Enhydra Multiserver

Administration Guide

version 2.0

Developed by Lutris Technologies, Inc.

CHAPTER 1

Overview of the Enhydra Multiserver

Overview

The Enhydra Multiserver consists of the Enhydra Multiserver Runtime Environment (Enhydra/rt) and the Enhydra Multiserver Development Environment (Enhydra/dev). These products allow developers to easily construct, execute and manage multi-tiered web applications and servlets.

This guide focuses on the administration and deployment of web applications and servlets with the Enhydra/rt product. A separate developer guide covers the Enhydra/dev product.

Enhydra Multiserver features include:

- 100% Java implementation
- Stand-alone HTTP Web server capability
- Authentication through *http-basic-auth* protocol
- Support for servlets
- Support for Enhydra web applications
- Ability to connect to Netscape web server through WAI (Web Application Interface)

- Ability to connect to remote web server through RMI
- Ability to connect to remote web server through CGI gateway
- Multiple URL to application mapping
- Built-in servlet and application administration console
- Built-in HTTP debugger console
- Customizable logging facility
- Application partitioning within Java Virtual Machine (multiple class loaders)
- Database manager for object to relational database mapping and JDBC connection management
- Sophisticated session (client state) management
- Extensible filter mechanism for manipulating HTTP requests
- Encapsulation of multi-tiered Java applications and resources into a single, easy to manage Java “jar” file
- Runs on all Unix systems; Windows 95, 98 & NT
- Scales from hand held devices through to clustered server environments

What can the Enhydra/rt do for me?

Enhydra Multiserver is a 100% Java application server for web based applications. It can operate as a self-contained Web server, a servlet runner or as a sophisticated web application server. It is able to operate stand-alone or in conjunction with virtually any web server.

Enhydra Multiserver can multiplex requests from a number of separate sources and direct them to hosted applications or servlets. For example, it would be possible to run an application and simultaneously allow access directly through http on a chosen port, through a Netscape web server over WAI and through an Apache web server via CGI gateway. Each method of access to an application can be individually administered, manipulated or debugged.

Enhydra Multiserver can also be used to give any web server a servlet running capability.

Enhydra Multiserver can scale from a small footprint Java web server for hand-held devices to a fully clustered enterprise strength application server.

The fact that the Enhydra Multiserver offers such a vast array of options and features makes it look a little daunting at first. However, by providing an overview of the operation of the server and describing each facility in turn, we hope to show that operating the Enhydra Multiserver is actually a simple exercise.

Servlets & Applications

What is a Servlet?

A natural question here might be "What is a servlet?" A servlet enhances an HTTP server by enabling request/response services. When a client sends a request to the Web server, the Web server can send the request information to a servlet and have the servlet construct the response that the server sends back to the client. In addition, a servlet can be loaded automatically when the Web server starts or it can be loaded the first time a client requests its services. After loading, a servlet continues to run, waiting for additional client requests.

Why is this significant? Simply put, it allows you to leverage your Web server to take advantage of servlets--a Java class that dynamically extends your server's function.

Servlets provide enormous advantages:

- Servlets run in the context of the Enhydra/rt server process. This saves the time and effort to create a new CGI process to serve an incoming request. Running servlets in the Enhydra/rt environment allows them to be dynamically managed and monitored.
- Servlets are written in Java, a stable, secure and easy to maintain language.
- Servlets let you create complicated, high-performance, cross-platform Web applications without having to create multiple versions of them. You can use Enhydra/rt as a stand-alone server or with your current server to take advantage of Java's "write-once, use everywhere" capabilities.

Servlets are not specific to the Enhydra Multiserver. The Servlet API is a standard extension to the Java Platform and a number of companies have implemented the Servlet API.

What is an Enhydra application?

In the context of this discussion, an Enhydra application is a special kind of servlet[1]. While a servlet offers the ability to extend a web server’s functionality, it is difficult to write large and complex web applications as a collection of individual servlets. The Enhydra/dev product provides a way to design and build a complete application as a single servlet. This “super-servlet” controls its own loading of classes, caching of resources & management of session information. It provides a way to build a robust multi-tiered application that is easy to deploy and easy to maintain.

With a traditional all-servlet approach to building an application, each dynamic web page is built as a separate servlet. With an Enhydra application, each dynamic page is built as a presentation object. These objects are easier to write since they need not be thread-safe (threading is handled internally), they may be written with a patent-pending page compilation technology called Enhydra Jolt and may be monitored and managed with the built-in administration and debugging console.

Enhydra applications may also interact with other 3rd party servlets hosted by the same Enhydra/rt. An example of this would be a shopping application that utilizes an off-the-shelf credit card authentication and payment servlet.

Multi-tiered applications

The Enhydra/dev guide encourages the development of 3-tier Enhydra applications. A summary of these layers is included here:

- **Presentation Objects** contain the logic that presents information to an external source and obtains input from that source. The presentation logic generally provides menus of options to allow the user to

---

[1] It is possible to execute Enhydra applications in any servlet runner environment, not just on the Enhydra multiserver.

navigate through the different parts of the application, and it manipulates the input and output fields on the display device. Frequently the presentation component also performs a limited amount of input data validation.

- **Business Objects** contain the application logic that governs the business function and process. Business objects are invoked either by a presentation component when a user requests an option, or by another business function. The business functions generally perform some type of data manipulation.
- **Data Objects** contain the logic that interfaces with a data storage system, such as database systems or hierarchical file systems, or with some other type of external data source such as a data feed or an external application system. Business objects invoke Data Objects to save persistent state.

In Multi-tiered applications, Servlets generally serve in the middle tier to implement business logic. Because of the static nature of html interfaces used by web browsers, it is impossible for dynamic displays to be implemented using standard HTML. The Enhydra multiserver introduces the notion of presentation objects to solve this problem. Presentation objects allow information to change dynamically on a web page. An Enhydra Application is a servlet that contains presentation objects.

Database Connectivity

Enhydra/rt uses JDBC (Java Database Connectivity API) to connect to relational databases. JDBC provides a level of abstraction so the actual database engine (e.g. Oracle, Informix) will be invisible to the application developer. JDBC is one of the standard enterprise Java APIs and has been adopted by all major database vendors. Enhydra/rt currently supports JDBC level 3 and 4 drivers with Informix, Oracle, Sybase, MSQL, or other standard implementations of JDBC. Enhydra/rt also offers sophisticated object to relational mapping technology and database connection pool management.

Enhydra/rt configuration files

The Enhydra Multiserver configuration files are described in detail in following chapters. It is only important at this stage to know that there are two types of configuration file:

- one file that contains the configuration information for the Enhydra/rt as a whole,
- another that is specific to each Enhydra application.

In the following discussion of connection methods, channels and filters all configuration choices are made in the Enhydra/rt configuration file[1] which is installed into /usr/local/enhydra/multiserver.conf by default.

The Enhydra Multiserver will generate a log file where error, warning or where debugging messages are written. This log file can track many different levels of information from sophisticated debugging messages and stack back traces, environmental warnings, server errors. In the event of errors or problems this trace file should be consulted for information. By default, the log file is located in /usr/local/enhydra/log/multiserver.log.

Connection Methods

Because of the large number of web servers deployed in the enterprise, Enhydra/rt provides a number of options for connecting to web servers. Enhydra/rt connects to web servers by WAI, RMI, or CGI-gateway. Enhydra/rt can function as a stand-alone web server directly accepting HTTP requests and offering standard web server functionality (e.g. http-basic-auth.)

Each method of listening for client requests is called a *connection method*.

When Enhydra/rt is accepting http requests directly it is using the “HTTP connection method”, when it is accepting requests from another web server, it is using one of “WAI, RMI or CGI connection method”.

---

1. All configuration can also be achieved through the Enhydra Multiserver administration console application.

HTTP

Setting the Enhydra/rt server to listen directly for HTTP requests is the easiest of the connection methods. It simply involves adding the port you wish to run the http server on to the ***multiserver.conf*** configuration file. Typically http servers listen on port 80. It is not necessary to run your http server on this port, although to avoid company firewall problems it may be necessary. Note that all TCP/IP ports below 1024 are specially controlled by the operating system, thus you will need to be a privileged user (*root* on UNIX systems). It is recommended that while experimenting with the Enhydra Multiserver that you run it over HTTP on a non-privileged port (i.e. port greater than 1024).

WAI

When using the Enhydra Multiserver in conjunction with Netscape web servers you may elect to connect via WAI. To do this the ***multiserver.conf*** file must define a WAI connection method, the fully specified DNS name or IP address of the server and a port number. The web server must have WAI support installed and enabled. All requests through WAI must begin with *"http://<server>/iiop/…"* as defined in the WAI documentation. Note that the Netscape web server can be configured to remap URLs to remove the *iiop* prefix.

RMI

The Enhydra Multiserver can operate as a separate process from any existing web server that supports the servlet interface. In this scenario, The Enhydra Multiserver must be configured to register under the chosen name with the RMI registry (which must be running and is part of the Java runtime environment). The Enhydra Multiserver RMI relay servlet must then be loaded on the web server to forward requests (via RMI) to the Enhydra Multiserver.

CGI

The Enhydra Multiserver CGI gateway binary may be used to relay requests from virtually any existing web server. The Enhydra Multiserver must be configured with the port that the CGI gateway communicates on (configurable) and the CGI binary installed and enabled on the web server. In this connection method, the client's request causes the CGI binary to be executed; this then forwards the request to the Enhydra Multiserver on the configured port. The response is passed back to the CGI binary that in turn passes it back to the web server.

The CGI binary is available for many different platforms, each of which has a slightly different setup procedure.

Channels

The previous section described connection methods – the ways that the Enhydra Multiserver can listen for requests. This does not describe how requests that have been received are passed to individual servlets or applications – this is the purpose of *channels*.

Each channel defines a route between a connection method and an application or servlet. Each servlet or application may have multiple channels. Each channel defines a connection method, a servlet or application and a URL prefix. When a request enters the Enhydra Multiserver it is sent down the channel that matches the selection criteria.

The channel can be considered a pipe through which requests can be sent to an application or servlet. As such, they can be independently enabled or disabled, added or removed without effecting the running application (other than to prevent access through themselves). They allow offer the perfect opportunity get access to the request or response in between the Enhydra Multiserver and the application. The defined filter API offers just that opportunity.

Figure 1 illustrates the role of connection methods, channels and introduces filters within the Enhydra Multiserver.

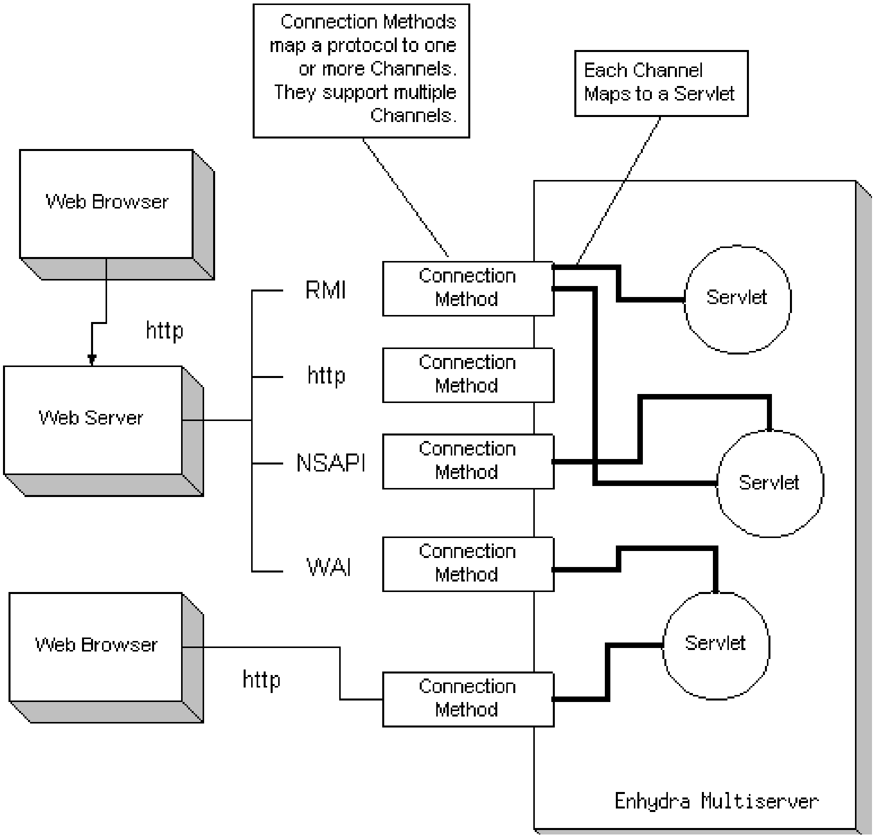


FIGURE 1. Connection methods, channels and filters in the Enhydra Multiserver

Filters

Filters log and debug requests and responses to and from servlets. Filters wrap themselves around servlet requests and responses by inserting themselves into a channel. Normally the design filters is such that they do not modify the behavior of the servlet. If a filter did not pass a call through to the original object, it would break the behavior of the servlet. Normally filters simply take note of the call, perhaps logging it to a file, perhaps printing a message to an OutputStream, then pass the call on.

The fact that filters do not change the behavior of the Servlet means a single servlet may have multiple filters associated with it. In addition, the multiple filters will be unaware of each other. If developer produces a filter that is not transparent, an administrator must be very careful of the order filters are composed. An example of a non-transparent filter is one that logs the calls to a servlet, but does not pass any calls on to the actual Servlet. Clearly this filter would need to be the first filter applied, so it is wrapped closest to the real servlet. Any filters between this hypothetical filter and the servlet would never be called.

Filters can be added or removed without disturbing a running application or servlet.

There are two standard filters supplied with Enhydra/rt:

Logging Filter

The Standard logging filter is used to generate a web server style log file. The name and location of the output file can be configured and the inclusion of this filter enables immediate logging on that channel. See the chapter on the ***multiserver.conf*** configuration file for more details on how to enable this manually, or read the administration chapter for details on how to added the filter via a web interface.

Debugging Filter

The administrator does not normally add the debugging filter. It is added automatically by the Enhydra Multiserver administration console whenever a particular application is being debugged. The debugger admirably demonstrates the capabilities of the channel filter concept.

Adding your own filter

To add your own filter you need to perform several tasks (a complete description of these is available in the Enhydra developers guild):

- Write a filter. The simplest method is to extend an example base filter defined by Enhydra/dev.
- Add the filter specifications to the ***multiserver.conf*** configuration

file (see chapter on configuration file format).

- Insert the filter into any required channel by either using the Enhydra Multiserver administration console application or by editing the channel specifications in the Enhydra Multiserver configuration file.

NOTE: The confusing concepts of connection methods and channels are abstracted when using the Enhydra Multiserver Administration Console. All you need to do is specify every way you want to connect of each application or servlet. The creation of appropriate connection methods and channel is made automatically and the Enhydra/rt configuration file is automatically updated. Understanding the inner workings of Enhydra/rt will help the administrator locate and fix problems. Please refer to Chapter 2 which covers the Enhydra Multiserver Administration Console.

Classpath Operation

Since the Enhydra Multiserver and all applications run by it are written in Java, it is necessary to correctly understand and configure classpath.

Any Java application is written as a set of classes. Each time a previously unreferenced class is accessed, the Java Virtual Machine (JVM) automatically loads the class through the class loader. For the class loader to be able to locate the class file, it looks in a sequential list of directories or archives (jar or zip files). This sequential list is know as the classpath

In the Enhydra Multiserver each application receives its own class loader. This prevents applications getting confused over class ownership and allows applications to run autonomously. When an Enhydra application or servlet requires a class the following sequence of events takes place:

1. The applications class loader tries to find the class in its “Additional Classpath” directories or archives. If found then a private version of that class is loaded for the application.

2. If not found, then the class loading responsibility is passed to the system class loader. This class loader looks in all directories and archives defined in the system classpath. This is a combination of the default Java classpath and those added by the Enhydra Multiserver start script[1]. Any class loaded by the system class loader is shared by all applications and therefore cannot be easy manipulated.

> **NOTE:** Some JDBC drivers can only be loaded by the system class loader. If you suspect JDBC problems then try adding the location of the JDBC classes to the system classpath and removing them from the application classpath.

Appendix A: Anatomy of an Application

There are three stages in the life of an application: development of the application, deployment of the application, and the management of the application. With the Enhydra product line, the lifecycle of an application is well-defined and easy to implement. The following figure illustrates the lifecycle of an Enhydra application.

---

[1] This is typically defined by the CLASSPATH environment variable.

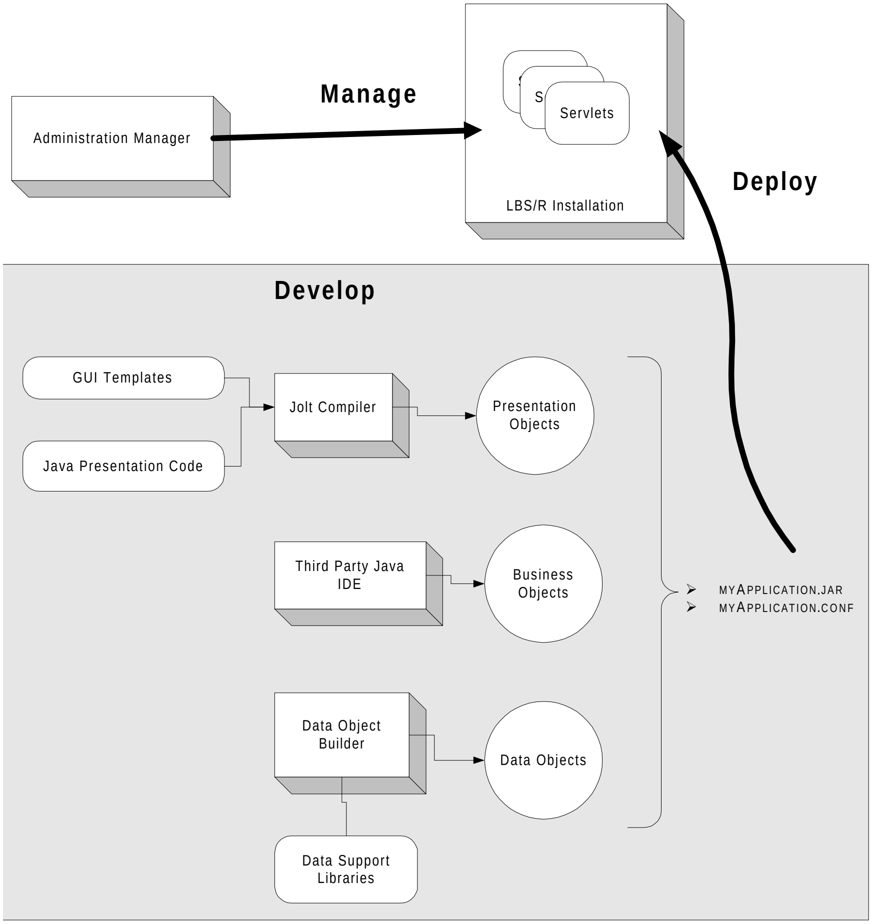


FIGURE 2. Lifecycle of an Enhydra Application

Develop

An Enhydra application has three tiers of logic: presentation, business, and data. Enhydra/dev provides tools to help construct objects for these layers as well as integrating with best-of-breed tools for other aspects of application development.

The presentation tier typically consists of a number of HTML pages generated with any number of page layout tools. The HTML can embed JOLT tags or fields in the HTML. Since the JOLT elements are valid HTML fields, page layout tools will not take exception to their presence. Enhydra/dev also provides templates for illustrating for developers some models for mixing HTML and JOLT.

The business tier typically consists entirely of Java code and encapsulates the actual business logic of the application. Other than our recommended source tree layout and group development tools, Enhydra/dev provides no other functionality for this layer. It is expected that a Java IDE will be used to develop business objects.

The data tier encapsulates the persistent data needed by the business tier. The data needed by the business logic is usually stored in a relational database. Enhydra provides the database manager classes for retrieving and querying that data. Enhydra/dev also provides a set of libraries and a tool built on that library to ease the mapping of relational data into objects.

Deploy

An Enhydra application consists of two files. A JAR file that contains all the objects that make up the application and any additional resources and a configuration file that contains information relevant to the start-up and resource parameters necessary for the application. Placing these files in a well-known location to an instance of Enhydra/rt, is the deployment model for servlets and Enhydra applications.

By using the Enhydra development tools and methodology, it is simple to generate the files that comprise an Enhydra application. It is not imperative that you use the tools supplied by Enhydra, but it makes the process of generating the deployment files very easy.

Manage

Once an application deploys to an installation of Enhydra/rt, the administration manager then controls the state of that application. The administration manager adds the application into the Enhydra/rt installation, modifies its resource parameters (e.g. database connection pool size), and controls the lifecycle of the Enhydra application.

CHAPTER 2

The Enhydra Multiserver Management Console

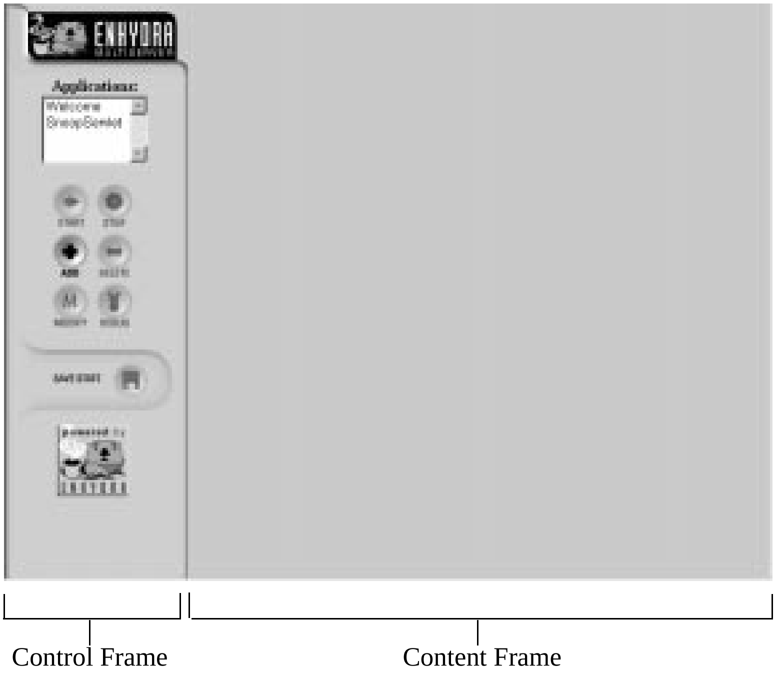
Once you have started the Enhydra Multiserver, the next step is to access the Management Console. Using the Management Console, you can start, stop, add, delete, and modify your applications and servlets.

To access the Management Console:

- Launch your default web browser. The Enhydra Multiserver Management Console is only supported on Netscape Navigator/Communicator version 3.0 and higher; or Microsoft Internet Explorer version 4.0 and higher.
- Open the URL `<http://<machine_name>:8001>`. Note that `<machine_name>` refers to the name of the server where the Enhydra Multiserver was started. This is the default URL and may be modified by the user. Please see *Chapter 3: The Enhydra Multiserver Configuration File Format* for more information.
- You will be prompted for a username and password. By default, the username is “admin” and the password is “enhydra”. To modify these values, please see *Chapter 3: The Enhydra Multiserver Configuration File Format.*
- At this time, your browser will connect to the Enhydra Multiserver Management Console and provide access to the Console

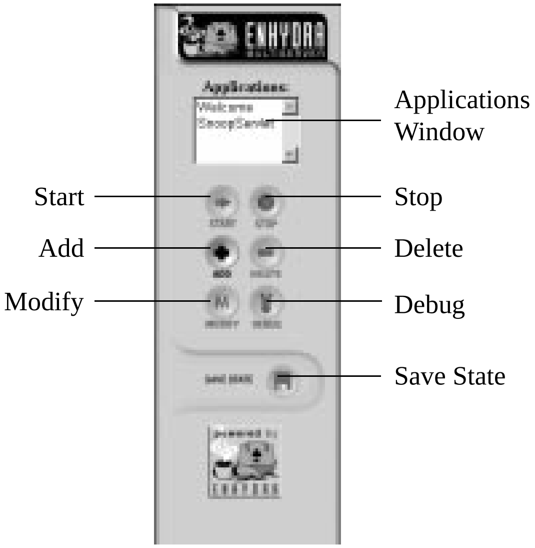
Tools.

The Management Console is divided into two main frames: the Control Frame and the Content Frame.



During your use of the Management Console, some tools will cause additional pop-up windows to appear. These windows will disappear when the "OK" or "Cancel" button is pressed within the window. Similarly, when accessing the Help button from any point in the Management Console, or when accessing another URL, an additional browser window will launch. These windows must be closed manually.

The Control Frame contains the Enhydra Multiserver Management Console Tools, which are used to select, start, stop, add, delete, modify, or debug an application or servlet. There is also a control for saving the current state of the server.



NOTE: *Before selecting an installed application or a servlet, only the "Add" tool is available. The remaining tools are grayed-out until an application or servlet is selected.*

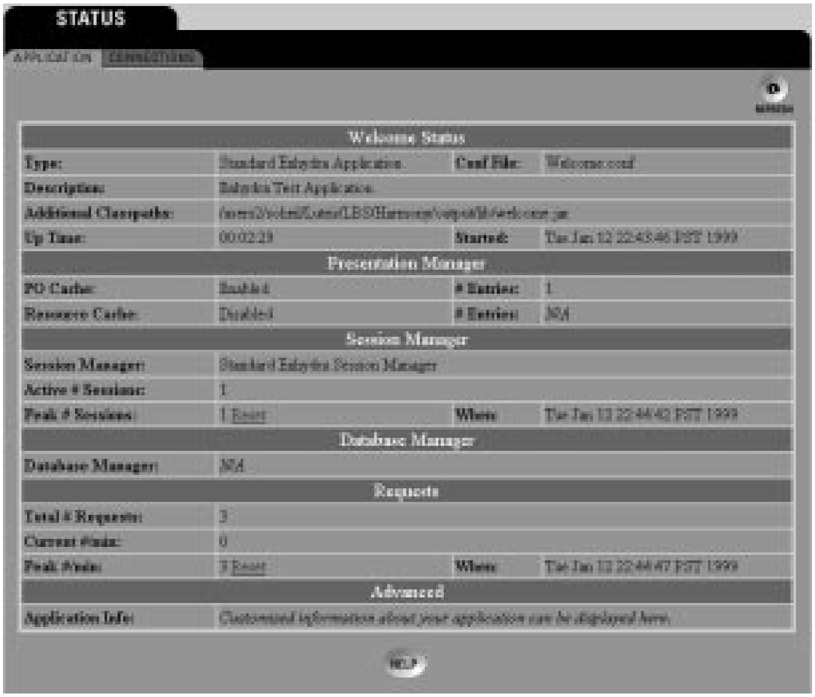
The Applications Window

The Applications Window, located at the top of the Control Frame in the Management Console, contains a list of all applications and servlets that have been added to the Enhydra Multiserver. Upon initial start-up of your server, the pre-installed "Welcome”, and “DemoCart” applications appear in this window as the only installed applications. Use the “Add” button in the Controls Frame to add applications or servlets to the Applications Window.

To manage an installed application or servlet, select its name in the Applications Window. Doing so will refresh your browser and display status information for the selected application or servlet in the Content Frame. Also, any of the Console Tools that depict valid operations for that application or servlet will be activated at this time. The next section describes the specifics of the application/servlet information that is displayed on selecting an application or servlet in this manner.

The Application/Servlet Status Display

The status display provides basic information for the selected application or servlet, such as the number of requests (if the servlet or application is currently running), requests per minute and uptime. There are controls provided for refreshing the current screen, and getting context sensitive help (currently disabled).



Welcome Status			
Type:	Standard Enhydra Application	**Conf File:**	Welcome.conf
Description:	Enhydra Test Application		
Additional Classpaths:	[illegible]		
Up Time:	00:02:23	**Started:**	Tue Jan 12 22:43:46 PST 1999
Presentation Manager			
PO Cache:	Enabled	**# Entries:**	1
Resource Cache:	Disabled	**# Entries:**	N/A
Session Manager			
Session Manager:	Standard Enhydra Session Manager		
Active # Sessions:	1		
Peak # Sessions:	1 Reset	**When:**	Tue Jan 12 22:44:42 PST 1999
Database Manager			
Database Manager:	N/A		
Requests			
Total # Requests:	3		
Current #/min:	0		
Peak #/min:	3 Reset	**When:**	Tue Jan 12 22:44:47 PST 1999
Advanced			
Application Info:	*Customized information about your application can be displayed here.*		

The following tables represent the standard fields in the application status display.

TABLE 1. Application Status -- General Information

Field Name	*Content*
Type	Type of application, usually "Standard Enhydra Application". If you have added a custom application, its type will be displayed in this field.
Conf File	The name of the configuration file, found in `server.confDir`. For example, "`Welcome.conf`"
Description	A modifiable description of the application.
Additional Classpaths	Location of additional classpaths necessary for the successful instantiation of this application. For example, `/usr/local/enhydra/`.
Up Time	The elapsed time since application start-up. If the application has not been started, "Not running" will appear in this field.
Started	The time of the most recent application start-up. If the application has not been started, "Not running" will appear in this field.

TABLE 2. Application Status -- Presentation Information

Field Name	*Content*
PO Cache	Indicates whether presentation objects are being cached or not. Values are “enabled” or “disabled”.
# Entries	Refers to the number of entries currently in the PO cache.
Resource Cache	Indicates whether resources are being cached or not. Values are “enabled” or “disabled”.

TABLE 3. Application Status -- Session Information

Field Name	*Content*
Session Manager	Type of session manager in use, usually "Standard Enhydra Session Manager". If you have installed a custom session manager, its type will be displayed. If there are no active sessions, “N/A” will be displayed in this field.
Active # Sessions	Number of currently active sessions. If there are no active sessions, “N/A” will be displayed in this field.
Peak # Sessions	Peak number of sessions achieved by this application during its current instantiation. If there are no active sessions, “N/A” will be displayed in this field. A “Reset” link is provided here to allow for resetting this value to zero.
When	The time and date when the peak number of sessions occurred.

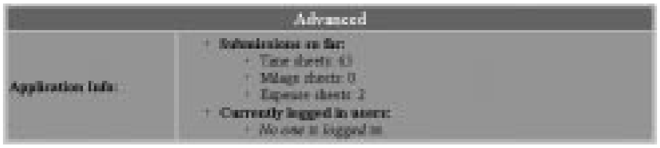
TABLE 4. Application Status -- Database Information

Field	*Content*
Database Manager	Type of database manager in use, usually "Standard Enhydra Database Manager". If you have installed a custom database manager, its type will be displayed. If the application does not require the use of the database manager, “N/A” will be displayed in this field.
Active # Connections	Number of active connections to the database manager, during the application's current instantiation. If the application does not require the use of the database manager, this field will not be displayed.
Peak # Connections:	Peak number of connections to the database manager during the application's current instantiation. If the application does not require the use of the database manager, this field will not be displayed. A “Reset” link is provided here to allow for resetting this value to zero.
When	The time and date when the peak number of connections occurred. If the application does not require the use of the database manager, this field will not be displayed.

TABLE 5. **Application Status -- Request Information**

Field	*Content*
Total # Requests	Total number of requests of this application during its current instantiation. If the application is not running, “N/A” will be displayed in this field.
Current #/min	Current number of requests of this application per minute, during its current instantiation. If the application is not running, “N/A” will be displayed in this field.
Peak #/min	Peak number of requests of this application per minute, during its current instantiation. If the application is not running, “N/A” will be displayed in this field. A “Reset” link is provided here to allow for resetting this value to zero.
When	The time and date when the peak number of requests per minute occurred.

The Advanced section of the application status display is an optional, application specific area. Information is only displayed here if the application has a toHtml method implemented. The figure below is a snapshot of the “application info” area of one of the Enhydra applications in use at Lutris to give you the flavor of the type of information that might be displayed here.



Advanced

Application Info:	- **Submissions so far:** - Time sheets: 43 - Milage sheets: 0 - Expense sheets: 2 - **Currently logged in users:** - *No one is logged in*

The following diagram, and tables represent the standard fields in the servlet status display.

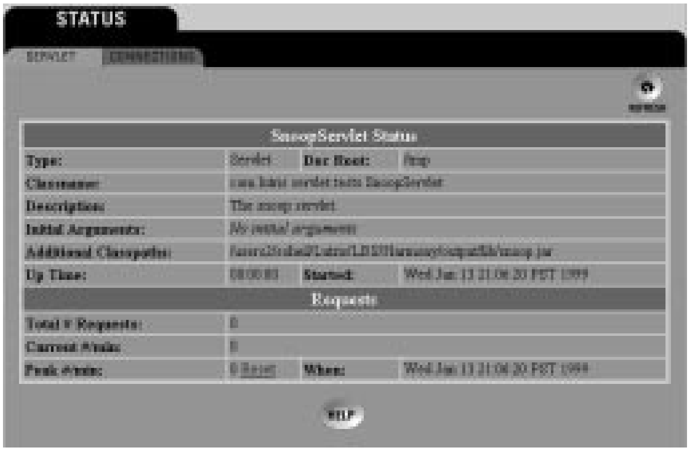


TABLE 6. Servlet Status -- General Information

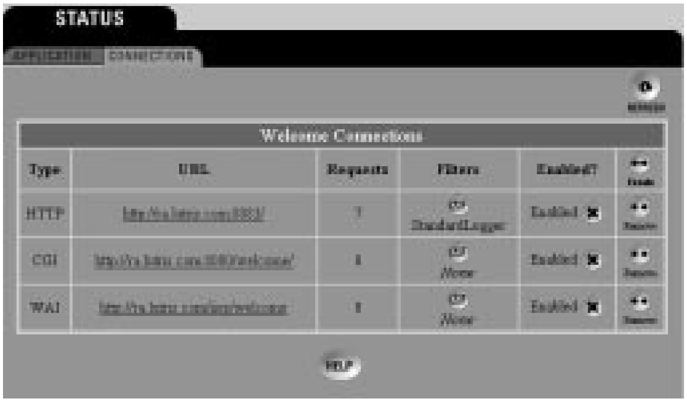
Field Name	*Content*
Type	By default, displays “Servlet”.
Doc Root	The file system location to associate with the document root of the servlet. For example, `/tmp`.
Classname	The class name of the servlet.
Description	A modifiable description of the servlet.
Initial Arguments	The initial arguments passed to the servlet, represented in “name = value” pairs. For example, “max = 256”. The text “no initial arguments” is displayed if there are none.
Additional Classpaths	Location of additional classpaths necessary for the successful instantiation of this servlet.
Up Time	The elapsed time since servlet start-up. If the servlet has not been started, "Not running" will appear in this field.
Started	The time of the most recent servlet start-up. If the servlet has not been started, "Not running" will appear in this field.

TABLE 7. Servlet Status -- Request Information

Field	*Content*
Total # Requests	Total number of requests of this servlet during its current instantiation. If the servlet is not running, “N/A” will be displayed in this field.
Current #/min	Current number of requests of this servlet per minute, during its current instantiation. If the servlet is not running, “N/A” will be displayed in this field.
Peak #/min	Peak number of requests of this servlet per minute, during its current instantiation. If the servlet is not running, “N/A” will be displayed in this field. A “Reset” link is provided here to allow for resetting this value to zero.
When	The time and date when the peak number of requests per minute occurred.

The Connections Status Display

Clicking on the Connections tab in the Status Display will display the connection status for the selected Application or Servlet. In addition to displaying the current status of connections for the Application or Servlet, the connections tab contains controls for manipulating connections. There are controls for adding new connections, removing existing connections, enabling and disabling connections, and modifying the filters associated with a connection.



Welcome Connections

Type	URL	Requests	Filters	Enabled?	
HTTP	[illegible]	[illegible]	StandardLogger	Enabled ☒	
CGI	[illegible]	[illegible]	None	Enabled ☒	
WAI	[illegible]	[illegible]	None	Enabled ☒	

Type

The first column describes the type of connection. Valid types are “HTTP”, “CGI”, “WAI”, and “RMI”.

URL

The second column describes the URL that may be used to access the application or servlet through the particular connection type. The URL’s listed are live links that may be clicked on to access the application or servlet. These links become inactive if the application or servlet is in the “stopped” state, or if the connection is set to the “Disabled” state.

Requests

The third column is a listing of the number of requests that have been received by the application or servlet through the particular connection.

Filters

The fourth column describes the filters associated with the particular connection. These may be used to filter selectively for certain types of data or conditions. Logs kept by the application can then be used to view information captured by these filters. The “MF” or modify filter button provided here allows for modification of the filter status. Selecting this button displays a popup window that displays currently configured filters, available filters, and buttons to add or remove these filters to the current connection.

Enabled?

The fifth column describes the status of the current connection. Valid states are “Enabled” and “Disabled”. A button depicting a check mark will allow for enabling a disabled connection, and a button depicting an “X” mark will allow for disabling a currently enabled connection.

Create/Remove

The sixth column provides a button for adding new connections, and a remove connection button for each of the existing connections.

Clicking the create connection button displays a popup window that allows for picking the type of connection desired, and the particular information required for creating that type of connection.

An **HTTP** connection opens a socket in the Multiserver (on your machine). You can pick any port number but if you do not have root access on your system, you must pick a number above 1000. If that port is already being used, the attempt to add the connection will fail. The URL Prefix allows you to connect one port to multiple applications, by giving each a unique first-part-of-the-URL. For example one application could use “/”, while another application could use “/app2/”. The URL Prefix is very similar to the part of the URL “/servlet/myServlet/” used when running servlets in other web servers, but you have total control of the prefix.

The **RMI** connection method is used in conjunction with the RMI Servlet. The RMI servlet is installed and configured on a remote server. When requests are sent to it, it forwards them on to this connection method, and they are actually handled by the application in the Multiserver. The service name is a Java RMI identifier; the remote servlet uses it to find the connection method. It does not appear in URL’s.

The **WAI** connection method uses a Netscape-specific remote procedure call mechanism to connect to a Netscape server. The URL prefix allows multiple applications to share a single WAI service. Specify the name of the Netscape server in the web server field (e.g. www.domain.com), and the port number on that Netscape server (e.g. 80). Be sure that WAI is enabled on the Netscape server (this can be accomplished via the graphical admin interface). Aside from enabling WAI on the Netscape server, no other configuration is needed: the Multiserver handles registering and unregistering the connections.

Documentation for the **CGI** connection method will be available soon.

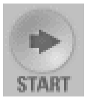
The Multiserver allows “many to many” interconnections between the outside world and applications. One connection to the net (e.g. a socket) may be connected to many applications. One application may be connected to the net in many ways (e.g. 4 sockets and WAI). Of course, the standard configuration of one port to one application is possible too :) This explains the “multi” in Multiserver.

Clicking the remove connection button will display a popup window asking for verification of removal. Clicking the “OK” button here will remove the connection.

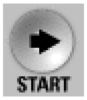
Additional buttons are provided on the connections status tab for refreshing the tab, and requesting context sensitive help (currently unavailable).

The Console Tools

The Console Tools, located directly below the Applications Window in the Control Frame, are a series of buttons that allow for the management of applications and servlets. If a tool is unavailable, the button will appear grayed-out in your browser window and will be disabled. Tools are grayed-out and disabled in this fashion to depict currently invalid operations. The two graphics of the "Start" tool below, depict the disabled and available states of this tool.



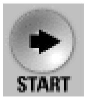
disabled



available

NOTE: *On Netscape browsers later than version 3, and Microsoft Internet Explorer browsers later than version 4, the Console Tools are animated and change to a bright color, when they are in the available state, and the user rolls over them with the mouse pointer.*

Start



Use the Start button to start an application or servlet that has already been installed.

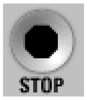
NOTE: *If the servlet or application you have selected is already running, the start button will be grayed-out and unavailable.*

To start a servlet or application:

- Select a servlet or application from the Applications Window by clicking on its name.
- Select the Start button from the Console Tools.

NOTE: *Since the "Start" operation changes the state of the application or servlet, performing this operation causes the server "Save State" tool to become activated. If you wish for this application or servlet to be auto-started the next time the Enhydra Multiserver is* *started, use the "Save State" tool to save this setting. Please refer to the "Save State" section below, for more details.*

Stop



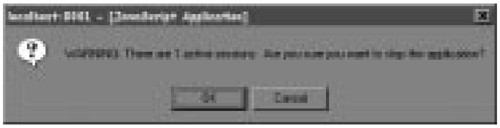
Use the Stop button to stop a servlet or application that is currently running.

NOTE: *If the servlet or application you have selected is not currently running, the Stop button will be unavailable.*

To stop a servlet or application:

- Select a servlet or application from the Applications Window by clicking on its name.
- Select the Stop button from the Console Tools.

NOTE: *If an application has active users, you will be prompted with the following, before the application is stopped:*



NOTE: *Since the "Stop" operation changes the state of the application or servlet, performing this operation causes the server "Save State" tool to become activated. If you wish for this application or servlet to be stopped the next time the Enhydra Multiserver is started, use the "Save State" tool to save this setting. Please refer to the "Save State"section below, for more details.*

Add

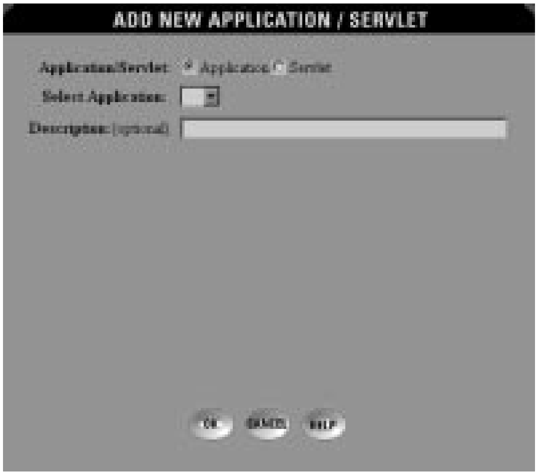
Use the Add tool to add a servlet or application to the Enhydra Multiserver.

> **NOTE:** *If you are adding an application, the application configuration file must be present in the Enhydra Multiserver* `server.confDir`*, /* `usr/local/enhydra/apps/` *by default.*

After selecting the Add tool from the Console Tools, a pop-up window will launch. By default the pop-up window displays the screen for adding Enhydra Applications. The radio buttons labeled “Application/Servlet” will toggle the popup between the screen for adding Applications, and the screen for adding Servlets.

To add an application:

- If it isn’t already selected, please be sure the “Application/Servlet” radio button, has “Application” selected.



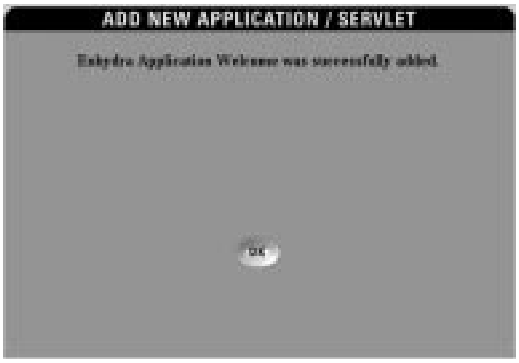
Application/Servlet: ● Application ○ Servlet

- Select an application from the pull-down menu. The names in the menu represent applications with configuration files located in the `server.confDir` in the Server Section of the Enhydra Multiserver Configuration File (see Chapter 3 for more information). By default, `server.confDir` is set to `/`

`usr/local/enhydra/apps/`. For each configuration file located in `server.confDir`, the associated application name appears in the menu; if the configuration file is called `Welcome.conf`, "Welcome" appears in the menu.

NOTE: *An application which has already been added to the Enhydra Multiserver does not appear as a selection in this menu. The menu will be empty if there are no more applications available to add.*

- Complete the optional Description field.
- Select "OK". The window will refresh and display notification of successful installation:



Enhydra Application Welcome was successfully added.

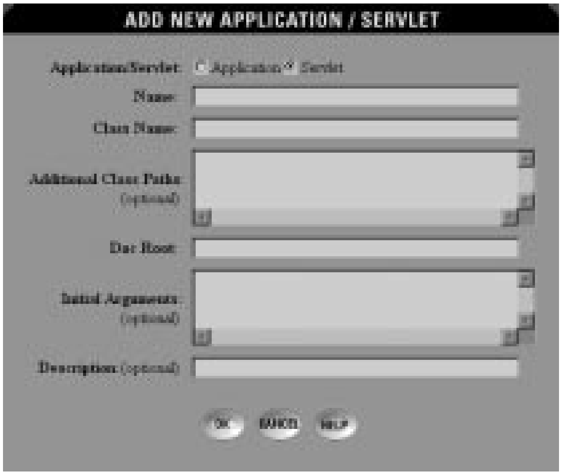
- Select "OK" to return to the Management Console.

The Applications Window is then updated to reflect the new application, and upon returning to the Management Console, the new application is selected by default. The status of the new application appears in the Content Frame.

To add a servlet:

- Please be sure the “Application/Servlet” radio button, has

"Servlet" selected.



- Complete the fields on the popup. Please refer to Table 8, "Adding Servlets -- Form Fields," below for information on these fields.

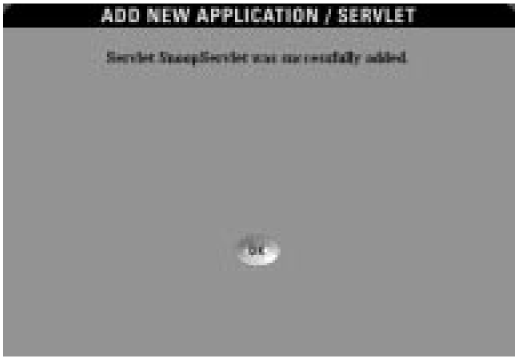
TABLE 8. Adding Servlets -- Form Fields

Field Name	*Content*
Name	This required field is the identifier string used to refer to this servlet.
Class Name	This required field is the name of the class to instantiate for this servlet. This class must implement the servlet interface.
Additional Class Paths (optional)	Any additional class paths that are required for this servlet. These class paths can be an array of directories, .zip files or .jar files. The servlet will be loaded by it's own class loader, which will look in these locations first. If a class is not found, the system class loader will be used (using the CLASSPATH with which the Enhydra Multiserver was started). Multiple class paths, each on its own line separated by carriage returns, can be specified in this field.
Doc Root	This required field specifies the root of the servlet's filesystem on disk.

TABLE 8. **Adding Servlets -- Form Fields**

Field Name	*Content*
Initial Arguments (optional)	Any initial arguments that are required for this servlet. These will be made available to the servlet's init() method. Multiple initial arguments, each on its own line separated by carriage returns, can be specified in this field.
Description (optional)	This optional field can be used to specify a description that will be associated with the servlet being added.

- Select "OK". The window will refresh and display notification of successful installation:



ADD NEW APPLICATION / SERVLET

Servlet SnoopServlet was successfully added.

- Select "OK" to return to the Management Console.

The Applications Window is then updated to reflect the new servlet, and upon returning to the Management Console, the new servlet is selected by default. The status of the new servlet appears in the Content Frame.

Delete

DELETE

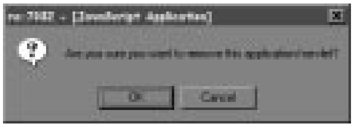
Use the Delete button to remove a servlet or application from the Enhydra Multiserver.

> **NOTE:** *If the application or servlet you have selected is currently running, the Delete button is unavailable. You must stop a running*

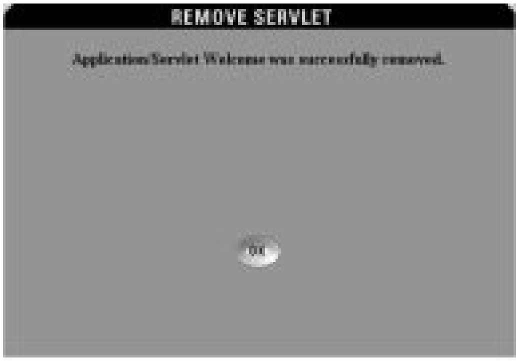
application or servlet prior to removing it.

To delete a servlet or application:

- Select a servlet or application from the Applications Window by clicking its name.
- Select the Delete button from the Console Tools.
- An alert will appear, asking you to confirm the deletion of the servlet or application.



- Select "OK" to confirm deletion, or "Cancel" to cancel the action.
- If you select “OK”, the following popup will appear, confirming the deletion:



Application/Servlet Welcome was successfully removed.

When a servlet or application is deleted, its entries are removed from the Enhydra Multiserver Configuration File. The servlet or application itself, however, is not removed from the file system, and its configuration file is not removed from `server.confDir`. You may add the servlet or application again at any time, using the Add tool.

NOTE: *Since the “Delete” operation changes the state of the application or servlet, performing this operation causes the server “Save State” tool to become activated. If you wish for this application or servlet to not be present the next time the Enhydra Multiserver is* *started, use the “Save State” tool to save this setting. Please refer to the “Save State”section below, for more details.*

Modify

Use the Modify tool to change configurable attributes of your application or servlet. Via the following modify screens you will be able to modify attributes associated with the selected application or servlet, that appear in that application or servlet’s own configuration file `/usr/local/enhydra/apps/<appname>.conf`; or attributes from the selected application or servlet, which appear in the Enhydra Multiserver configuration file `/usr/local/enhydra/multiserver.conf`. Once modified, the new settings for your application or servlet can then be saved to the appropriate configuration file.

NOTE: *If the servlet or application you have selected is currently running, the Modify button is unavailable.*

To modify a servlet or application:

- Select a servlet or application from the Applications Window by clicking its name.
- Select the Modify button from the Console Tools.
- The Content Frame will refresh, presenting you with several configuration sections. Tabs indicate the sections: Application/Servlet, Sessions, Database, Advanced.

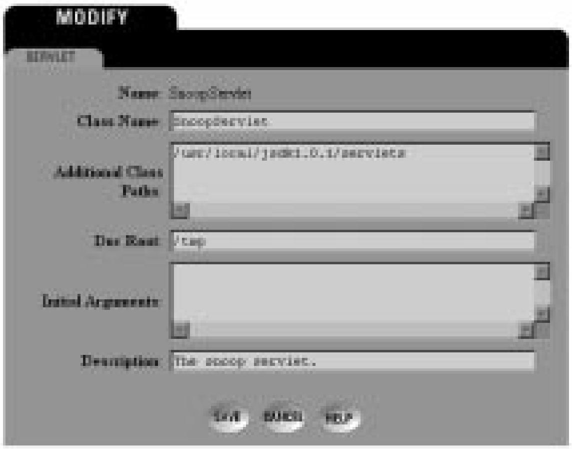
NOTE: *While modifying servlets, there will only be one tab, “Servlet”, displayed. The other tabs are specific to Enhydra applications and will only appear when modifying an application.*

- Select the tab containing the information you wish to modify.

Following are descriptions of the modifiable information found behind each tab. In most cases, specific parameters available on each tab are dependent upon the individual application or servlet.

> **NOTE:** *On each tab, the "Save" button can be used to write back your modifications to the appropriate configuration file; and the "Cancel" button can be used to cancel any changes and return to the status screen for the selected application or servlet. There is no need to use the "Save" button on every tab that you make modifications. Simply use whichever "Save" button is convenient, once you have finished making all of your modifications*.

Servlet Tab: This tab presents the only configurable parameters for servlets. All of the attributes presented for modification here, are read from and saved to the Enhydra Multiserver configuration file, `/usr/local/enhydra/multiserver.conf`. Please refer to Table 9, "Modifying Servlets -- Servlet Form Fields," below for details concerning the servlet attributes appearing on this tab.



Name	SnoopServlet
Class Name	SnoopServlet
Additional Class Paths	/usr/local/jsdk1.0.1/servlets
Doc Root	/tmp
Initial Arguments	
Description	The snoop servlet.

TABLE 9. Modifying Servlets -- Servlet Form Fields

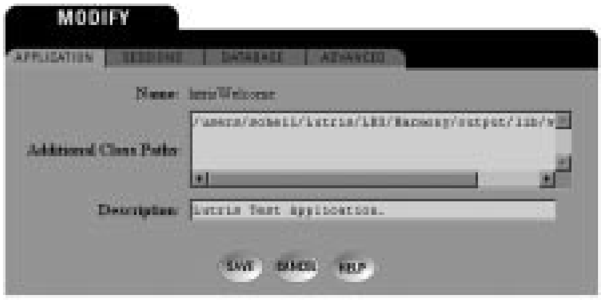
Field Name	*Content*
Name	This field is the identifier string used to refer to this servlet. It is not modifiable once set, and appears here for reference only.
Class Name	This required field is the name of the class to instantiate for this servlet. This class must implement the servlet interface.

TABLE 9. **Modifying Servlets -- Servlet Form Fields**

Field Name	*Content*
Additional Class Paths	Any additional class paths that are required for this servlet. These class paths can be an array of directories, .zip files or .jar files. The servlet will be loaded by it's own class loader, which will look in these locations first. If a class is not found, the system class loader will be used (using the CLASSPATH with which the Enhydra Multiserver was started). Multiple class paths, each on its own line separated by carriage returns, can be specified in this field.
Doc Root	This required field specifies the root of the servlet's filesystem on disk.
Initial Arguments	Any initial arguments that are required for this servlet. These will be made available to the servlet's init() method. Multiple initial arguments, each on its own line separated by carriage returns, can be specified in this field.
Description	This optional field can be used to specify a description that will be associated with the servlet being added.

Application Tab: This tab presents the general configurable parameters for the selected application. The "Additional Class Paths" field is read from and saved to the application's configuration file, `/usr/local/enhydra/apps/<appname>.conf`. The "Description" field is read from and saved to the Enhydra Multiserver configuration file, `/usr/local/enhydra/multiserver.conf`. Please refer to

“Table 3 Modifying Applications -- Application Form Fields”, for details concerning the application attributes appearing on this tab.



Description: Lutris Test application.

TABLE 10. Modifying Applications -- Application Form Fields

Field Name	*Content*
Name	This field is the identifier string used to refer to this application. It is not modifiable once set, and appears here for reference only.
Additional Class Paths	Any additional class paths that are required for this application. These class paths can be an array of directories, .zip files or .jar files. The application will be loaded by it's own class loader, which will look in these locations first. If a class is not found, the system class loader will be used (using the CLASSPATH with which the Enhydra Multiserver was started). Multiple class paths, each on its own line separated by carriage returns, can be specified in this field.
Description	This optional field can be used to specify a description that will be associated with the application being added.

Sessions: This tab presents session specific parameters for the selected application. All of the fields presented here are read from and saved to the application’s configuration file, `/usr/local/enhydra/apps/<appname>.conf.` Please refer to

Table 11, “Modifying Application -- Sessions Form Fields,” for details concerning the application attributes appearing on this tab.

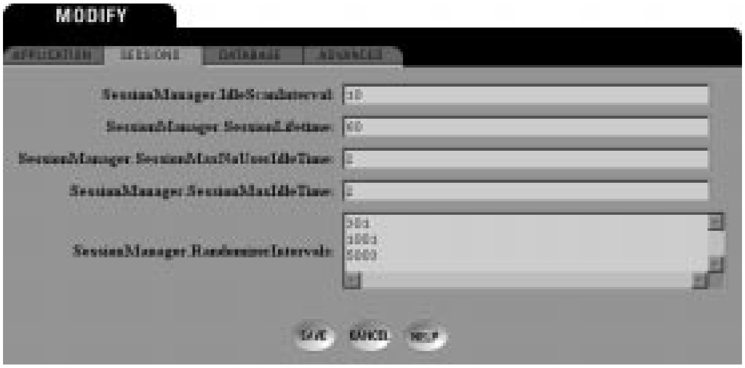
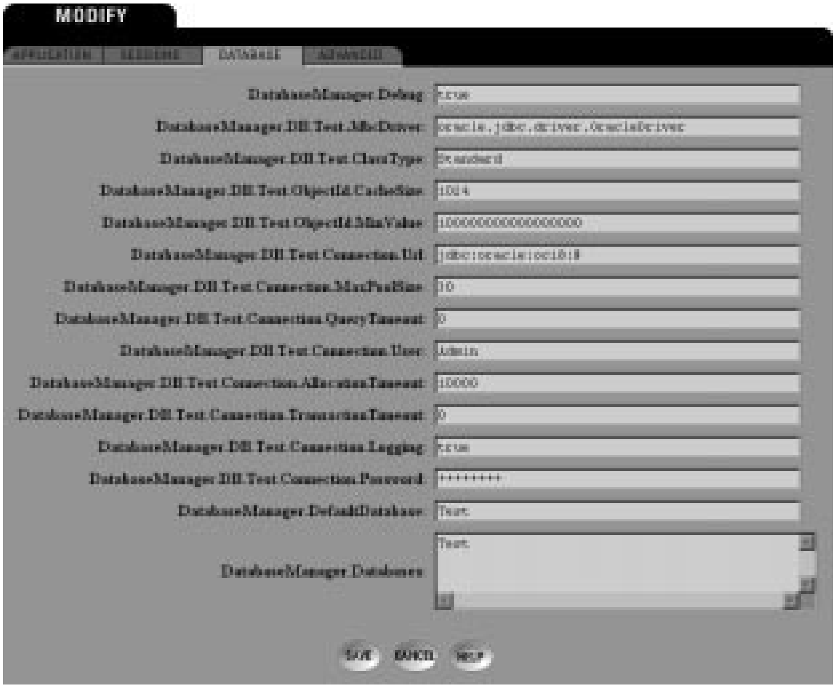

MODIFY
APPLICATION
SESSIONS
DATABASE
ADVANCED
SessionManager.IdleScanInterval
SessionManager.SessionLifetime
SessionManager.SessionMaxNoUserIdleTime
SessionManager.SessionMaxIdleTime
SessionManager.RandomizerIntervals
SAVE
CANCEL
HELP

TABLE 11. Modifying Application -- Sessions Form Fields

Field Name	*Content*
IdleScanInterval	How often, in seconds, the session manager tests if any sessions should be expired. This parameter must be greater than zero.
SessionLifetime	The maximum number of minutes a session is valid. If this value is zero, then there is no lifetime limit. This parameter must be set.
SessionMaxNoUserIdleTime	Maximum number of minutes a session that does not have a User object associated with it may be idle before the session is expired (deleted). If this value is less than or equal to zero, then the session may be idle indefinitely. Default is the same as MaxIdle.
SessionMaxIdleTime	Maximum number of minutes a session may be idle before the session is expired (deleted). If this value is less than or equal to zero, then the session may be idle indefinitely.
RandomizerIntervals	The set of varying intervals to use for adding user-generated entropy to the random number generator. It is a good idea to use several different prime numbers. These intervals are in seconds. Multiple intervals, each on its own line separated by carriage returns, can be specified in this field.

NOTE: *Not all of these attributes may appear for your selected application. The above table represents the list of all available attributes.*

Database: This tab presents database specific parameters for the selected application. All of the fields presented here are read from and saved to the application’s configuration file, `/usr/local/enhydra/apps/<appname>.conf`. Please refer to Table 12, “Modifying Application -- Database Form Fields,” for details concerning the application attributes appearing on this tab.



DatabaseManager.Debug	true
DatabaseManager.DB.Test.JdbcDriver	oracle.jdbc.driver.OracleDriver
DatabaseManager.DB.Test.ClassType	Standard
DatabaseManager.DB.Test.ObjectId.CacheSize	1024
DatabaseManager.DB.Test.ObjectId.MinValue	1000000000000000000
DatabaseManager.DB.Test.Connection.Url	jdbc:oracle:oci8:@
DatabaseManager.DB.Test.Connection.MaxPoolSize	10
DatabaseManager.DB.Test.Connection.QueryTimeout	0
DatabaseManager.DB.Test.Connection.User	Admin
DatabaseManager.DB.Test.Connection.AllocationTimeout	10000
DatabaseManager.DB.Test.Connection.TransactionTimeout	0
DatabaseManager.DB.Test.Connection.Logging	true
DatabaseManager.DB.Test.Connection.Password	********
DatabaseManager.DefaultDatabase	Test
DatabaseManager.Databases	Test

TABLE 12. Modifying Application -- Database Form Fields

Field Name	*Content*
Debug	Specify true to enable Query and Transaction logging, false to disable it. Optional, false if not specified.
DB.*dbname*.JdbcDriver	The JDBC driver to use to access the database. Mandatory.

TABLE 12. Modifying Application -- Database Form Fields

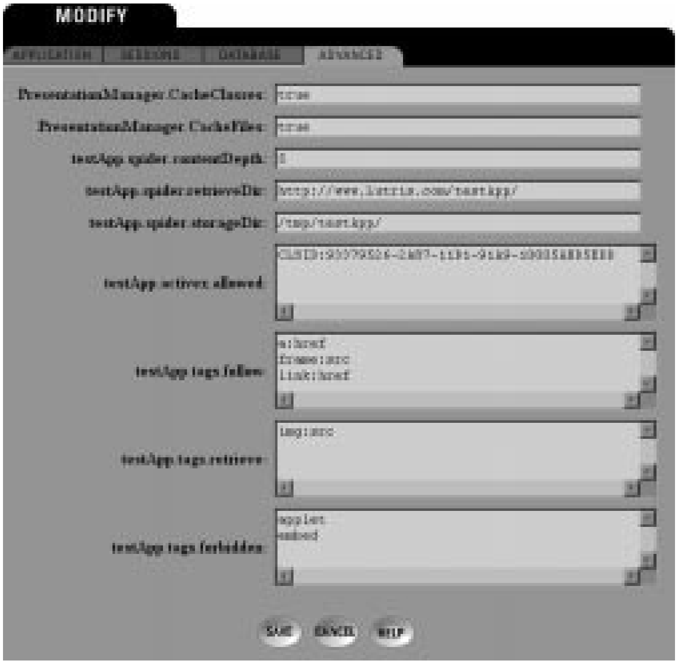
Field Name	*Content*
DB.*dbname*.ClassType	This is an optional field which specifies the class of the logical database implementation or a symbolic name if one on the standard types are selected. This is recommended because although JDBC abstracts the data access, the functionality of each database is slightly different and this parameter allows for optimized usage. The following are standard types: • Oracle - For optimized Oracle 7/8 usage. • Informix - For optimized Informix usage. • Sybase - For optimized Sybase usage. • Msql - For optimized Microsoft MSQL usage. • Standard - For all other JDBC databases. (Default) • class - For vendor supplied database classes.
DB.*dbname*.ObjectId.CacheSize	The number of object id's to cache between database queries. Optional, if not specified, then it defaults to 1024.
DB.*dbname*.ObjectId.MinValue	The starting number of Object ID allocation. This will only be used if the Object ID table is empty and thus is useful in development and testing. Optional, if not specified it defaults to 100000000000000000. Note that the largest number that can be associated with an OID in the Enhydra Multiserver is "database: DECIMAL(19,0)"
DB.*dbname*.Connection.Url	The JDBC URL of the database. Mandatory. e.g., "jdbc:sequelink://dbHost:4000/[Informix];Database=dummy"
DB.*dbname*.Connection.MaxPoolSize	The maximum number of open connections to the database. Optional, if not specified, then it defaults to 0. A value of 0 means that connections are allocated indefinitely or until the database (JDBC) refuses any new connections.
DB.*dbname*.Connection.QueryTimeout	The amount of time (in seconds) that a query will block before throwing an exception. If <= 0 then the query will not block. Optional, if not specified, then the value defaults to 0. This is not implemented by all logical databases.
DB.*dbname*.Connection.User	The database user used to access the database. Mandatory.

TABLE 12. **Modifying Application -- Database Form Fields**

Field Name	*Content*
DB.*dbname*.Connection.AllocationTimeout	The Maximum amount of time that a thread will wait for a connection from the connection allocator before an exception is thrown. This will prevent possible deadlocks. The timeout is in milliseconds. If the timeout value is <= 0, the allocation of connections will wait indefinitely. Optional, if not specified, timeout defaults to 1000 (ms).
DB.*dbname*.Connection.Transaction Timeout	The amount of time (in seconds) that a transaction will block before throwing an exception. If the timeout value is <= 0 then the transaction will not block. Optional, if not specified, then timeout defaults to 0. This is not implemented by all logical databases.
DB.*dbname*.Connection.Logging	Specify true to enable SQL logging, false to disable it. Optional, false if not specified.
DB.*dbname*.Connection.Password	The database user's password. Mandatory.
DefaultDatabase	The default logical database used by this application. Optional, if not specified, then the first entry in "Databases" is used.
Databases	A list of logical SQL database names.

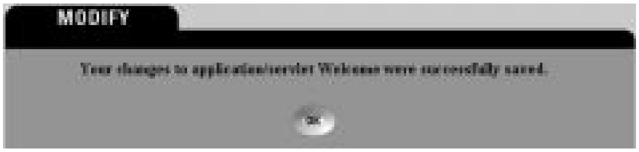
NOTE: *There may be no modifiable database attributes depending on the specific application selected. In which case, the text "Nothing to modify." appears under the Database tab.*

Advanced: This tab presents all of the application specific parameters for the selected application that are not related to sessions, and databases. Since the fields for this tab are very application specific, the example screen below is by no means representative of a typical application, and is shown only to illustrate the application specific nature of this tab. Please consult your specific application's documentation for further details on any of its attributes displayed here. All of the fields presented here are read from and saved to the application’s configuration file, `/usr/local/enhydra/apps/<appname>.conf`.



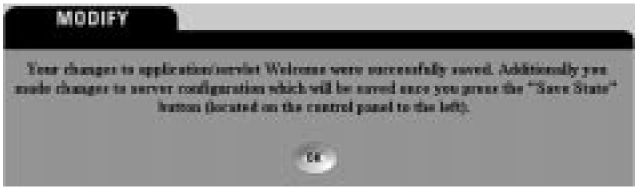
PresentationManager.CacheClasses: true
PresentationManager.CacheFiles: true
testApp.spider.contentDepth: [illegible]
testApp.spider.retrieveDir: http://www.lutris.com/testApp/
testApp.spider.storageDir: /tmp/testApp/
testApp.activex.allowed: CLSID:[illegible]
testApp.tags.follow: a:href
frame:src
link:href
testApp.tags.retrieve: img:src
testApp.tags.forbidden: applet
embed

- After making the appropriate modifications to your servlet or applications, select "SAVE". The following alert will appear in your browser window:



Your changes to application/servlet Welcome were successfully saved.

- However, if you made changes that also affected the Enhydra Multiserver configuration file, the following alert will appear in your browser window:



Your changes to application/servlet Welcome were successfully saved. Additionally you made changes to server configuration which will be saved once you press the "Save State" button (located on the control panel to the left).

- Select "OK" to return to Status information for the selected servlet or application. Any modifications that have been made to the servlet- or application-specific configuration files have been saved, and these changes will take effect upon starting the servlet or application.
- To save changes that affect the Enhydra Multiserver Configuration File, select the Save State tool in the Control Frame. Please see the Save State section below for details on this operation.

Debug

The debugging utility is used as a window into the operation of a running servlet or application to aid in debugging or maintenance. Enabling this feature will display a separate debugging control panel which is used to control debugging specific functions. Debugging information is then displayed in the “content frame” area of the display.

NOTE: *If the servlet or application you have selected is currently not running, or has no active connections the Debug button is disabled.*

To begin debugging a servlet or application:

- Select a servlet or application from the Applications Window by clicking its name.
- Select the Debug button from the Console Tools.
- A popup containing the debugging controls and event viewing space will appear.
- The name of the application/servlet will be appended with

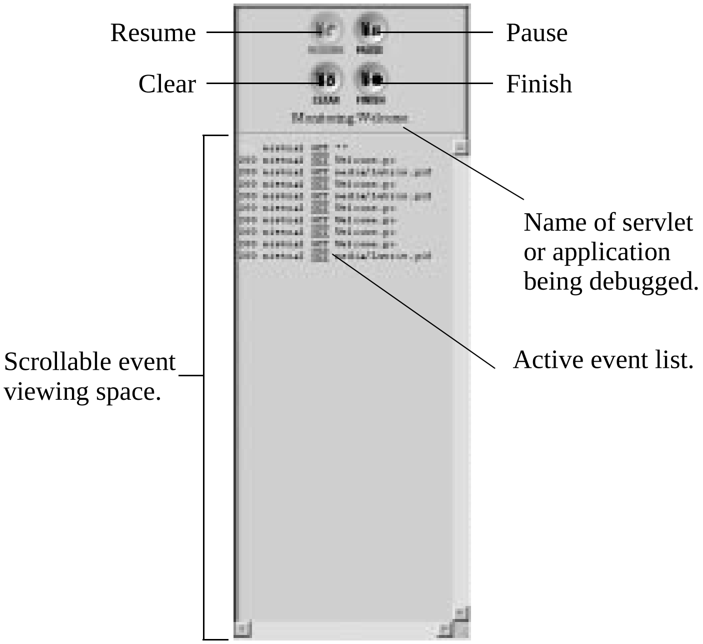
the designation “(D)” in the applications window to designate that this application is being debugged.

- You can now view events that are being handled by the application or servlet being debugged in the event viewing space.

NOTE: Events will display here only if the application or servlet is actively being used. If you do not see any events being displayed, you can access the servlet or application being debugged yourself to cause some event activity. You will then see the events your access caused, in the event viewing space.

The Debugging Controls

Once debugging is started. The following controls window appears. This window houses the debugging specific controls, and the scrollable event viewing space.



The “Pause” button pauses the debugging function, and stops the accumulation, and scrolling of events in the events area. This tool is useful while debugging a really busy application with a fast scrolling event list that needs to be paused for perusal. This button becomes disabled if the pause function is already on.

The “Resume” button resumes the debugging function, and restarts the accumulation and scrolling of events in the events area. This button is disabled until the “Pause” button is used to pause debugging.

The “Clear” button clears all of the current accumulated events listed in the event viewing space.

The “Finish” button halts the debugging function, and closes the debugging controls window. The application or servlet being debugged loses its “(D)” designation in the applications window.

The Active Event List

Once an application/servlet is in debugging mode, any event that is handled by this application appears in the debugging controls window’s event viewing space. Each line of the event list will have a format similar to the following:

```
200 mistral GET Welcome.po
```

The first column of this format (here: “200”) designates the status code returned to the browser as a result of this request. These status codes are color coded to designate different types of status codes that might need attention.

The second column (here: “mistral”) refers to the name of the remote host that originated this request.

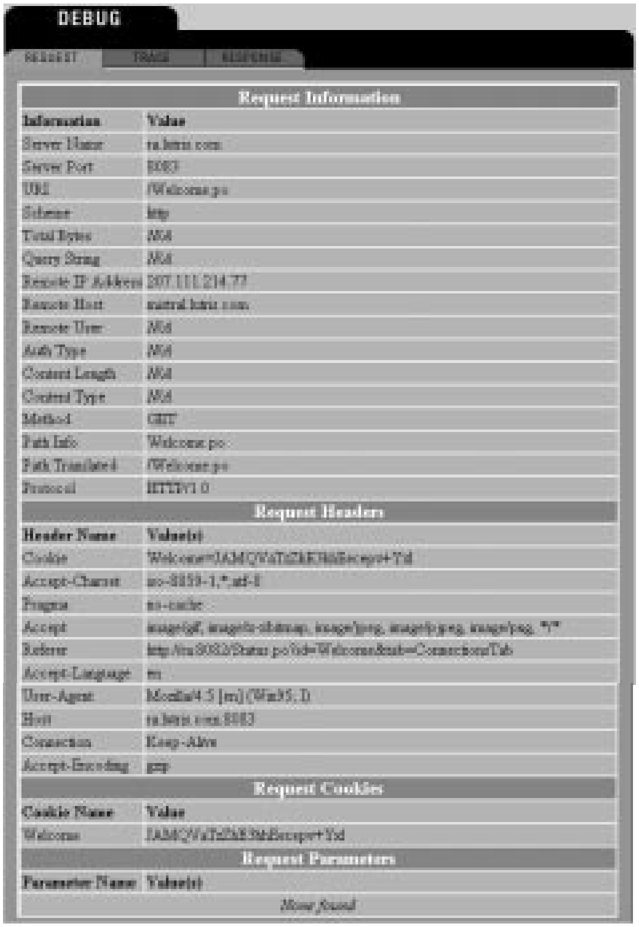
The third column (here: “GET”) refers to the type of event that this event is. Here, we have a “get” request (the other option is a “POST” request). This column is special in that the name of the request type (i.e., the word “GET”) is a live (clickable) link. Clicking on this link will refresh the administration console’s content frame with detailed information about this event. Please refer to “The Debugging Event Details” section below for an explanation of the information in these screens.

The fourth column (here: “Welcome.po”) refers to the resource being requested by the browser at the remote host.

The Debugging Event Details

Clicking the name of the type of event (GET or POST) from the third column of the debugging active event list (see “The Active Event List” section above), will display detailed information about that particular event. This detailed information will appear in the content frame of the administration console. The information is split between three tabs: “Request”, “Trace”, and “Response”. Following are descriptions of the information found behind each of the 3 tabs displayed for these event detail screens.

Request Tab: This tab represents information about the request sent from the remote browser to the Enhydra Multiserver. There are areas for general information, headers, cookies, and parameters associated with this request.



Request Information	
Information	**Value**
Server Name	ra.lutris.com
Server Port	8083
URI	/Welcome.po
Scheme	http
Total Bytes	*N/A*
Query String	*N/A*
Remote IP Address	207.111.214.77
Remote Host	mistral.lutris.com
Remote User	*N/A*
Auth Type	*N/A*
Content Length	*N/A*
Content Type	*N/A*
Method	GET
Path Info	Welcome.po
Path Translated	/Welcome.po
Protocol	HTTP/1.0

Request Headers	
Header Name	**Value(s)**
Cookie	Welcome=[illegible]
Accept-Charset	iso-8859-1,*,utf-8
Pragma	no-cache
Accept	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Referer	http://ra:8082/Status.po?id=Welcome&tab=ConnectionsTab
Accept-Language	en
User-Agent	Mozilla/4.5 [en] (Win95; I)
Host	ra.lutris.com:8083
Connection	Keep-Alive
Accept-Encoding	gzip

Request Cookies	
Cookie Name	**Value**
Welcome	[illegible]

Request Parameters	
Parameter Name	**Value(s)**
None found	

Please refer to the table below for details on this tab.

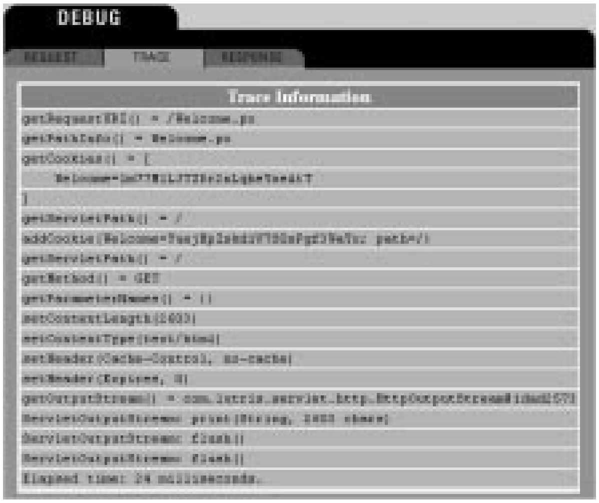
TABLE 13. Debugging Detail -- Request Tab Fields

Field Name	*Content*
Server Name	The name of the server hosting the Enhydra Multiserver.
Server Port	The port number on the server that is used to access the Enhydra Multiserver.

TABLE 13. Debugging Detail -- Request Tab Fields

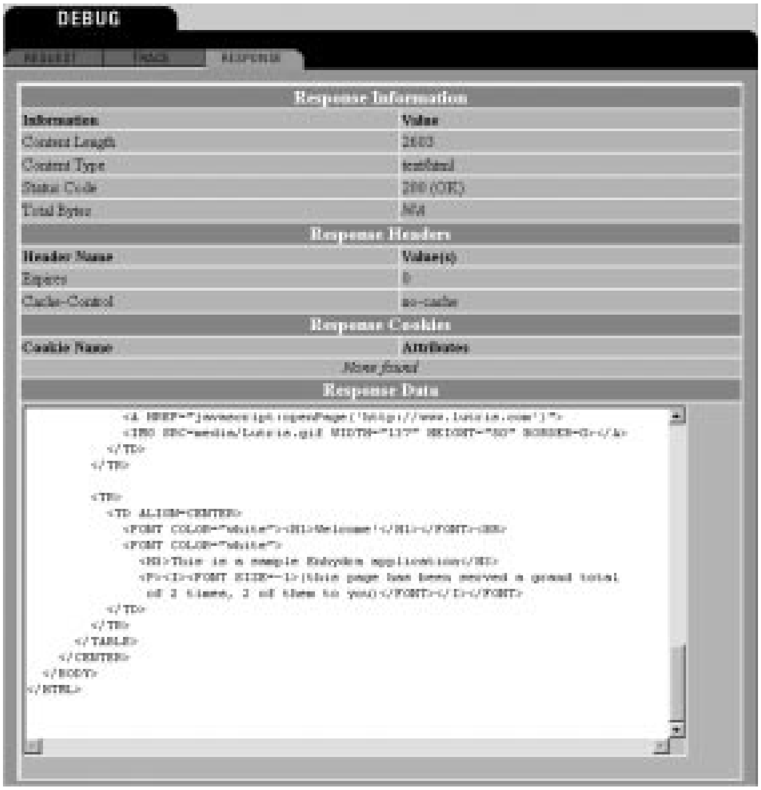
Field Name	*Content*
URI	The URI used to request a resource for this event.
Scheme	The scheme used to make the request.
Total Bytes	The number of bytes (if applicable) of this request.
Query String	The query string (if any) of this request.
Remote IP Address	The IP address of the remote client making this request.
Remote Host	The host name of the remote client making this request.
Remote User	The remote user name (if available) making this request.
Auth Type	The authentication type (if applicable) being used for this request.
Content Length	The content length (if applicable) of this request.
Content Type	The content type (if available) of this request.
Method	The method (get or post) applicable to this request.
Path Info	The path information for the resource being requested.
Path Translated	The translated path information for the resource being requested.
Protocol	The protocol and version being used for the communication of this request.
Cookie	The name and value of the header cookie associated with this request.
Accept-Charset	The acceptable character set specified in the header of this request.
Pragma	The pragma header for this request.
Accept	The accept header values for this request.
Referer	The referer URL header for this request.
Accept-Language	The header setting the acceptable language parameter.
User-Agent	The browser originating this request.
Host	The server host name and port number.
Connection	The connection type header setting.
Accept-Encoding	The acceptable encoding types for this request.
Cookie Name	The cookie name (if applicable) for this request, and its value.
Parameter Name	The parameters (if any) for this request.

Trace Tab: This tab represents a trace listing of calls made to the servlet API by the application as a result of this event. The last item on this list the amount of elapsed time for the application to handle this request. The diagram below is an example of a trace tab.



```
getRequestURI() = /Welcome.po
getPathInfo() = Welcome.po
getCookies() = [
    Welcome=[illegible]
]
getServletPath() = /
addCookie(Welcome=[illegible]; path=/)
getServletPath() = /
getMethod() = GET
getParameterNames() = []
setContentLength([illegible])
setContentType(text/html)
setHeader(Cache-Control, no-cache)
setHeader(Expires, 0)
getOutputStream() = com.lutris.servlet.http.HttpOutputStream@[illegible]
ServletOutputStream: print(String, [illegible] chars)
ServletOutputStream: flush()
ServletOutputStream: flush()
Elapsed time: 24 milliseconds.
```

Response Tab: This tab represents information about the response sent back to the remote browser from the Enhydra Multiserver as a result of this request. There are areas for general information, headers, cookies, and the actual response data. This last item is of note as the actual html source sent back to the browser will be displayed for html responses.



Response Information	
Information	**Value**
Content Length	2603
Content Type	text/html
Status Code	200 (OK)
Total Bytes	N/A

Response Headers	
Header Name	**Value(s)**
Expires	0
Cache-Control	no-cache

Response Cookies	
Cookie Name	**Attributes**
None found	

Response Data

Please refer to the table below for details on this tab.

TABLE 14. Debugging Detail -- Response Tab Fields

Field Name	*Content*
Content Length	The content length (if applicable) of this response.
Content Type	The content type (if available) of this response.
Status Code	The status code being returned to the browser with this response.
Total Bytes	The number of bytes (if applicable) of this response.

TABLE 14. **Debugging Detail -- Response Tab Fields**

Field Name	*Content*
Expires	The expiry time for the response data if cached.
Cache-Control	The cache control setting.
Cookie Name	The name of the cookie (if any) for this response.
Response Data	The data being returned as part of this response.

Save State

The Save State button allows for saving the current state of the Enhydra Multiserver. This current state is saved to the Enhydra Multiserver configuration file: `/usr/local/enhydra/multiserver.conf`, and is read the next time the server is started.

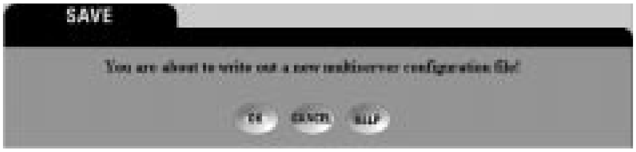
The Save State button starts out in a grayed-out and unavailable state. However, the first change to the state of the server causes the button to become active and available. The following is a list of operations that cause a change to the state of the server, and cause the Save State button to become active:

- Starting an application or servlet via the Start button.
- Stopping an application or servlet via the Stop button.
- Adding a new application or servlet via the Add button.
- Deleting an existing application or servlet via the Delete button.
- Modifying any of the fields for a servlet, using the Modify button. See Table 9, “Modifying Servlets -- Servlet Form Fields,” for details on these fields.
- Modifying the “Description” field for an application, using the Modify button. See Table 10, “Modifying Applications -- Application Form Fields,” for details.
- Adding a new connection to an application or servlet using the Add Connection button. See “The Connections Status Display” on page 27 for more details.

- Removing an existing connection from an application or servlet using the Remove Connection button. See “The Connections Status Display” on page 27 for more details.
- Enabling a disabled connection of an application or servlet using the Enable Connection button. See “The Connections Status Display” on page 27 for more details.
- Disabling an enabled connection of an application or servlet using the Disable Connection button. See “The Connections Status Display” on page 27 for more details.
- Adding a new filter to an existing connection of an application or servlet using the Modify Filters button. See “The Connections Status Display” on page 27 for more details.
- Removing an existing filter from a connection of an application or servlet using the Modify Filters button. See “The Connections Status Display” on page 27 for more details.

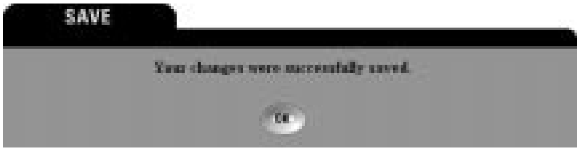
To save the current state of the Enhydra Multiserver:

- Perform one or more of the operations listed above, and arrive at the server state that you wish to save.
- Select the Save State button from the Console Tools.
- The Content Frame will refresh, presenting you with the following alert:



You are about to write out a new multiserver configuration file!

- Select “CANCEL” if you wish to abort the save operation, or select “OK” to proceed with the save. If you select “OK”, the window will refresh and display notification of a successful save operation:



Your changes were successfully saved.

- Select "OK" to return to the Management Console.

NOTE: *The Enhydra Multiserver configuration file:* `/usr/local/enhydra/multiserver.conf`*, can also be viewed and modified using any text editing/viewing program. Please refer to Chapter 3, “The Enhydra Multiserver Configuration File Format,” for details on the structure of this file.*

INDEX