.)

Web Browser Message (HTML)

```
[alerts]
applicationErrorMessage = 'HTML markup here'
serviceNotFoundMessage = 'HTML markup here'
launchRefusedMessage = 'HTML markup here'
submitRefusedMessage = 'HTML markup here'
goodbyeMessage = 'HTML markup here'
sessionExpiredMessage = 'HTML markup here'
```

The Web Browser Message is sent to the user's web browser when an alert is encountered. Each type of alert has its own Web Browser Message.

The Web Browser Message may contain any valid HTML markup. If you do not specify a message, the user's browser may display an error such as "**`Document contains no data.`**" To work around such browsers, you can include markup that does not display any text, such as **`<h1></h1>`**.

> **Note:** In the Server Settings tool, the Application Error message can be set on either the Alert Conditions page or the Application Errors page. When set on one page, the other page is updated automatically.

Log to File

```
[alerts]
applicationErrorLogAlert = true
serviceNotFoundLogAlert = true
launchRefusedLogAlert = true
submitRefusedLogAlert = true
goodbyeLogAlert = false
sessionExpiredLogAlert = true
```

This option works in conjunction with the **Log Alerts** option (see page 3) to determine whether or not alerts are written to the log file. If **Log Alerts** is Off (false), then no alerts are written to the log file, even if **Log to File** is chosen. If **Log Alerts** is **On** (**`true`**), then alert types for which **Log to File** is chosen (**`true`**) are written to the log file. Alert types for which **Log to File** is not chosen (**`false`**) are never written to the log file.

	Log Alerts On	Log Alerts Off
Log to File Chosen	logged	not logged
Log to File Not Chosen	not logged	not logged

The log file is specified in the **Log File Name** option, below.

Log File Name

```
[alerts]
logFileName = 'vwave.log'
```

The Log File Name determines the file into which alerts are written. There is only one log file. It is used by all servers and applications running in the server environment.

In the Server Settings tool, the **Log File Name** can be set on either the **Alert Conditions** page or the **Application Errors** page. When set on one page, the other page is updated automatically.

By default, the **Log File Name** is **`vwave.log`**. The filename is resolved with respect to the current working directory.

Application Errors

An application error is any condition that would have brought up a notifier in a screen-oriented application. For example, errors in application logic or execution are application errors.

Web Browser Message (HTML)

```
[alerts]
applicationErrorMessage = 'HTML markup text'
```

This Web Browser Message is sent to the user's web browser when an application error is encountered. This Web Browser Message is the same as the one displayed for Application Errors on the Alert Conditions page

(see page 4). When set on one page, the Web Browser Message is updated on the other automatically. There is only one entry in the configuration file.

Log to File

```
[alerts]
applicationErrorLogAlert = true
```

This Log to File option is the same as the one displayed for Application Errors on the Alert Conditions page (see page 4). When set on one page, the Log to File option on the other is updated automatically. There is only one entry in the configuration file.

Log File Name

```
[alerts]
logFileName = 'vwave.log'
```

This Log File Name option is the same as the one displayed for Application Errors on the Alert Conditions page (see page 5). When set on one page, the Log File Name option is updated on the other automatically. There is only one entry in the configuration file.

Show Message

```
[alerts]
applicationErrorShowMessage = true
```

The Show Message option determines whether or not the Web Browser Message is sent to the user's browser when an application error occurs. When checked (**`true`**), the Web Browser Message is displayed. When not checked (**`false`**), the Web Browser Message is not displayed. Application Errors are the only type of alert for which the Web Browser Message can be turned off.

Show Error

```
[alerts]
applicationErrorShowError = true
```

The Show Error option determines whether or not the error-notifier text is sent to the user's browser when an error occurs. The error-notifier text is what would have appeared in the top of the notifier window in a screen-oriented application. When checked (**`true`**), the error is displayed. When not checked (**`false`**), the error is not displayed, although the web browser message and stack trace may be (depending on the values of the Show Message and Show Stack options).

Show Stack

```
[alerts]
applicationErrorShowStack = false
```

The Show Stack option determines whether or not the stack trace is sent to the user's browser when an error occurs. When checked (**true**), the stack trace is displayed. When not checked (**false**), the stack trace is not displayed, although the web browser message and error may be (depending on the values of the Show Message and Show Error options).

Look Policies

Look policies determine how your applications appear in the user's web browser. The VisualWave Server environment has these look policy options:

- Show Layout Borders
- User_Agent Pattern for BasicHTMLLookPolicy and for EnhancedHTMLLookPolicy.

Show Layout Borders

```
[lookPolicy]
showLayoutBorders = false
```

The Show Layout Borders option determines whether or not the HTML forms that are sent to the web browser include table borders. (All HTML forms generated by VisualWave Server are laid out using tables.) Showing borders is useful when you are trying to debug or polish the way your application looks in the browser. In server environments that serve released applications, you probably do not want to show layout borders.

User_Agent Patterns

```
[lookPolicy]

BasicHTMLLookPolicyUserAgents =
  #('*EINet WinWeb 1.1*'
  '*Cyberjack Web 7*' ...)
  EnhancedHTMLLookPolicyUserAgents =
  #('*Mozilla/1.1*' '*Mozilla/1.2*' ...)
```

VisualWave Server automatically generates HTML that is appropriate for the user's web browser. VisualWave Server can generate HTML that includes tables (EnhancedHTMLLookPolicy) or HTML that does not include tables (BasicHTMLLookPolicy). VisualWave Server bases its HTML generation on the User_Agent in the web request.

The User_Agent Patterns options allows you to specify which look policy VisualWave Server should use for various kinds of web browsers. (The initial list is based on testing done by Cincom.) Wildcards are supported in this list:

- An asterisk (*) represents one or more characters.
- A pound sign (#) represents any single character.

In the Server Settings tool, each pattern should be on a separate line. In the configuration file, each pattern should be enclosed in single quotation marks as a literal in an array (Smalltalk syntax).

If the user's web browser is not listed in either the BasicHTMLLookPolicy or EnhancedHTMLLookPolicy options, VisualWave Server uses EnhancedHTMLLookPolicy.

Memory Configuration

Initial Memory Size

```
[memoryPolicy]
minimumMemory = 16000000
```

This option determines the amount of memory (in bytes) that the VisualWave Server environment uses at startup. This is also the minimal amount of memory that the VisualWave Server environment will use while it is running.

Maximum Memory Size

```
[memoryPolicy]
maximumMemory = 64000000
```

This option determines the maximum about of memory (in bytes) that the VisualWave Server environment will use. As memory use approaches this limit, the VisualWave Server environment begins to take actions to reduce the load, based on the Memory Stress Level and Server Actions groups of options.

Server Actions

There are four actions that VisualWave Server environment can take when it begins to approach the Maximum Memory Size:

- Purge Cached Resources
- Aggressive Session Expiration

- Defer Launch Requests
- Defer Submits

For each of these, you can set the memory stress level at which the server environment will take action. There are five memory stress levels:

1. Minimum
2. Light
3. Medium
4. Heavy
5. Maximum

(The exact percentage of memory that corresponds to each of these is determined by the Memory Stress Percentages settings, described on page 11.)

When setting the memory stress levels for these actions, consider which action you want the server environment to take first and set that action to have the lowest stress level of the group. Similarly, consider the order in which you want the other actions to occur and set their stress levels accordingly.

Purge Cached Resources

```
[memoryPolicy]
purgeStressLevel = 1
```

The server environment caches resources such as generated GIF images so that they can be reused by applications. The Purge Cached Resources action causes the server environment to clear out such resources. When resources are purged, applications must generate new GIF images (for Smalltalk images or buttons whose boundaries are reinforced with GIFs) every time they are used.

By default, resources are purged before any other actions are taken. They are purged when memory reaches the minimum stress level (1).

Aggressive Session Expiration

```
[memoryPolicy]
expireStressLevel = 2
```

Session resolvers keep track of sessions and expire those that have been inactive for some period of time. Each session resolver has two session timeout periods: one for normal activity and one for stressed conditions. The Aggressive Session Expiration action allows you to set the memory level that is considered “stressed” conditions.

By default, sessions are aggressively expired after cached resources are purged and before launches or submits are deferred. They are aggressively expired when memory reaches the light stress level (2).

If most of the applications that you serve require long periods of inactivity while the user performs some actions in or analyzes material presented in the browser (or your applications are memory intensive) then you may want to set this to a higher value. If Aggressive Session Expiration is higher than Defer Launch Requests, user interactions will be short, but a larger number of users will be able to initiate interactions with the server environment.

If a user tries to interact with an expired session, a Session Expired alert condition occurs. “Alert Conditions” on page B-2

Defer Launch Requests

```
[memoryPolicy]
deferLaunchStressLevel = 3
```

A launch request is the initiation of a new interaction with the server environment. It can be a simple request such as an echo or a request to start an application.

By default, launches are deferred after resources are purged and sessions are aggressively expired. They are purged when memory reaches the medium stress level (3).

If you want to allow as many users as possible to access your server environment at the possible cost of performance, set Defer Launch Requests to a high stress level. If you want to improve performance at the possible cost of refusing some connections, set Defer Launch Requests to a lower stress level.

When launches are deferred, requests made to the server environment raise a Launch Refused alert condition. “Alert Conditions” on page B-2

Defer Submits

```
[memoryPolicy]
deferSubmitStressLevel = 4
```

A submit is the continuation of an interaction with the server environment. Submits occur when the user responds to an HTML form that was generated and sent by an application in the server environment.

By default, submits are deferred after all other actions have been taken. They are deferred when memory reaches the heavy stress level (4).

When submits are deferred, submits made to the server environment raise a Submit Refused alert condition. “Alert Conditions” on page B-2

Memory Stress Levels

The VisualWave Server environment can take action at any of five memory stress levels: Minimum, Light, Medium, Heavy and Maximum.

The exact percentage of memory that corresponds to each of these is determined by the Memory Stress Percentages settings.

Percentages are based on the setting for Maximum Memory Size (page 8). The percentage of memory should be the lowest for the minimum stress level (1) and highest for the maximum stress level (5). The defaults are shown here:

```
[memoryPolicy]
stress1Percentage = 80
stress2Percentage = 90
stress3Percentage = 94
stress4Percentage = 97
stress5Percentage = 99
```

Memory Sizes

The Memory Sizes options allow you to customize how object memory is allocated at startup. You can set the amount of memory for an area to be a percentage of the value in the leftmost column by moving the slider in the middle column. For information about allocating object memory, see the *VisualWorks Application Developer's Guide*. The defaults are shown here:

```
[memorySize]
largeMultiplier = 1.0
survivorMultiplier = 1.0
cacheMultiplier = 1.0
headroomMultiplier = 1.0
stackMultiplier = 1.0
edenMultiplier = 1.0
```

Memory Sizes options are saved in the configuration file. Note that because they affect object memory, they must be saved in the server image as well. Changes to Memory Size options do not take effect until you save and restart the image.

Time Zones

VisualWave Server allows you to choose and customize the time zone in which the server environment is running. Time stamps are included in all HTML information sent to web browsers and used by the web browsers to determine whether to use the new information or cached information.

Time Zone Information

```
[timeZone]
zoneName = usaPacific
```

You can choose one of several preset time zones or choose to create a custom time zone:

Server Settings menu item	Configuration File value
Japan +9	`japan`
Germany +1	`germany`
UK; Greenwich 0	`ukGreenwich`
USA; Eastern -5	`usaEastern`
USA; Central -6	`usaCentral`
USA; Mountain -7	`usaMountain`
USA; Pacific -8	`usaPacific`
Custom	`custom`

Customization

The Server Settings tool displays the characteristics for the selected time zone, including difference from Greenwich Mean Time and the Daylight Savings Time specifications. You can edit these characteristics to create a custom time zone. All configuration file options are part of the section `[timeZone]`.

Server Settings field	Configuration File syntax
Difference from GMT	`timeDifferenceFromGmt = -8`
DST Offset	`dstOffset = 1`
DST Start Hour	`dstStartHour = 2`
DST start day (menu)	`dstStartWeekday = Sunday`
DST start day number	`dstStartDayOfYear = 97`
DST end day number	`dstEndDayOfYear = 304`

Inline Web Images

Dynamic GIF-Compatible Image Encoding

```
[gifEncoding]
preferredEncoderClassName = GUFEncoder
```

If an application contains Smalltalk graphic images, VisualWave Server renders those images as either GIF-compatible images or as Java-rendered graphics. VisualWave Server also generates GIF or Java images for a variety of widgets that do not have HTML equivalents, such as including menu bars and action buttons with graphical labels.

VisualWave Server can generate three kinds of images:

- True GIF images with LZW compression (GIFEncoder)
- Uncompressed GIF-compatible “GUF” images (GUFEncoder)
- Java-rendered graphics (jrenderer)

By default, VisualWave Server generates GUF images. All browsers that support GIF images should also display GUF images. GUF images are larger than the equivalent GIF images and, as a result, load more slowly. However, they are not subject to the patent licensing issues related to GIF.

To enable true GIF compression, you must load the **`VisualWaveGIFWithLZW.pcl`** parcel into the server image. This parcel is distributed with the VisualWave Developer environment, and should be located in its release directory (typically, **`wavedev`**.) Before installing the supporting code, the GIF image licensing agreement is displayed. You must read and agree to the licensing agreement before completing the load operation.

> **Warning:** Before you generate true GIF images, read the license information displayed by the Server Settings tool.

To generate compressed GIF images, choose **True GIFs** in the Server Settings or set **`preferredEncoderClassName`** to **`GIFEncoder`**.To generate Java-rendered images, choose **Java Renderer** or set preferredEncoderClassName to **`javarenderer`**.

Default File Names

VisualWave Server recognizes two file name options:

- Image file name
- Configuration file name

Default values are saved in the server environment image; non-default values can be set in the configuration file.

Image File Name

```
[default]
imageFileName = 'c:\vwserver\image\vwserver.im'
```

The Image File Name option is included for reference only. The image file name is always read from the command line at startup.

When you save options from the Server Settings tool, the server environment includes the pathname for the current image as the **imageFileName** in the configuration file. You do not need to use the configuration file with the named image; the value is for reference only.

Configuration File Name

```
[default]
configFileName = 'vwave.ini'
```

The Configuration File Name option specifies the name of a configuration file to read at startup. The system default is **vwave.ini** for Windows NT and **.vwaverc** for UNIX. You can change the system default:

1 In the Server Settings tool, click the **Save All Settings…** button.

2 When prompted for a filename, enter the name you want to use as the new system default.

3 Save the server environment.

When you restart the server environment, it looks for the new system default configuration file.

To specify a non-default configuration file to read at startup, set the **configFileName** option on the command line:

```
vwave vwserver.im [default]
  configFileName='.vwavecfg' &
```

Note: At startup, VisualWave Server always looks for the system default configuration file. If found, those options are always loaded first. If you specify a configuration file on the command line, that file is loaded after the system default file.

When used within a configuration file, the **`configFileName`** option acts as an "include," allowing you to chain together several configuration files. Be careful not to create loops between configuration files.

Oracle Configuration

Oracle configuration is done either by editing the configuration file or with command line options. There are no Oracle settings in the configuration tool. Database settings for Sybase configuration are made in a similar fashion.

To add Oracle settings, add a section to the configuration file headed:

```
[oracleInterface]
```

Oracle DLL Name

```
[oracleInterface]
ntLibraryFile = 'ora73.dll'
```

On Windows NT systems, you specify the Oracle DLL by setting the **`ntLibraryFile`** option. This option is relevant only if you use the VisualWorks Database Connect for Oracle7 on a Windows NT system.

The default Oracle DLL name in VisualWave Server is **`ora73.dll`**.

No path information is required if the Oracle DLL is located somewhere on the standard path (e.g., **`\WINDOWS`**). For special Oracle installations, you may need to specify additional path information.

Non-blocking Oracle Settings

VisualWave Server supports non-blocking access to Oracle databases via the VisualWorks Threaded Interconnect API (THAPI). Using THAPI, calls are made directly to the Oracle library. Refer to the VisualWorks Database documentation for details on using the non-blocking calls with the Oracle library. The RPC mechanism has been superseded by the new THAPI mechanism.

Command-Line Shortcuts

Any option that can be set in the configuration file can also be set on the command line at startup. To specify an option on the command line, include the section name, option name, and value. For example, to set the Configuration File Name, use:

```
vwave vwserver.im [default] configFileName =
'.vwaverc' &
```

Some options also have shortcuts. For example, the shortcut **-f** can be used for the **configFileName** option above:

```
vwave vwserver.im -f '.vwaverc' &
```

The following table contains a complete list of command-line shortcuts for configuration options. Command-line options to load parcel (**-pcl**) and parcel configuration files (**-cnf**) are also accepted.

Command-line Shortcuts

Shortcut	Section	Option Name
-f	default	configFileName
-b	lookPolicy	showLayoutBorders
-log	alerts	logFileName
-m	memory	maximumMemory
-max	memory	maximumMemory
-min	memory	minimumMemory
-orantdll	oracleInterface	ntLibraryFile

Index

Z