BEA WebLogic Server™

and BEA WebLogic Express™

Administration Guide

BEA WebLogic Server Version 6.1
Document Date: June 24, 2002

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA WebLogic Server Administration Guide

Document Date	Software Version
June 24, 2002	BEA WebLogic Server Version 6.1

Contents

3. Node Manager

6. Using Log Messages to Manage WebLogic Servers

7. Deploying Applications

8. Configuring WebLogic Server Web Components

9. Proxying Requests to Another HTTP Server

10. Proxying Requests to a WebLogic Cluster

11. Installing and Configuring the Apache HTTP Server Plug-In

12. Installing and Configuring the Microsoft Internet

Information Server (ISAPI) Plug-In

13. Installing and Configuring the Netscape Enterprise Server Plug-In (NSAPI)

14. Managing Security

15. Managing Transactions

16. Managing JDBC Connectivity

17. Managing JMS

18. Using the WebLogic Messaging Bridge

19. Managing JNDI

20. Managing the WebLogic J2EE Connector Architecture

21. Managing WebLogic Server Licenses

A. Using the WebLogic Java Utilities

B. WebLogic Server Command-Line Interface Reference

C. WebLogic SNMP Agent Command-Line Reference

D. Parameters for Web Server Plug-ins

Index

About This Document

This document explains the management subsystem provided for configuring and monitoring your WebLogic Server implementation. It is organized as follows:

- Chapter 1, “Overview of WebLogic Server Management,” describes the architecture of the WebLogic Server management subsystem.
- Chapter 2, “Starting and Stopping WebLogic Servers,” explains the procedures for starting and stopping WebLogic Servers.
- Chapter 3, “Node Manager,” explains how to set up and use the Node Manager which you can use for remote starting and stopping of WebLogic Servers.
- Chapter 4, “Configuring WebLogic Servers and Clusters,” explains the facilities for configuring resources in a WebLogic Server domain.
- Chapter 5, “Monitoring a WebLogic Server Domain,” describes the WebLogic facilities for monitoring the resources that make up a WebLogic Server domain.
- Chapter 6, “Using Log Messages to Manage WebLogic Servers,” describes the use of the WebLogic Server local log and the domain-wide log for managing a WebLogic Server domain.
- Chapter 7, “Deploying Applications,” describes installation of applications on the WebLogic Server and the deploying of application components.
- Chapter 8, “Configuring WebLogic Server Web Components,” explains the use of WebLogic Server as a Web Server.
- Chapter 9, “Proxying Requests to Another HTTP Server,” describes use of WebLogic Server to serve as a proxy forwarding HTTP requests to other Web servers.
- Chapter 10, “Proxying Requests to a WebLogic Cluster,” describes proxying HTTP requests to a cluster of WebLogic Servers.

- Chapter 11, “Installing and Configuring the Apache HTTP Server Plug-In,” explains how to install and configure the WebLogic Server Apache plug-in.
- Chapter 12, “ Installing and Configuring the Microsoft Internet Information Server (ISAPI) Plug-In,” explains how to install and conifgure the WebLogic Server plug-in for the Microsoft Internet Information Server.
- Chapter 13, “Installing and Configuring the Netscape Enterprise Server Plug-In (NSAPI),” explains how to to install and configure the Netscape Enterprise Server proxy plug-in.
- Chapter 14, “Managing Security,” discusses WebLogic Server security resources and how to manage them.
- Chapter 15, “Managing Transactions,” explains how to manage the Java Transaction subsystem within a WebLogic Server domain.
- Chapter 16, “Managing JDBC Connectivity,” discusses the management of Java Database Connectivity (JDBC) resources within a WebLogic Server domain.
- Chapter 17, “Managing JMS,” discusses the management of Java Message Service within a WebLogic Server domain.
- Chapter 19, “Managing JNDI,” discusses how to use the WebLogic JNDI naming tree, including viewing and editing objects on the JNDI naming tree and binding objects to the JNDI tree.
- Chapter 20, “Managing the WebLogic J2EE Connector Architecture,” describes how extensions to the WebLogic J2EE platform that allow connections to other Enterprise Information Systems are managed.
- Chapter 21, “Managing WebLogic Server Licenses,” describes how to update your BEA license.
- Appendix A, “Using the WebLogic Java Utilities,” describes a number of utilities that are provided for developers and system administrators.
- Appendix B, “WebLogic Server Command-Line Interface Reference,” describes the syntax and usage of the command-line interface for managing a WebLogic Server domain.
- Appendix C, “WebLogic SNMP Agent Command-Line Reference,” describes using the WebLogic SNMP agent’s command line interface to retrieve the value of WebLogic Server attributes and generate and receive WebLogic Server traps.

- Appendix D, “Parameters for Web Server Plug-ins,” discusses the parameters for Web server plug-ins.

Audience

This document is intended mainly for system administrators who will be managing the WebLogic Server application platform and its various subsystems.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation.

How to Print the Document

You can print a copy of this document from a Web browser, one main topic at a time, by using the File→Print option on your Web browser.

A PDF version of this document is available on the WebLogic Server documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Server documentation Home page, click Download Documentation, and select the document you want to print.

Adobe Acrobat Reader is available at no charge from the Adobe Web site at http://www.adobe.com.

Contact Us!

Your feedback on BEA documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the documentation.

In your e-mail message, please indicate the software name and version you are using, as well as the title and document date of your documentation. If you have any questions about this version of BEA WebLogic Server, or if you have problems installing and running BEA WebLogic Server, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Usage
Ctrl+Tab	Keys you press simultaneously.
italics	Emphasis and book titles.

Convention	Usage

`monospace text`	Code samples, commands and their options, Java classes, data types, directories, and file names and their extensions. Monospace text also indicates text that you enter from the keyboard.
	Examples:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

`monospace italic text`	Variables in code. *Example*: `String CustomerName;`
UPPERCASE TEXT	Device names, environment variables, and logical operators. *Examples*: LPT1 BEA_HOME OR
`{ }`	A set of choices in a syntax line.
`[ ]`	Optional items in a syntax line. *Example*: `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`
`|`	Separates mutually exclusive choices in a syntax line. *Example*: `java weblogic.deploy [list|deploy|undeploy|update] password {application} {source}`
`...`	Indicates one of the following in a command line: ■ An argument can be repeated several times in the command line. ■ The statement omits additional optional arguments. ■ You can enter additional parameters, values, or other information

Convention	Usage
. . .	Indicates the omission of items from a code example or from a syntax line.

1 Overview of WebLogic Server Management

The following sections describe the tools available to manage WebLogic Server:

- Domains, the Administration Server and Managed Servers
- Administration Console
- Run-time and Configuration Objects
- Central Point of Access to Log Messages
- Creating a New Domain

Your implementation of BEA WebLogic Server™ software provides a set of interrelated resources for users. Managing these resources includes such tasks as starting and stopping servers, balancing the load on servers or connection pools, selecting and monitoring the configuration of resources, detecting and correcting problems, monitoring and evaluating system performance, and deploying Web applications, Enterprise Javabeans (EJBs) or other resources.

The main tool that WebLogic provides to accomplish these tasks is a Web-based Administration Console. The Administration Console is your window into the WebLogic Administration Service. The Administration Service — an implementation of Sun's Java Management Extension (JMX) standard — provides the facilities for managing WebLogic resources.

Through the Administration Console you can configure attributes of resources, deploy applications or components, monitor resource usage (such as server load or Java Virtual Machine memory usage or database connection pool load), view log messages, start or shut down servers, or perform other management actions.

Domains, the Administration Server and Managed Servers

An inter-related set of WebLogic Server resources managed as a unit is called a *domain*. A domain includes one or more WebLogic Servers, and may include WebLogic Server clusters.

The configuration for a domain is defined in Extensible Markup Language (XML). Persistent storage for the domain's configuration is provided by a single XML configuration file *`install_dir`*`/config/`*`domain_name`*`/config.xml` (where *`install_dir`* is the directory under which the WebLogic Server software has been installed). For more information on the `config.xml` file, see BEA WebLogic Server Configuration Reference.

A domain is a self-contained administrative unit. If an application is deployed in a domain, components of that application cannot be deployed on servers that are not a part of that domain. When a cluster is configured in a domain, all of its servers must be a part of that domain as well. A domain may contain multiple clusters.

A J2EE application is a collection of components that are grouped together into a deployment unit (such as an EAR, WAR or JAR file). The various WebLogic resources required for an application — EJBs or Web applications, servers or clusters, JDBC connection pools, and so on — are defined within a single domain configuration. Grouping these resources into a single, self-contained domain provides a unified viewpoint, and point of access, for managing these interrelated resources.

A WebLogic Server running the Administration Service is called an *Administration Server*. The Administration Service provides the central point of control for configuring and monitoring the entire domain. The Administration Server must be running in order to perform any management operation on that domain.

Note: The Administration Server must be running the same version of WebLogic Server as the Managed Servers in its domain. The Administration Server must also have the same or later service pack installed as the Managed Servers in its domain. For example, the Administration Server could be running version 6.1, Service Pack 2 while the Managed Servers are running version 6.1, Service Pack 1.

In a domain with multiple WebLogic Servers, only one server is the Administration Server; the other servers are called *Managed Servers*. Each WebLogic Managed Server obtains its configuration at startup from the Administration Server.

The same class, `weblogic.Server`, may be started as either the Administration Server for a domain or as a WebLogic Managed Server. A WebLogic Server not started as a Managed Server is an Administration Server.

In a typical configuration for a production system, the applications and components with your business logic would be deployed across Managed Servers and the role of the Administration Server would be that of configuring and monitoring the Managed Servers. If the Administration Server should go down, the applications deployed on the Managed Servers are not affected and continue processing client requests; in such a situation, the Administration Server can regain administrative control of the active domain once it is restarted. (For information on how this is done, see Restarting the Administration Server when Managed Servers are Running.)

Distributing an application and its components across a set of Managed Servers has a number of possible advantages. EJBs or other components that do processing can be distributed to ensure ready availability for the main application entry point. Performance may be enhanced if components that do different functions, such as database access and account transactions, are segregated to different Managed Servers. A component such as an EJB that is a resource for a variety of functions or applications can be isolated, so that its availability is independent of the state of other components. Multiple applications can be deployed within a single domain.

A domain is *active* if the Administration Server was started using the configuration for that domain. While a domain is active, only the Administration Server can modify the configuration file. The Administration Console and the command-line administration utilities provide windows into the Administration Server which enable you to modify the domain configuration. Once the domain is activated, you can monitor and configure the resources of the entire domain via the Administration Console.

Additional non-active domain configurations may reside in the configuration repository, and you can edit them using the Administration Console. The configuration repository consists of a series of subdirectories (at least one) under the `/config` directory. Each domain is defined in a distinct `config.xml` file residing in a separate subdirectory; the name of that subdirectory is the domain name. To access non-active domain configurations, follow the `Domain Configurations` link on the Administration Console Welcome page when you start the Console.

Administration Console

The Administration Console is a JSP-based application hosted by the Administration Server. You can access the Administration Console using a Web browser from any machine on the local network that can communicate with the Administration Server (including a browser running on the same machine as the Administration Server). The Administration Console allows you to manage a WebLogic Server domain containing multiple WebLogic Server instances and applications. The management capabilities include:

- Configuration
- Stopping and starting servers
- Monitoring server performance
- Monitoring application performance
- Viewing server logs
- Editing deployment descriptors for Web Applications, EJBs, J2EE Connectors, and Enterprise Applications.

Using the Administration Console, system administrators can easily perform all WebLogic Server management tasks without having to learn the JMX API or the underlying management architecture. The Administration Server persists changes to attributes in the `config.xml` file for the domain you are managing.

For more information, see:

- Administration Console Online Help at `http://e-docs.bea.com/wls/docs61/ConsoleHelp/index.html`. (The online help is also available from the Administration Console by clicking on the "?" icons.)

After starting the Administration Server (see Starting and Stopping WebLogic Servers), you can start the Administration Console by directing your browser to the following URL:

`http://`*`hostname`*`:`*`port`*`/console`

The value of *hostname* is the DNS name or IP address of the Administration Server and *port* is the address of the port on which the Administration Server is listening for requests (7001 by default). If you started the Administration Server using Secure Socket Layer (SSL), you must add `s` after `http` as follows:

`https://`*hostname*`:`*port*`/console`

If you have your browser configured to send HTTP requests to a proxy server, then you may need to configure your browser to not send Administration Server HTTP requests to the proxy. If the Administration Server is on the same machine as the browser, then ensure that requests sent to `localhost` or `127.0.0.1` are not sent to the proxy.

The left pane in the Administration Console contains a hierarchical tree — the *domain tree* — for navigating to tables of data, configuration pages and monitoring pages, or accessing logs. By selecting (that is, left-clicking) an item in the domain tree, you can display a table of data for resources of a particular type (such as WebLogic Servers) or configuration and monitoring pages for a selected resource. The top-level nodes in the domain tree are containers. If leaf nodes are present in those containers, you can click on the plus sign at the left to expand the tree to access the leaf nodes.

The entity tables — tables of data about resources of a particular type — can be customized by adding or subtracting columns that display values for attributes. You can customize a table by following the `Customize this table` link at the top of the table. Each column in the table corresponds to an attribute that has been selected for inclusion in the table.

When started, the Administration Console prompts for a password. The first time the Administration Console is started, you can use the user name and password under which the Administration Server was started. You can use the Administration Console to add users to the Administrators group. Once users (or groups of users) have been added to the Administrators group, these users can also perform administrative tasks via the Administration Console. The default member of the Administrators group is `system`.

Because an Administration Server can manage only one active domain, you can access only one active domain at a time using the Administration Console. If you have separate Administration Servers running, each with its own active domain, you can switch from managing one domain to the other only by invoking the Administration Console on the Administration Server that you want to access.

Run-time and Configuration Objects

The Administration Server is populated with JavaBean-like objects called Management Beans (MBeans), which are based on Sun's Java Management Extension (JMX) standard. These objects provide management access to domain resources.

The Administration Server contains both configuration MBeans and run-time MBeans. Configuration MBeans provide both SET (write) and GET (read) access to configuration attributes.

Run-time MBeans provide a snapshot of information about domain resources, such as current HTTP sessions or the load on a JDBC connection pool. When a particular resource in the domain (such as a Web application) is instantiated, an MBean instance is created which collects information about that resource.

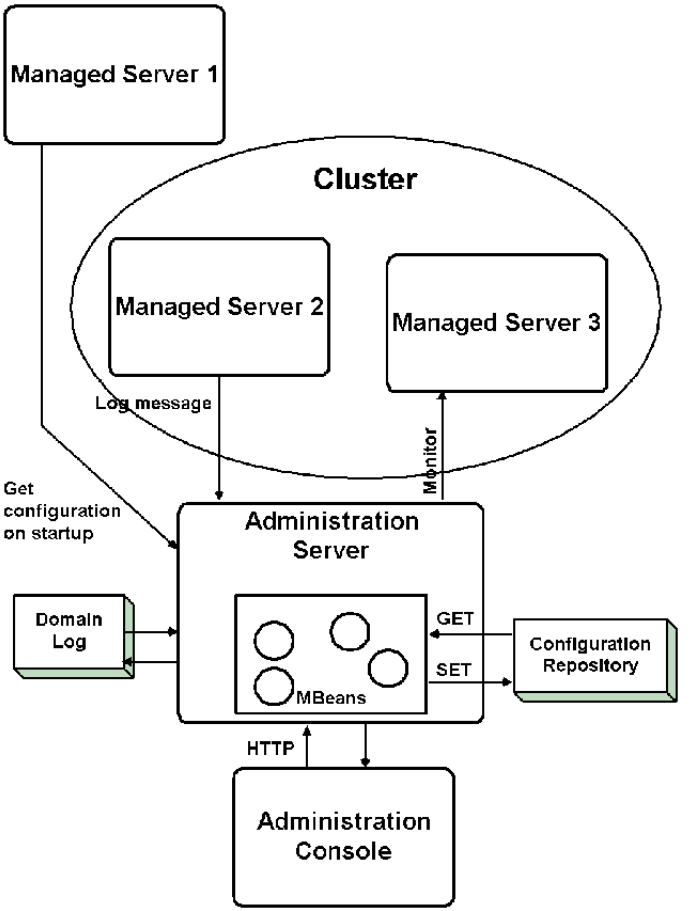
When you access the monitoring page for particular resources in the Administration Console, the Administration Server performs a GET operation to retrieve the current attribute values.

The Administration Service allows you to change the configuration attributes of domain resources dynamically — while the WebLogic Servers are running. For many attributes, you do not need to restart the servers for your change to take effect. In that case, a change in configuration is reflected in both the current run-time value of the attribute as well as the persistently stored value in the configuration file. (For more information about configuring WebLogic Servers, see Configuring WebLogic Servers and Clusters.)

In addition to the Web-based Administration Console, WebLogic Server provides a command-line utility for accessing configuration and monitoring attributes of domain resources. This tool is provided for those who want to create scripts to automate system management. For more information, see:

- WebLogic Server Command-Line Interface Reference for information on using the comman-line utility.
- BEA WebLogic Server Configuration Reference for information on using Mbeans and the config.xml file.

Figure 1-1 WebLogic Server Management Subsystem



Central Point of Access to Log Messages

The Administration Server also provides central access to critical system messages from all the servers via the domain log. JMX provides a facility for forwarding messages to entities that subscribe for specified messages. Subscriber entities specify

which messages to forward by providing a filter that selects messages of interest. A message forwarded to other network entities on the initiative of a local WebLogic Server is called a *notification*. JMX notifications are used to forward selected log messages from all WebLogic Servers in the domain to the Administration Server.

When a WebLogic Managed Server starts, the Administration Server registers to receive critical log messages. Such messages are stored in the *domain log*. A single domain log filter is registered with each WebLogic Server by the Administration Server to select the messages to be forwarded. You can change the domain log filter, view the domain log, and view the local server logs using the Administration Console. (For details, see Using Log Messages to Manage WebLogic Servers.)

Creating a New Domain

This section describes how to create a new domain. The configuration information for all of the WebLogic administrative domains reside in the configuration repository, which is located under the `/config` directory. Each domain has a separate subdirectory under the `/config` directory. The name of the subdirectory for a domain must be the name of that domain.

When you first install WebLogic Server software, it is recommended that you create a zip file that has a copy of the default *`install_dir`*`/config/`*`mydomain`* configuration directory (where *`install_dir`* is the root directory where you installed the WebLogic Server software and *`mydomain`* is the default domain configuration you specified at installation). You should keep a copy of this zip file as a backup that you can use for creating new domains. This subdirectory contains components that are required for a working configuration, such as a `fileRealm.properties` file and a configuration file.

In the procedure below `mydomain` is assumed as the default configuration directory name selected during installation. If you used a name other than `mydomain` for the default configuration directory, substitute that name for `mydomain` throughout.

To create a new domain, do the following:

1. Start the Administration Server under an existing domain such as `mydomain`.
2. Invoke the Administration Console by pointing your browser to:

`http://`*`hostname`*`:`*`port`*`/console`

where *`hostname`* is the name of the machine where you started the Administration Server and *`port`* is the Administration Server's listen port (default is 7001).

3. Select mydomain→Create or edit other domains.

   This displays the domains table.

4. Select Default→Create a new Domain.

   Enter the name of the new domain and click Create.

5. Select the new domain from the list of domains at left to make that the current domain.

6. Now you will need to create an Administration Server entry for the new domain:

   a. Select Servers→Create a new Server.

   b. Enter the name of the new Administration Server and click Create. Each server must have a unique name — even if the servers are in different domains

7. The Administration Console will have created a new subdirectory with the name of your domain and a configuration file, `config.xml`, under that subdirectory. Now you need to create an `\applications` subdirectory in that domain directory. You can create an `\applications` subdirectory in a command shell or, on Windows, by using Explorer.

8. The default `mydomain` directory contains start scripts for starting the WeLogic Server. For Windows installations, these are `startWebLogic.cmd` and `startManagedWebLogic.cmd`. For UNIX installations, these are `startWebLogic.sh` and `startManagedWebLogic.sh`. Copy these start scripts to the new domain directory.

9. You will need to edit the start scripts in a text editor. By default, the name of the domain is set as:

   ```
   -Dweblogic.Domain=mydomain
   ```

   Replace `mydomain` with the name of the new domain.

   By default the name of the Administration Server is set as:

   ```
   -Dweblogic.Name=MyServer
   ```

   Replace `MyServer` with the name of the new Administration Server.

10. At the end of the start script there is a `cd` command:

    ```
    cd config\mydomain
    ```

    Replace `mydomain` with the subdirectory name of the new domain. There is also a line in the start script that reads:

    ```
    echo startWebLogic.cmd must be run from the config\mydomain
    directory.
    ```

    Replace `mydomain` here with the name of the new domain.

11. Copy the file `SerializedSystemIni.dat` and the file `fileRealm.properties` from the default `mydomain` directory to your new domain directory. Do not try to boot the new Administration Server before copying these files.

Once you have completed this procedure, you can start the Administration Server for your new domain.

2 Starting and Stopping WebLogic Servers

The following sections describe procedures for starting and stopping Administration Servers and Managed Servers:

- WebLogic Administration Server and WebLogic Managed Servers
- Starting the WebLogic Administration Server
- Adding a WebLogic Managed Server to the Domain
- Starting a WebLogic Managed Server
- Stopping WebLogic Servers from the Administration Console
- Setting Up a WebLogic Server Instance as a Windows Service
- Registering Startup and Shutdown Classes

WebLogic Administration Server and WebLogic Managed Servers

A WebLogic Server **domain** may consist of one or more WebLogic Servers. WebLogic Server can be started as either an Administration Server or as a Managed Server. One (and no more than one) of the WebLogic Servers in a domain must be the

Administration Server for the domain. Additional WebLogic Servers in the domain are **managed** servers. Whether a WebLogic Server is an Administration Server or a Managed Server depends on the command-line options used when starting the server.

The default role for a WebLogic Server is the Administration Server. Therefore, if there is only one WebLogic Server in a domain, that server is the Administration Server. In a multi-server domain, a WebLogic Server becomes a Managed Server only if it is instructed to obtain its configuration from a running Administration Server when started.

The Administration Server controls access to the configuration for a WebLogic Server domain and provides other management services such as monitoring and log message browsing. The Administration Server serves up the Administration Console which provides user access to the management services offered by the Administration Server.

When a WebLogic Managed Server is started, it obtains its configuration from the Administration Server. For this reason, booting a multi-server WebLogic Server domain is a two-step procedure: First you start the Administration Server, and then you start the Managed Servers.

Note: The Administration Server and all Managed Servers in a domain must be the same WebLogic Server version. The Administration Server must be either at the same service-pack level or at a later service-pack level than the Managed Servers. For example, if the Managed Servers are at release 6.1 SP1, then the Administration Server can be either release 6.1 SP1 or SP2. However, if the Managed Servers are at SP2, then the Administration Server must be at SP2. Each server must have a unique name — even if the servers are in different domains.

Startup Messages

When a WebLogic Server is starting, the normal logging subsystem is not yet available for logging. Accordingly, any errors encountered during startup are logged to `stdout`. If you start a Managed Server remotely from the Administration Console, using the Node Manager, these messages are also displayed in the right pane of the Administration Console.

Failover Considerations for the Administration Server

Because the Administration Server contains the configuration repository (`config.xml`), security files, and application files for your domain, you should keep an archived copy of these files in case a failure of the Administration Server causes them to become unavailable. Common methods of archiving include periodic back-ups, fault tolerant disks, and manually copying files whenever they are changed. Remember that any configuration changes you make to Weblogic Server, either by using the Administration Console, the `weblogic.admin` command, or the JMX API are persisted in the `config.xml` file.

To provide for quick failover in case of an Administration Server crash or other failure, you may wish to create another instance of the Administration Server on a different machine that will be ready to use if it the original Administartion Server fails.

As long as you have back ups of your configuration, security, and application files, you can safely restart the Administration Server on another machine without interrupting the functioning of the Managed Servers. For instructions, see "Restarting the Administration Server when Managed Servers are Running" on page 2-13.

Starting the WebLogic Administration Server

There are several ways in which the WebLogic Administration Server can be started:

- From the command line

  The command to start the WebLogic Server can be either typed in a command shell manually or it can be placed in a script to avoid retyping the command each time the server is started. For information on the sample scripts provided see Starting the WebLogic Managed Servers Using Scripts.

- From the Start Menu (Windows only)
- A WebLogic Server installed as a Windows service will start automatically when the computer is rebooted.

Note: When starting WebLogic Server, JDK 1.3 may throw an OutOfMemory error if you are trying to load a large number of classes. This error occurs even though there appears to be plenty of memory available. If you encounter a `java.lang.OutOfMemory` error exception when you start WebLogic Server, increase the value of the following JMS option:

```
java -XX:MaxPermSize=<value>
```

where <value> is some number in kilobytes.

For JDK1.3.0, the JVM default value for `MaxPermSize` is `max` value of 32m (where m stands for megabytes). For JDK1.3.1, the default value for MaxPermSize is 64m.

Use of Passwords When Starting the WebLogic Server

During installation you are asked to specify a password that will be required when the server is started. If you use start scripts to start an Administration Server or a Managed Server, you can include the password as a command-line argument (See Starting the WebLogic Administration Server from the Command Line.) If you start the server using a script without the password specified as a command-line argument, you will be prompted to enter the password. You can avoid being prompted if the password is specified as a command-line argument, but then the password will be stored in clear text in the script file.

Starting the WebLogic Administration Server from the Start Menu

If you installed WebLogic Server on Windows with the BEA Installation program, you can use the WebLogic Server shortcut on the Windows Start menu to start the WebLogic Administration Server. Select:

Start→Programs→BEA WebLogic E-Business Platform→Weblogic Server *Version*→Start Default Server

where *`version`* is the WebLogic Server software version number.

Invoking the WebLogic Server from the Start menu executes the start script `startWeblogic.cmd` (which is located in *`install_dir`*`/config/`*`domain_name`* where *`domain_name`* is the name of the domain and *`install_dir`* is the directory where you installed the WebLogic Server software). You will be prompted to enter the password.

Starting and Stopping the WebLogic Server as a Windows Service

When installed as a Windows service, the WebLogic Server starts automatically when you boot the Windows computer. The WebLogic Server is started by executing a start script such as `startWeblogic.cmd`. A WebLogic Server started using `startWebLogic.cmd` is started as an Administration Server. See Starting the WebLogic Administration Server from the Command Line.

To run the WebLogic Server as a Windows service, you must have installed it as such. For information on installing and removing the WebLogic Server as a Windows service, see Setting Up a WebLogic Server Instance as a Windows Service.

You can also stop and start the WebLogic Server easily from the Service Control Panel.

1. Select Start→Settings→Control Panel.
2. Double-click the Services Control Panel to open it.
3. In the Services Control Panel, scroll to the end to find `WebLogic Server`. If WebLogic is Started, you will have the option to Stop it when you select it, by clicking the Stop button to the right. If WebLogic is Stopped, the Start button will be available.

You can make the Windows service Automatic, Manual, or Disabled by clicking the Startup button and selecting a mode.

Starting the WebLogic Administration Server from the Command Line

The WebLogic Server is a Java class file, and like any Java application, you can start it with the `java` command. The arguments needed to start the WebLogic Server from the command line can be quite lengthy and typing it out whenever you need to start the server can be tedious. To make sure that your startup commands are accurate, BEA Systems recommends that you incorporate the command into a script that you can use whenever you want to start a WebLogic Server.

The following arguments are required when starting the WebLogic Administration Server from the `java` command line:

- Specify the minimum and maximum values for Java heap memory.

  For example, you may want to start the server with a default allocation of 64 megabytes of Java heap memory to the WebLogic Server. To do so, you can start the server with the `java -ms64m` and `-mx64m` options.

  For best performance it is recommended that the minimum and maximum values be the same so that the JVM does not resize the heap.

  The values assigned to these parameters can dramatically affect the performance of your WebLogic Server and are provided here only as general defaults. In a production environment you should carefully consider the correct memory heap size to use for your applications and environment.

- Set the `java -classpath` option.

  The minimum content for this option is described under Setting the Classpath Option.

- Specify the name of the server.

  The domain configuration specifies configuration by server name. To specify the name of the server on the command line, use the following argument:

  `-Dweblogic.Name=`*`servername`*

  The default value is *`myserver`*.

- Specify the listen address of the server.

  If you want to run Managed Servers on other machines in the same domain (or run the Administration Server and a Managed Server on a multi-homed

machine) or start Managed Servers remotely using the Node Manager, you need to set the listen address of the Administration Server. To set the listen address, include the following argument:

`-Dweblogic.ListenAddress=`*`host`*

where *`host`* is the DNS name or IP address of the Administration Server.

- Provide user password.

  The default user is `system` and the required password is the password specified during installation. To enter the password, include the following argument:

  `-Dweblogic.management.password=`*`password`*

- Specify the WebLogic Server root directory if you do not want to use the current directory (that is, the directory from which you invoke the `weblogic.Server` command) as the server root directory. To specify a root directory, include the following argument:

  `-Dweblogic.RootDirectory=`*`path`*

  where *`path`* specifes the directory above the `config\`*`server-root`* directory.

  For example, if a server's root directory is `c:\myproject\config\Mydomain`, then specify `-Dweblogic.RootDirectory=c:\myproject`. For more information, refer to "A Server's Root Directory" on page 2-11.

- Specify the location of the `bea.home` directory:

  `-Dbea.home=`*`root_install_dir`*

  where *`root_install_dir`* is the root directory under which you installed the BEA WebLogic Server software.

- If you want to start the server with Secure Socket Layer (SSL) protocol, you need to pass the server the private key password at startup so that the server can decrypt the SSL private key file. To pass the SSL private key password to the server on startup, include the following argument on the command line:

  `-Dweblogic.management.pkpassword=`*`pkpassword`*

  where *`pkpassword`* is the SLL private key password.

  Note: When storing the password to the `pkpassword.ini` file, WebLogic Server encrypts the value using the encryption service associated with the domain. This means that you can use the password file only with the domain in which it was created.

Note: Secure a plain text copy of the private key password before you allow WebLogic Server to write the password to a file. You will not be able to retrieve the plain text password from `pkpassword.ini` after booting the server.

- When using SSL, you can turn off host name verification. By default, the Host Name Verifier in WebLogic Server compares the SubjectDN of a digital certificate with the host name of the server that initiated the SSL connection. If the SubjectDN and the host name do not match, the SSL connection is dropped. Should you decide to turn off host name verification (for example, to use the demonstration digital certificates shipped with WebLogic Server) include the following argument on the command line:

  ```
  -Dweblogic.security.SSL.ignoreHostnameVerification=true
  ```

  Note that BEA does not recommend using the demonstration digital certificates or turning off host name verification in any type of production deployment.

- To use a custom Host Name Verifier with WebLogic Server include the following argument on the command line:

  ```
  -Dweblogic.security.SSL.HostnameVerifier=hostnameverifierimplmentation
  ```

  where *`hostnameverifierimplmentation`* is the name of the class that implements the `weblogic.security.SSL.HostnameVerifier` interface.

- SSL session caching is turned on by default. To modify the server-session caching default size and time-to-live, include the following arguments on the command line:

  ```
  -Dweblogic.security.SSL.sessionCache.size=sessionCacheSize
  -Dweblogic.security.SSL.sessionCache.ttl=sessionCacheTimeToLive
  ```

  where *`sessionCacheSize`* is the size of the session cache and *`sessionCacheTimeToLive`* is the session cache time-to-live in seconds. The minimum, maximum, and default values for the two parameters are:

  `sessionCache.size`: min 1, max 65537, default 211
  `sessionCache.ttl`: min 1, max `Integer.MAX_VALUE`, default 90

- You can specify the name of the domain configuration when starting the Administration Server by using the following argument on the command line:

  ```
  -Dweblogic.Domain=domain_name
  ```

where *domain_name* is the name of the domain. This will also be the subdirectory which has the configuration file that will be used to boot the domain.

The configuration repository consists of the domains under the `/config` directory. The configuration repository may contain a variety of possible domain configurations. Each such domain is located under a separate subdirectory, with the subdirectory name being the name of that domain. When you specify *domain_name* you are thus specifying this subdirectory name. The subdirectory thus specified contains the XML configuration file (`config.xml`) and the security resources for that domain. The file `config.xml` specifies the configuration for that domain.

The domain configuration with which the Administration Server is started becomes the active domain. Only one domain can be active.

- If you want to override the default WebLogic Server stream handler for HTTP and HTTPS protocols, create an implementation of `java.net.URLStreamHandlerFactory` and specify it on the command line like this:

  `-Dweblogic.net.http.URLStreamHandlerFactory=`*`factory_class`*

  The factory class must be specified on the command line because a factory can be set only once for each URL protocol per JVM instance. Your factory class must have a `main()` method which can perform any one-time initialization required, and then must call `java.net.URL.setURLStreamHandlerFactory()` with an instance of the class.

  Your factory must implement the `createURLStreamHandler()` method, which returns an instance of your custom `java.net.URLStreamHandler` implementation for arguments of “http” and “https”. For other all other protocols, it should return null. See the Javadocs for `java.net.URL` for details about how the URLStreamHandlerFactory is located.

- You can also specify values for WebLogic Server configuration attributes on the command line. These values become the runtime value for that attribute, and any value stored in the persistent configuration is ignored. The format for setting a runtime value for a WebLogic Server attribute on the command line is:

  `-Dweblogic.`*`attribute=value`*

- The auto-deployment feature, which is turned on by default, does polling of the `\applications` directory of the active domain to detect changes in deployed applications. This feature only works on the Administration Server, because the

AppManager thread that polls the applications directory for changes is only created on Administration servers. This feature is not recommended for use in a production environment. If you want to ensure that the Administration Server is started with the auto-deployment feature disabled, include the following argument on the command line:

```
-Dweblogic.ProductionModeEnabled=true
```

- When you run the JSP compiler on Windows systems, output files names are always created with lower case names. To prevent this behavior, and preserve the case used in class names, set the system property, include the following argument on the command line:

  ```
  -Dweblogic.jsp.windows.caseSensitive=true
  ```

  See "Running JSPC on Windows Systems" in *Programming WebLogic JSP.*

Setting the Classpath Option

The following must be included as values to the `-classpath` option on the `java` command line:

- If you have installed a WebLogic Server service pack, include the following file:
  `/weblogic``/lib/weblogic_sp.jar`

  Depending on which WebLogic Server release, service pack, or patch that you have installed, this file might not exist on your system. Regardless of whether the file currently exists on your system, we recommend that you include *`weblogic`*`/lib/weblogic_sp.jar` on your classpath to ensure compatibility with any updates. You must add this file to the classpath before you add `weblogic.jar`.

- *`/weblogic`*`/lib/weblogic.jar`

- WebLogic Server comes with a trial version of an all-Java database management system (DBMS) called Cloudscape. If you will be using this DBMS, then you will need to include the following in the classpath:
  `/weblogic``/samples/eval/cloudscape/lib/cloudscape.jar`

- If you will be using WebLogic Enterprise Connectivity, you will need to include the following:
  `/weblogic``/lib/poolorb.jar`

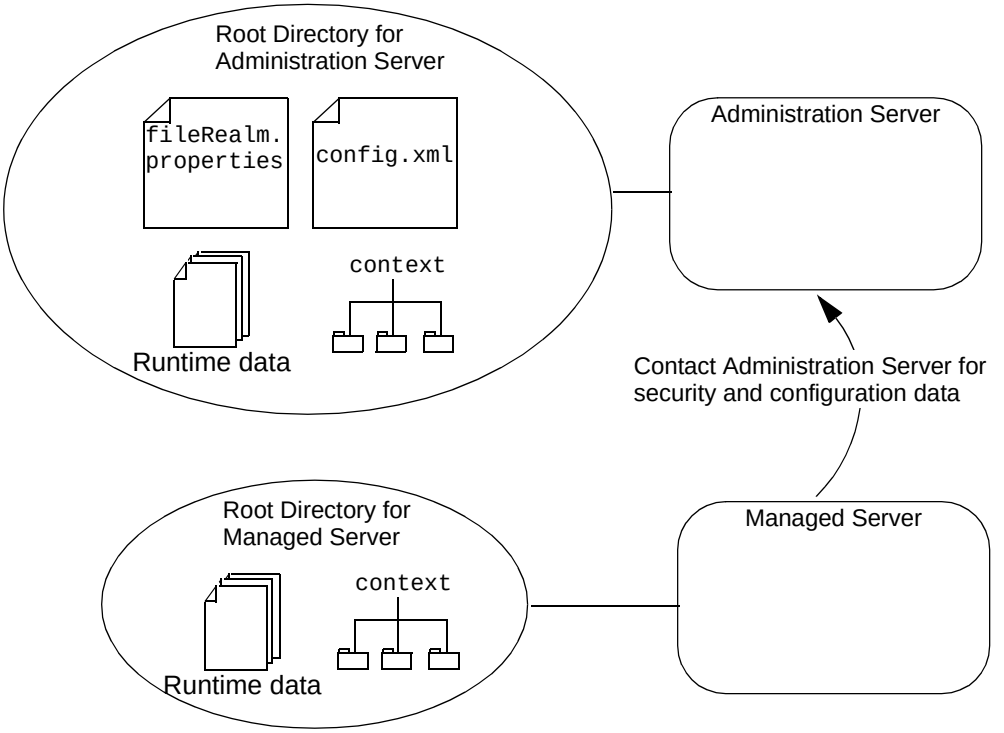
where *`weblogic`* is the directory where you installed WebLogic Server.

A Server's Root Directory

All instances of WebLogic Server use a root directory to store runtime data and to provide the context for any relative pathnames in the server's configuration. For example, if you specify `./MyLogFile` as the name and location of a server's log file, then the server creates a file named `MyLogFile` in its root directory.

In addition, an Administration Server uses its root directory as a repository for the domain's configuration data (such as `config.xml`) and security resources (such as `fileRealm.properties`). See Figure 2-1.

Figure 2-1 Root Directory for WebLogic Server Instances



A server's root directory must be below a directory named `config`. For example, the following directories are valid root directories:

```
c:\config\MyManagedRootDir
c:\config\MyDomain
```

But `c:\MyManagedRootDir` is not.

By convention, the root directory for an Administration Server is named after the domain. For example, if the domain is named `myDomain`, the root directory for the Administration Server is `c:\config\myDomain`.

Multiple instances of WebLogic Server can use the same root directory. However, if your server instances share a root directory, make sure that all relative filenames are unique. For example, if two servers share a directory and they both specify `.\MyLogFile`, then each server instance will overwrite the other's `.\MyLogFile`.

By default, the directory from which you start a WebLogic Server instance is the server's root directory. For example, if you run the `weblogic.Server` command from `c:\config\MyDomain`, then `c:\config\MyDomain` is the root directory.

If you want to start a server instance from a location other than the server's root directory, you can specify a different location by passing the following argument to the `weblogic.Server` startup command:

`-Dweblogic.RootDirectory=`*`path`*

where *`path`* specifes the directory above the `config\`*`server-root`* directory.

For example, if a server's root directory is `c:\myproject\config\Mydomain`, then specify `-Dweblogic.RootDirectory=c:\myproject`.

To make it easier to maintain your domain configurations and applications across upgrades of WebLogic Server software, it is recommended that the root directory not be the same as the installation directory for the WebLogic Server software.

Starting the Administration Server Using a Script

Sample scripts are provided with the WebLogic Server distribution that you can use to start WebLogic Servers. You will need to modify these scripts to fit your environment and applications. Separate sample scripts are provided for starting the Administration Server and the Managed Server. The scripts for starting the Administration Server are called `startWebLogic.sh` (UNIX) and `startWeblogic.cmd` (Windows). These scripts are located in the configuration subdirectory for your domain.

To use the supplied scripts:

- Pay close attention to classpath settings and directory names.

- Change the value of the variable `JAVA_HOME` to the location of your JDK.
- UNIX users must change the permissions of the sample UNIX script to make the file executable. For example:

  ```
  chmod +x startWebLogic.sh
  ```

- If you are going to run a Managed Server in the same domain on another machine (or on a multi-homed machine with the Administration Server), or want to be able to start and kill Managed Servers using the Node Manager, you will need to edit the WebLogic Server startup command by adding the argument to set the listen address of the Administration Server:

  `-Dweblogic.ListenAddress=`*`host`*

  where *`host`* is the DNS name or IP address of the Administration Server.

Restarting the Administration Server when Managed Servers are Running

For a typical production system it is recommended that you not deploy applications containing your critical business logic on the Administration Server. In such a scenario, the role of the Administration Server is that of configuring and monitoring the Managed Servers. If the Administration Server should become unavailable in such a configuration, the applications running on the Managed Servers can continue to process client requests.

When the Administration Server is started, it makes a copy of the configuration file that was used to boot the active domain. This is saved in the file

`install_dir``/config/`*`domain_name`*`/config.xml.booted`

where *`install_dir`* is the directory where you installed the WebLogic Server software and *`domain_name`* is the name of the domain. The Administration Server creates the `config.xml.booted` file only after it has successfully completed its startup sequence and is ready to process requests.

You should make a copy of this file so that you have a working configuration file that you can revert to if you need to back out of changes made to the active configuration from the Administration Console.

If the Administration Server goes down while Managed Servers continue to run, you do not need to restart the Managed Servers that are already running in order to recover management of the domain. The procedure for recovering management of an active domain depends upon whether you can restart the Administration Server on the same machine it was running on when the domain was started.

Restarting the Administration Server on the Same Machine

If you restart the WebLogic Administration Server while Managed Servers continue to run, the Administration Server can detect the presence of the running Managed Servers if you instruct the Administration Server to perform a discovery. To instruct the Administration Server to do a discovery of Managed Servers, enter the following argument on the command line when starting the Administration Server:

```
-Dweblogic.management.discover=true
```

The default value of this attribute is true. (If you omit this property, the Administration Servers still executes the discovery. Make sure, however, that this property is either not defined or not set to `false` in the command line you use to start WebLogic Server.) The configuration directory for the domain contains a file `running-managed-servers.xml` which is a list of the Managed Servers that the Administration Server knows about. When the Administration Server is instructed to perform discovery upon startup, it uses this list to check for the presence of running Managed Servers.

Restart of the Administration Server does not update the runtime configuration of the Managed Servers to take account of any changes made to attributes that can only be configured statically. WebLogic Servers must be restarted to take account of changes to static configuration attributes. Discovery of the Managed Servers does enable the Administration Server to monitor the Managed Servers or make runtime changes in attributes that can be configured dynamically.

Restarting the Administration Server on Another Machine

If a machine crash prevents you from restarting the Administration Server on the same machine, you can recover management of the running Managed Servers as follows:

1. Install the WebLogic Server software on the new administration machine (if this has not already been done).

Note: If you do not have Service Pack 2 of WebLogic Server 6.1 installed, the new administration machine must have the same host name as the machine that hosted the failed Administration Server.

2. Make your application files available to the new Administration Server by copying them from backups or by using a shared disk. Your application files should be available in the same relative location on the new file system as on the file system of the original Administration Server.

3. Make your configuration and security files available to the new administration machine by copying them from backups or by using a shared disk. These files are located in the `/config/myDomain` directory of the Administration Server, where *`myDomain`* is the name of the domain being managed by the Administration Server.

4. Restart the Administration Server on the new machine with the addition of the following argument on the command line:

   ```
   -Dweblogic.management.discover=true
   ```

   This argument will force the Administration Server to discover the presence of the Managed Servers that are running. (If you omit this property, the Administration Servers still executes the discovery. Make sure, however, that this property is either not defined or not set to `false` in the command line you use to start WebLogic Server.)

   When the Administration Server starts, it communicates with the managed servers and informs them that the Administration Server is now running on a different IP address.

Server Startup Process

When you start a WebLogic Server, it takes the following actions:

1. Retrieves its configuration and bootstrap security data.

   An Administration Server retrieves the configuration and security data from the domain's configuration files. A Managed Server contacts the Administration Server for its configuration and security data. If you set up SSL, a Managed Server uses its own set of certificate files, key files, and other SSL-related files and contacts the Administration Server for the remaining configuration and security data.

2. Starts its kernel-level services, which include logging and timer services.
3. Initializes subsystem-level services, which retrieve their configurations from MBeans. These services include the following:

- Security Service
- RMI Service
- Cluster Service
- IIOP Service
- Naming Service
- RMI Naming Service
- File Service
- JCA Container
- JDBC Container
- EJB Container
- Web Container
- Deployment Manager
- JMS Provider
- Remote Management
- Transaction Service

4. Deploys modules in the appropriate container and in the order that you specify in the WebLogic Server Administration Console.
5. Loads and runs any startup classes that are configured.

Adding a WebLogic Managed Server to the Domain

Before you can run a WebLogic Server as a managed server, you must first create an entry for that server in the configuration for the domain. To do this, do the following:

1. Start the Administration Server for the domain.
2. Invoke the Administration Console by pointing your browser at `http://`*`hostname`*`:`*`port`*`/console`, where *`hostname`* is the name of the machine where the Administration Server is running and *`port`* is the listen port number that you have configured for the Administration Server (default is 7001).
3. Create an entry for the server machine (Machines→Create a new machine) (if it is different than the Administration Server machine).

4. Create an entry for the new server (Servers→Create a new server). Set the machine for this Managed Server to the machine you just created an entry for. Each server must have a unique name — even if the servers are in different domains.

For more information on configuring servers, see Configuring WebLogic Servers and Clusters.

Starting a WebLogic Managed Server

WebLogic Managed Servers can be started in either of the following ways:

- Remotely from the Administration Console, using a Node Manager on the target machine where the Managed Server needs to be started.
- Locally, by invoking the server on the java command line in a command shell.

This section discusses how to start the WebLogic Managed Server locally. For information on setting up and using the Node Manager to start Managed Servers remotely, see Node Manager.

Note: If you right click on the name of a server in the left pane of the Administration Console, one of the options is **Start this server...**. This option can only be used to start a Managed Server if you have a Node Manager running on the machine where the Managed Server is located. For more information, see Node Manager.

Once you have added WebLogic Managed Servers to your configuration (see Adding a WebLogic Managed Server to the Domain), you can start the Managed Servers from the `java` command line. The command to start the WebLogic Server can be either typed in a command shell manually or it can be placed in a script to avoid retyping the command each time the server is started. For information on the sample scripts provided see Starting the WebLogic Managed Servers Using Scripts.

The main way in which the startup parameters for a Managed Server differ from an Administration Server is that you need to provide an argument that identifies the location of the Administration Server from which the Managed Server requests its configuration. A WebLogic Server started without this parameter runs as an Administration Server.

When starting a WebLogic Managed Server, you need to specify the parameters that you would specify when starting an Administration Server (see Starting the WebLogic Administration Server from the Command Line) but with the addition of the following:

- Specify the name of the server.

  When a WebLogic Managed Server requests its configuration information from the Administration Server, it identifies itself to the Administration Server by server name. This enables the Administration Server to respond with the appropriate configuration for that WebLogic Server. For this reason, you must also set the server name when starting a managed server. This can be specified by adding the following argument to the command line when starting the WebLogic Managed Server:

  `-Dweblogic.Name=`*`servername`*

- Specify the host name and listen port of the WebLogic Administration Server

  When starting a managed server, it is necessary to specify the host name and listen port of the Administration Server from which the managed server is to request its configuration. This can be specified by adding the following argument to the command line when starting the managed server:

  `-Dweblogic.management.server=`*`host`*`:`*`port`*

  or

  `-Dweblogic.management.server=http://`*`host`*`:`*`port`*

  where *`host`* is the name or IP address of the machine where the Administration Server is running and *`port`* is the Administration Server's listen port. By default the Administration Server's listen port is 7001.

  If you are using Secure Socket Layer (SSL) for communication with the Administration Server, the Administration Server must be specified as:

  `-Dweblogic.management.server=https://`*`host`*`:`*`port`*

  To use SSL protocol in communication between the Managed Servers and the Administration Server, you need to enable SSL on the Administration Server. For details on how to set this up, see Managing Security.

  Note: Any WebLogic Server that is started without specifying the location of the Administration Server is started as an Administration Server.

  Note: Because the Managed Server receives its configuration from the Administration Server, the Administration Server specified must be in the same domain as the Managed Server.

Informational Thread Dumps When Starting Clusters

If a Managed Server is configured as follows, it prints an informational thread dump to standard out during its startup cycle:

- Is a member of a cluster
- Prints messages of severity INFO and higher to standard out

  (Specifed in the Administration Console on the Server > Logging > General tab.)

- Logs remote exceptions

  (Specified in the Administration Console on the Server > Logging > Debugging tab.)

The thread dump is similar to the following abbreviated example:

```
Starting Cluster Service ....

<Nov 9, 2001 4:23:19 PM CST> <Info> <Connector> <Initializing J2EE
 Connector Service>
<Nov 9, 2001 4:23:20 PM CST> <Info> <Dispatcher> <Exception thrown
 by rmi server: 'weblogic.rmi.cluster.ReplicaAwareServerRef@9 -
 jvmid:
'-5643957423891326779S:10.1.2.151:[7001,7001,7002,7002,7001,7002,
-1]:***specific server***', oid: '9', implementation:
'weblogic.jndi.internal.RootNamingNode@28c19b''
javax.naming.NameNotFoundException: Unable to resolve
weblogic.transaction.resources.Server01. Resolved:
'weblogic.transaction.resources' Unresolved:'Server01' ; remaining
 name '' at
weblogic.jndi.internal.BasicNamingNode.newNameNotFoundException
(BasicNamingNode.java:802)
        at
weblogic.jndi.internal.BasicNamingNode.lookupHere(BasicNamingNode
.java:209)
        at
weblogic.jndi.internal.ServerNamingNode.lookupHere(ServerNamingNo
de.java:129)
.
.
.
```

If the Administration Server prints messages of INFO severity to standard out and logs remote exceptions, it prints thread dumps while members of clusters are starting.

You can ignore these thread dumps; they do **not** indicate an error in your configuration.

Note: The log files do not contain this or other thread dumps.

Starting the WebLogic Managed Servers Using Scripts

Sample scripts are provided with the WebLogic Server distribution that you can use to start WebLogic Servers. You will need to modify these scripts to fit your environment and applications. Separate scripts are provided for starting the Administration Server and the Managed Server. The sample scripts to start Managed Servers are called `startManagedWebLogic.sh` (UNIX) and `startManagedWebLogic.cmd` (Windows). These scripts are located in the configuration subdirectory for your domain. These are templates that you will need to modify to create your own start scripts.

To use the supplied scripts:

- Pay close attention to classpath settings and directory names.
- Change the value of the variable `JAVA_HOME` to the location of your JDK.
- UNIX users must change the permissions of the sample UNIX script to make the file executable. For example:

  ```
  chmod +x startManagedWebLogic.sh
  ```

There are two ways to start the Managed Server using the script:

- If you set the value of the environment variables `SERVER_NAME` and `ADMIN_URL`, you do not need to provide these as arguments when invoking the start script. `SERVER_NAME` should be set to the name of the WebLogic Managed Server that you wish to start. `ADMIN_URL` should be set to point to the host (host name or IP address) and port number where the Administration Server is listening for requests (default is 7001). For example:

  ```
  set SERVER_NAME=bigguy
  set ADMIN_URL=peach:7001
  startManagedWebLogic
  ```

- You can invoke the start script and pass the name of the Managed Server and the URL for Administration Server on the command line:

  ```
  startManagedWebLogic server_name admin:url
  ```

where *server_name* is the name of the Managed Server you are starting and *admin_url* is either `http://host:port` or `https://host:port` where *host* is the host name (or IP address) of the Administration Server and *port* is the port number for the Administration Server.

Stopping WebLogic Servers from the Administration Console

If you right click on a server in the left pane of the Administration Console, you will see two options, **Kill this server...** and **Stop this server...**. If you select the **Kill this server...** option, the Administration Server sends a request to the Node Manager running on the machine where the Managed Server is running. The Node Manager then kills the target WebLogic Server process. The **Kill this server...** option cannot be used to shut down the Administration Server. The **Kill this server...** option assumes you have a Node Manager running on the machine where the target Managed Server is. For information on setting up and starting Node Manager, see Node Manager.

If you select the **Stop this server...** option, the Administration Server sends an administrative shutdown request to the selected server. The Node Manager is not used in that case. Unlike the **Kill this server...** option, the **Stop this server...** option can be used to shutdown the Administration Server.

Because the **Stop this server...** option uses the administrative capability of a Managed Server to initiate a shutdown, it can only be used if the server is alive and responding to administrative requests. The **Kill this server...** option would be typically used in situations where the target Managed Server is hung or not responding to administrative requests from the Administration Server.

Shutting Down a Server from the Command Line

You can also shut down a WebLogic Server from the command line with the following command:

```
java weblogic.Admin -url host:port SHUTDOWN -username adminname
-password password
```

where:

- *host* is the name or IP address of the machine where the WebLogic Server is running.
- *port* is the WebLogic Server's listen port (default is 7001).
- *adminname* designates a user that is a member of the Console Access Control List (ACL) (or a member of a group that is a member of the Console ACL) for the target WebLogic Server. Default member of the Console ACL is `system`.
- *password* is the password for *adminname*.

Setting Up a WebLogic Server Instance as a Windows Service

If you want a WebLogic Server instance to start automatically when you boot a Windows host, you can set up the server as a Windows service.

For each server that you set up as a Windows service, WebLogic Server creates a key in the Windows Registry under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`. The registry entry contains such information as the name of the server and other startup arguments. When you start the Windows host, it passes the information in the registry to the JVM.

Before you can set up a server instance to run as a Windows service, you must create the server in a domain.

The following tasks describe setting up and managing WebLogic Server Windows services:

- "Setting Up Windows Services: Main Steps" on page 2-23
- "Removing WebLogic Server as a Windows Service" on page 2-33
- "Changing Passwords for a Server Installed as a Windows Service" on page 2-33

Setting Up Windows Services: Main Steps

The following main steps assume that you created at least one WebLogic Server instance in the default domain that is installed with WebLogic Server:

1. In the default Windows-service installation script, specify the name of the Windows Service, the name of the server instance, and the WebLogic Server password for starting the server instance. For more information, refer to "Specifying the Name of the Server Instance and the Windows Service" on page 2-23.
2. If you are installing a Managed Server as a Windows service, add a command option that specifies the location of the domain's Administration Server. For more information, refer to "Specifying the Location of the Administration Server" on page 2-24.
3. If you set up both an Administration Server and a Managed Server to run as Windows services on the same computer, make sure the Managed Server starts only after the Administration Server finishes its startup cycle. For more information, refer to "Require Managed Servers to Start After the Administration Server" on page 2-25.
4. If you want a server instance to shut down gracefully when you use the Windows Control Panel to stop the Windows service, include a command option that causes a server instance to shut down gracefully. For more information, refer to "Enabling Graceful Shutdowns from the Windows Control Panel" on page 2-27.
5. If you want to see the messages that a server instance prints to standard out and standard error (including stack traces and thread dumps), redirect standard out and standard error to a file. For more information, refer to "Redirecting Standard Out and Standard Error to a File" on page 2-28.
6. If you have created additional Java classes that you want the WebLogic Server instance to invoke, add them to the server's classpath. For more information, refer to "Adding Classes to the Classpath" on page 2-31.
7. Run the Windows service installation script. For more information, refer to "Run the Installation Script" on page 2-32.

Specifying the Name of the Server Instance and the Windows Service

To specify the name of the server instance that you want to run as a Windows service:

1. Make a backup copy of *weblogic*\config*mydomain*\installNTService.cmd (where *weblogic* is the directory where WebLogic Server was installed and *mydomain* is the domain you created when you installed WebLogic Server).
2. Open installNTService.cmd in a text editor.
3. In the line that starts with set CMDLINE=, change the value of -Dweblogic.Name= to specify the name of the server that you want to start as a Windows service. For example, -Dweblogic.Name=myserver.
4. The last command in the script invokes the beasvc utility. In this command, do the following:

   a. Change the value of -svcname to specify a unique name for the service. For example, use the name of the domain and server instance: -svcname:mydomain_myserver

   b. At the end of the beasvc command, append -password:*password* where *password* is the password for the system user.

      When you install the Windows service, the beasvc utilty encrypts the password and stores the encrypted value in the Windows registry. After you finish installing this server as a Windows service, remove the password from the installNTService.cmd script.

   The modified beasvc command will resemble the following command:

```
"D:\bea\wlserver6.1\bin\beasvc" -install
-svcname:mydomain_myserver
-javahome:"D:\bea\jdk131" -execdir:"D:\bea\wlserver6.1"
-extrapath:"D:\bea\wlserver6.1\bin" -cmdline:%CMDLINE%
-password:weblogic
```

Specifying the Location of the Administration Server

If you are installing a Managed Server as a Windows service, you must modify the installNTService.cmd script so that it specifies the location of the Administration Server. A Managed Server must contact the Administration Server to receive its configuration information.

To specify the location of the Administration Server:

1. Open installNTService.cmd in a text editor.

2. In the line that starts with `set CMDLINE=`, add the following argument after the `-Dweblogic.Name` argument:
   `-Dweblogic.management.server=http://host:port`

   where *host* is the name or IP address of the machine where the Administration Server is running and *port* is the Administration Server's listen port

   If you are using Secure Socket Layer (SSL) for communication with the Administration Server, the Administration Server must be specified as:

   `-Dweblogic.management.server=https://host:port`

Require Managed Servers to Start After the Administration Server

If you set up both an Administration Server and a Managed Server to run as Windows services on the same computer, you can specify that the Managed Server starts only after the Administration Server.

To require a Managed Server to start after the Administration Server Windows service:

1. Make a backup copy of *weblogic*`\config\`*mydomain*`\installNTService.cmd` (where *weblogic* is the directory where WebLogic Server was installed and *mydomain* is the domain you created when you installed WebLogic Server).
2. If you have already installed the Administration Server as a Windows Service, remove the service. For more information, refer to "Removing WebLogic Server as a Windows Service" on page 2-33.
3. Before you install (or reinstall) the Administration Server as a Windows Service, do the following:

   a. In a text editor, open the *weblogic*`\config\`*mydomain*`\installNTService.cmd` script.

   b. Add the following argument to the command that invokes the `beasvc` utility:

      `-delay:`*delay_milliseconds*
      This specifies the number of milliseconds to wait before the Windows Service Control Manager (SCM) changes the service status from `SERVER_START_PENDING` to `STARTED`.

      For example, if your Administration Server requires 2 minutes to complete its startup cycle and begin listening for requests, then specify `-delay=120000`. When you boot the Windows host computer, the Windows SCM reports a status

of SERVER_START_PENDING for 2 minutes. Then it changes the status to STARTED.

The modified beasvc invocation for the Administration Server will resemble the following:

```
"D:\bea\wlserver6.1\bin\beasvc" -install
-svcname:mydomain_myAdminServer
-javahome:"D:\bea\jdk131"
-delay:120000
-execdir:"D:\bea\wlserver6.1"
-extrapath:"D:\bea\wlserver6.1\bin" -cmdline:%CMDLINE%
```

For more information about beasvc, enter the following command at a command prompt: *weblogic*\server\bin\beasvc -help, where *weblogic* is the directory in which you installed WebLogic Server.

4. Install the Administration Server Windows service.
5. Before you install the **Managed Server** as a Windows service, do the following:
   a. In a text editor, open the *weblogic*\config*mydomain*\installNTService.cmd script.
   b. Add the following argument to the command that invokes the beasvc utility:

      -depend:*Administration-Server-service-name*

      where *Administration-Server-service-name* is the name of the Administration Server Windows service. To verify the service name, look on the Windows Services Control Panel.

      With this option, the Windows SCM will wait for the Administration Server Windows service to report a status of STARTED before it starts the Managed Server Windows service.

      For example, the modified beasvc invocation for the Managed Server will resemble the following:

```
"D:\bea\wlserver6.1\bin\beasvc" -install
-svcname:mydomain_myManagedServer
-javahome:"D:\bea\jdk131"
-depend:"mydomain_myAdminServer"
-execdir:"D:\bea\wlserver6.1"
-extrapath:"D:\bea\wlserver6.1\bin" -cmdline:%CMDLINE%
```

You can also add the `-delay:`*`delay_milliseconds`* option to a Managed Server Windows service if you want to configure when the Windows SCM reports a status of `STARTED` for the service.

Enabling Graceful Shutdowns from the Windows Control Panel

By default, if you use the Windows Control Panel to stop a server instance, the Windows Service Control Manager (SCM) kills the server's Java Virtual Machine (JVM). If you kill the JVM, the server immediately stops all processing. Any session data is lost. If you kill the JVM for an Administration Server while the server is writing to the `config.xml` file, you can corrupt the `config.xml` file.

To enable graceful shutdowns from the Windows Control Panel:

1. In a text editor, open the *`weblogic`*`\config\`*`mydomain`*`\installNTService.cmd` script.
2. Add the following argument to the command that invokes the `beasvc` utility:

   `–stopclass:weblogic.Server`

   With this argument, when you stop a server Windows service from the Windows Control Panel, the Windows SCM invokes the `stop()` method of the server's `ServerRuntime` MBean. This management method gracefully shuts down a server.

   For example, the modified `beasvc` invocation for the server instance will resemble the following:

   ```
   "D:\bea\wlserver6.1\bin\beasvc" -install
   -svcname:mydomain_myserver
   -javahome:"D:\bea\jdk131"
   -stopclass:weblogic.Server
   -execdir:"D:\bea\wlserver6.1"
   -extrapath:"D:\bea\wlserver6.1\bin" -cmdline:%CMDLINE%
   ```

   For more information about `beasvc`, enter the following command at a command prompt: *`weblogic`*`\server\bin\beasvc -help`, where *`weblogic`* is the directory in which you installed WebLogic Server.
3. Consider modifying the default timeout value that the Windows SCM specifies.

   By default, when you use the Windows 2000 Control Panel to stop a Windows service, the Windows SCM waits 30 seconds for the service to stop before it kills the service and prints a timeout message to the System event log.

If you use `-stopclass` to gracefully shut down a server, 30 seconds might not be enough time for the server to gracefully end its processing.

To configure a timeout period on Windows 2000, create a `REG_DWORD` registry value named `ServicesPipeTimeout` under the following registry key: `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control`

The key value must be in milliseconds.

This value is read from the registry during the startup of the Windows operating system and it affects all services that are installed.

4. Save your changes to the `installNTService.cmd` script.

Redirecting Standard Out and Standard Error to a File

By default, when you install a WebLogic Server instance as a Windows service, you cannot see the messages that the server or JVM print to standard out and standard error.

To view these messages for a server instance that is installed as a Windows service, you must redirect standard out and standard error to a file:

1. Open `installNTService.cmd` in a text editor.
2. In `installNTService.cmd`, the last command in the script invokes the `beasvc` utility. At the end of the `beasvc` command, append the following command option:

   `-log:"`*`pathname`*`"`
   where *`pathname`* is a fully qualified path and filename of the file that you want to store the server's standard out and standard error messages.

   The modified `beasvc` command will resemble the following command:

   ```
   "D:\bea\wlserver6.1\bin\beasvc" -install
   -svcname:mydomain_myserver
   -javahome:"D:\bea\jdk131" -execdir:"D:\bea\wlserver6.1"
   -extrapath:"D:\bea\wlserver6.1\bin" -cmdline:%CMDLINE%
   -password:weblogic
   -log:"d:\bea\wlserver6.1\config\mydomain\myserver-stdout.txt
   ```

3. By default, every 24 hours the Windows service archives messages to a file named *`pathname-yyyy_mm_dd-hh_mm_ss`*. New messages collect in the file that you specified in the previous step.

For information on changing the default behavior, see "Changing the Default Rotation Criteria" on page 2-29.

After you install the service and restart the Windows host, to view the messages that the server and JVM write to standard out or standard error, do one of the following:

- Make a copy of the file that you specified and view the copy. The Windows file system cannot write to files that are currently opened.
- To view the messages as they are being printed to the file, open a command prompt and use the DOS command `tail -f stdout-filename`.

Changing the Default Rotation Criteria

By default, every 24 hours the Windows service archives messages to a file named *`pathname-yyyy_mm_dd-hh_mm_ss`*. New messages collect in the file that you specified when you set up the service.

You can change the time interval or you can set up rotation to occur based on the size of the message file instead of a time interval.

To change the default criteria at which the Windows service rotates message files:

1. If the Windows service is running, shut it down.
2. Edit the file you specified in the `-log: pathname` argument. If a file does not exist, create one.

   For example, if you issued the example command in step 2. in the previous section, create a file named `d:\bea\wlserver6.1\config\mydomain\myserver-stdout.txt`.
3. Do one of the following:
   - If you want the Windows service to rotate the message file at a specific time interval regardless of file size, add the following statements at the top of the file, each statement on a separate line (make sure to press the Enter or Return key after typing the last line):

     ```
     # ROTATION_TYPE = TIME
     # TIME_START_DATE = date-in-required-format
     # TIME_INTERVAL_MINS = number-of-minutes
     ```

     where `TIME_START_DATE` specifies when the first rotation should take place. If the specified time has already passed, the first rotation occurs when the time interval specified in `TIME_INTERVAL_MINS` expires. You must use the

following format to specify the start time: *Month Day Year Hour:Minutes:Seconds*
where *Month* is the first 3 letters of a Gregorian-calendar month as written in English
Day is the 2-digit day of the Gregorian-calendar month
Year is the 4-digit year of the Gregorian calendar
Hour:*Minutes*:*Seconds* expresses time in a 24-hour format

and TIME_INTERVAL_MINS specifies how frequently (in minutes) the Windows service rotates the file.

For example:

```
# ROTATION_TYPE = TIME
# TIME_START_DATE = Jul 17 2003 05:25:30
# TIME_INTERVAL_MINS = 1440
```

When the time interval expires, the Windows service saves the file as *pathname-yyyy_mm_dd-hh_mm_ss*. It then creates a new file named *pathname*. This new file, which contains all of the headers that you specified originally, collects new standard out and standard error messages.

If you specify `# ROTATION_TYPE = TIME` but do not include the other lines, the Windows service rotates the message file every 24 hours.

- If you want the Windows service to rotate the message file after the file grows beyond a specified size, add the following statements at the top of the file, each statement on its own line (make sure to press the Enter or Return key after typing the last line):

  ```
  # ROTATION_TYPE = SIZE
  # SIZE_KB = file-size-in-kilobytes
  # SIZE_TRIGGER_INTERVAL_MINS = polling-interval
  ```

  where SIZE_KB specifies the minimal file size (in kilobytes) that triggers the Windows service to move messages to a separate file.

  and SIZE_TRIGGER_INTERVAL_MINS specifies (in minutes) how frequently the Windows service checks the file size. If you do not include this header, the Windows service checks the file size every 5 minutes.

  For example:

  ```
  # ROTATION_TYPE = SIZE
  # SIZE_KB = 1024
  # SIZE_TRIGGER_INTERVAL_MINS = 3
  ```

When the Windows service checks the file size, if the file is larger than the size you specify, it saves the file as *pathname-yyyy_mm_dd-hh_mm_ss*. It then creates a new file named *pathname*. This new file, which contains all of the headers that you specified originally, collects new standard out and standard error messages.

If you specify `# ROTATION_TYPE = SIZE` but do not include the other lines, the Windows Service checks the size of the message file every 5 minutes. If the file is larger than 1 megabytes, it rotates the file.

Printing Thread Dumps to Standard Out

To cause the WebLogic Server instance to print a thread dump to standard out, do either of the following:

- Use the `weblogic.Admin THREAD_DUMP` command. For more information, refer to "THREAD_DUMP" on page -24.
- Open a command prompt and enter the following command:

  weblogic`\bin\beasvc -dump -svcname:`*service-name*
  where *weblogic* is the directory in which you installed WebLogic Server and *service-name* is the Windows service that is running a server instance.

  For example:
  `D:\bea\wlserver6.1\bin\beasvc -dump -svcname:mydomain_myserver`

Adding Classes to the Classpath

The **classpath** is a declaration of the location of Java classes that a JVM can invoke. When you install a server instance as a Windows service, the `installNTService.cmd` script specifies all classes required to run a server instance. If you want to extend WebLogic Server by adding your own Java classes, you must add them to the classpath.

To add classes to the classpath:

1. Make a backup copy of *weblogic*`\config\`*mydomain*`\installNTService.cmd` (where *weblogic* is the directory where WebLogic Server was installed and *mydomain* is the domain you created when you installed WebLogic Server).
2. Open `installNTService.cmd` in a text editor.
3. In the line that starts with `set CLASSPATH` statement, add your Java classes.

For example if you archived your class in a file named `c:\myJar`, the modified statement will be as follows:

```
set
CLASSPATH=.;D:\bea\wlserver6.1\lib\weblogic_sp.jar;D:\bea\wlser
ver6.1\lib\weblogic.jar;c:\myJar
```

Note: Win32 systems have a 2K limitation on the length of the command line. If the classpath setting for the Windows service startup is very long, the 2K limitation could be exceeded.

To work around this limitation:

a. Place the value of the `set CLASSPATH` command in a separate text file and save the text file in the *`weblogic`*`\server\bin` directory.

b. In the *`weblogic`*`\config\`*`mydomain`*`\installNTService.cmd` script, find the `set CMDLINE` command.

c. Within the `set CMDLINE` command, replace the `-classpath \"%CLASSPATH%\"` option with the following option:

   `-classpath @`*`filename`*
   where *`filename`* is the name of the file that contains the classpath values.

   For example:

   ```
   set CMDLINE="-ms64m -mx64m -classpath @myClasspath.txt
   -Dweblogic.Domain=mydomain -Dweblogic.Name=myserver
   -Djava.security.policy==\"D:\bea\wlserver6.1/lib/weblogic.po
   licy\" -Dbea.home=\"D:\bea\" weblogic.Server"
   ```

4. Save your changes to the `installNTService.cmd` script.

Run the Installation Script

1. Open a command prompt and change to *`weblogic`*`\config\`*`mydomain`*.
2. Enter `installNTService.cmd`.

   The command prompt runs the script as a `batch` file.

   If the script runs successfully, it creates a Windows service named *`DOMAIN_NAME_SERVER_NAME`* and prints a line to standard out that is similar to the following:

   ```
   mydomain_myserver installed.
   ```

By default, standard out is the command prompt in which you run the server-specific `batch` file.

3. Remove your password from the `installNTService.cmd` script. Leaving this password on the filesystem in an unencrypted format opens a security vulnerability.

Removing WebLogic Server as a Windows Service

To remove the WebLogic Server as a Windows service, do the following:

1. Navigate to the *`weblogic`*`\config\`*`mydomain`* directory (where *`weblogic`* is the directory where WebLogic Server was installed and *`mydomain`* is the subdirectory with your domain's configuration).
2. Open the script `uninstallNTService.cmd` in a text editor.
3. Change the value of `-svcname:` to specify the name of the service that you want to remove.
4. Save and execute `uninstallNTService.cmd`.

Changing Passwords for a Server Installed as a Windows Service

If you install the Default Server as a Windows service, the system password that you entered during installation of the WebLogic Server software is used when creating the service. If this password is later changed, you must do the following:

1. Uninstall the WebLogic Server as a Windows service using the `uninstallNTService.cmd` script (located in the directory *`install_dir`*`/config/`*`domain_name`* where *`install_dir`* is the directory where you installed the product).
2. The `installNTservice.cmd` script contains the following command:

```
rem *** Install the service
"C:\bea\wlserver6.1\bin\beasvc" -install -svcname:myserver
-javahome:"C:\bea\jdk130" -execdir:"C:\bea\wlserver6.1"
```

```
-extrapath:"C\bea\wlserver6.0\bin" -cmdline:
%CMDLINE%
```

You must append the following to the command:

`-password:"`*`your_password`*`"`

where *`your_password`* is the new password.

3. Execute the modified `installNTservice.cmd` script. This will create a new service with the updated password.

Registering Startup and Shutdown Classes

WebLogic Server provides a mechanism for performing tasks whenever a WebLogic Server starts up or gracefully shuts down. A startup class is a Java program that is automatically loaded and executed when a WebLogic Server is started or restarted. For more information about when a server loads and runs startup classes, refer to "Server Startup Process" on page 2-15.

Shutdown classes work the same way as startup classes. A shutdown class is automatically loaded and executed when the WebLogic Server is shut down either from the Administration Console or using the `weblogic.admin shutdown` command.

In order for your WebLogic Servers to use startup or shutdown classes, it is necessary to register these classes, which you can do from the Administration Console.

You can register a startup or shutdown class by doing the following:

1. Access the Startup & Shutdown table from the domain tree (in the left pane) in the Administration Console. This table provides options for creating entries for shutdown or startup classes in the domain configuration.
2. Provide the class name and necessary arguments, if any, on the Configuration tab page for the startup or shutdown class you are adding.

See the Administration Console Online Help for more information on:

- Startup classes
- Shutdown classes

CHAPTER

3 Node Manager

The following sections describe how to use the Node Manager:

- Overview of Node Manager
- Setting Up Node Manager
- Platform Support for Node Manager
- Starting the Node Manager from the Command Line
- Starting the Node Manager Using Start Scripts
- Remote Starting and Killing of Managed Servers
- Setting Up Node Manager as a Windows Service

Overview of Node Manager

Node Manager is a Java program that enables you to start and kill WebLogic Managed Servers remotely from the Administration Console. Node Manager is a separate Java program that is provided with the WebLogic Server software.

Using the Node Manager to kill a remote Managed Server is an alternative to the facility for stopping Managed Servers provided in the Administration Console. Killing a remote server process is intended for situations where the server is hung or is unresponsive.

In order to enable remote starting of Managed Servers, you need to configure and run one Node Manager on each machine where your Managed Servers will be running. A single Node Manager process on a machine can handle remote starting and killing of

all the Managed Servers on that machine. To ensure availability of Node Manager, Node Manager should be configured as a daemon on UNIX machines or as a Windows NT service on Windows NT machines. This ensures that the Node Manager is available for starting the Managed Servers on that machine.

When the Node Manager is running, it can start or kill any Managed Server installed and configured on its machine at the request of the Administration Server. All communication between Node Manager and the Administration Server uses the Secure Socket Layer protocol.

Node Manager Logs

When you start the WebLogic Server, various startup or error messages may be printed to `STDOUT` or `STDERROR`. These messages are also displayed in the right pane of the Administration Console during startup of a server. At other times, these files can be retrieved by right clicking on the server in the left pane of the Administration Console and selecting the option **Get StdOut for this server** or **Get StdErr for this server**.

The Node Manager saves these messages in files in the Node Manager log file directory. By default this directory is called `NodeManagerLogs` and is created in the directory where you start the Node Manager. If you want to change the name of the directory, this can be done from the command line when starting the Node Manager. For more information, see Command-Line Arguments.

A separate log file subdirectory is created for each Managed Server started by the Node Manager on that machine. The logs stored in this directory include:

`servername``.pid`
: This saves the process ID of the Managed Server named *`servername`*. This is used by the Node Manager to kill the server process when requested by the Administration Server to do so.

`config`
: This saves startup configuration information passed to the Node Manager from the Administration Server when starting a Managed Server.

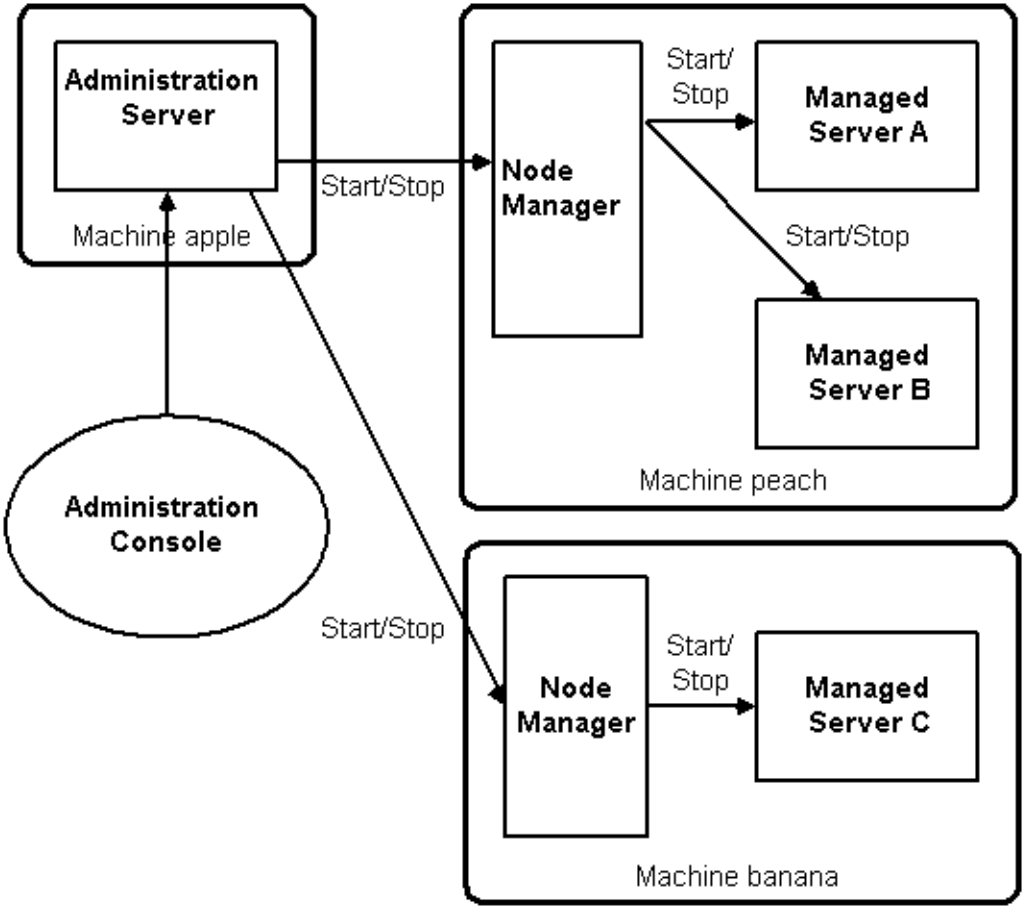
`servername``-output.log`
: This saves the information printed to `StdOut` when an attempt is made by the Node Manager to start the Managed Server named *`servername`*. If a new attempt is made to start the server, this file is renamed by appending `_PREV` to the file name.

servername`-error.log`
: This saves information printed to `StdErr` when an attempt is made by the Node Manager to start the Managed Server named *servername*. If a new attempt is made to start the server, this file is renamed by appending `_PREV` to the file name.

The Node Manager logs are also stored by the Administration Server in temporary files in a directory called `/config/NodeManagerClientLogs` on the Administration Server machine. There is a subdirectory for each Managed Server which you have attempted to start via the Node Manager. Each log in these subdirectories corresponds to an attempt to carry out some action, such as starting or killing the server. The name of the log file includes a timestamp that indicates the time at which the action was attempted. It is recommended that you periodically delete the accumulated client logs from past actions using the Node Manager.

The Node Manager architecture is illustrated in Figure 3-1.

Figure 3-1 Node Manager Architecture



Setting Up Node Manager

All communication between Node Manager and the Administration Server uses the Secure Socket Layer protocol, which provides authentication and encryption. Client authentication is enforced to ensure mutual authentication is used in all communication between the Administration Server and the Node Manager. For further

security, Node Manager also uses a list of trusted hosts; only commands from an Administration Server on one of these hosts will be accepted. To configure the Node Manager you will need to edit the trusted hosts file to add one line for each machine with an Administration Server that can send commands to this Node Manager. By default, the trusted hosts file is named `nodemanager.hosts` and is installed under the `\config` directory. By default this file contains two entries:

```
localhost
127.0.0.1
```

You can change the name of the file where Node Managed looks for the list of trusted hosts on the command line. For more information, see Command-Line Arguments.

You can use either the IP address or DNS name of each trusted host. However, if you use a DNS name, you will need to enable reverse DNS lookup when starting the Node Manager. To do this, you would use the following command-line argument:

```
-Dweblogic.nodemanager.reverseDnsEnabled=true
```

By default reverse DNS lookup is disabled.

In a typical production environment the Node Manager will not be running on the same machine as the Administration Server. Therefore, you will need to edit the trusted hosts file so that it lists only the machines where you will be running an Administration Server that will be starting or killing the Managed Servers on that machine. Each entry in the trusted hosts file consists of a single line that is either the DNS host name or IP address of the machine of an Administration Server machine.

Note: In order for the Node Manager to be able to communicate with the Administration Server when it starts Managed Servers, the listen address of the Administration Server must have been set to a DNS name or IP address when the Administration Server was started.

Setting Up Node Manager for Secure Socket Layer Protocol

The Node Manager uses the Secure Socket Layer (SSL) protocol in its communication with the Administration Server. To provide security in communication between the Node Manager and the Administration Server, two-way SSL authentication is used.

Authentication requires use of the public key infrastructure. This includes a private key as well as a certificate. The certificate typically contains the public key of the user and is signed by the issuer of the certificate to authenticate the binding between the user name and the enclosed public key.

Node Manager uses certificates in X509 format. The private keys used with Node Manager conform to the Private Key Cryptography Standards (PKCS) #5 and #8. PKCS #5 is an the password-based encryption standard and describes the method for encrypting private keys with a password. PKCS #8 is the private key syntax standard and specifies the characteristics of the private key.

The various pieces of the public key infrastructure used by Node Manager differ from the format used by WebLogic Server digital certificates, which conform to an earlier standard. The main differences are:

- Node Manager uses a single certificate file that contains the private key as well as the certificate containing the public identity of the user.
- The private key used by the Node Manager must be password-protected, in conformity with the PKCS #5/#8 standards.

A demonstration certificate for use with Node Manager is provided with the WebLogic software. This is located at `/config/demo.crt`. It is recommended that you get a new certificate for a production environment.

The steps for setting up digital certificates for use with Node Manager are as follows:

Step 1: Obtain a Digital Certificate and Private Key

Use one of the following methods to obtain digital certificates for use with Node Manager:

- Obtain a private key and X509-format digital certificates using the instructions in Obtaining a Private Key and Digital Certificate in the *Administration Guide*. If the private key is not in PKCS #5/#8 format, you will need to convert it using the WebLogic Server key conversion tool, as described in Step 2. If you obtain a private key in PKCS #5/#8 format, skip to Step 3: Merging the Certificates into a Single Certificate File.
- Use the WebLogic Server certificate generator to generate certificates, then can convert them for use with Node Manager.

Step 2: Converting a WebLogic-Style Private Key

If you want to use WebLogic-style certificates with Node Manager, you will first need to convert the private key to the newer PKCS #5/#8 format. A tool to do this is provided with the WebLogic software.

The tool for converting WebLogic-style certificates for use with Node Manager is called `wlkeytool` and is located in:

- The `/bin` directory under the root WebLogic installation directory on Windows systems
- The `/lib` directory of the root WebLogic installation directory on UNIX systems

The syntax for using `wlkeytool` is:

`wlkeytool` *`old_key new_key`*

You will be prompted for the private key password to unlock the old key. Press return if it has no password. You will then be prompted to enter a password to use in encrypting the new key. A password is required for use with Node Manager.

For example:

```
wlkeytool demokey.pem demokey_new
```

Step 3: Merging the Certificates into a Single Certificate File

WebLogic Server uses separate certificate files (with `.pem` file extension) for the private key, the public key, and the certificate authority (or a series of certifying authorities). In addition to the requirement that the private key be password-protected PKSC #5/#8 format, Node Manager combines these components of the certificate into a single certificate file (with a `.crt` file extension).

Note: Although the components of the user SSL identity are combined in a single file, the private key information is not transmitted between servers.

The three components are simply concatenated into a single file with a `.crt` extension. For example:

```
cat demokey_new democert.pem ca.pem > demo.crt
```

In this example, `ca.pem` is the WebLogic certificate authority file and is identical in content to the default trustedCerts file, `trusted.crt`, and `democert.pem` is the public key file. The file `demokey_new` is the result of running `wlkeytool` on `demokey.pem`, as described in Step 2: Converting a WebLogic-Style Private Key.

For more information about digital certificates and Secure Sockets Layer, see Managing Security.

Setting Up the Administration Server to Use Node Manager

To configure the Administration Server to use Node Manager to start and stop WebLogic Managed Servers, there are several steps you need to carry out. You can accomplish these tasks using the WebLogic Administration Console.

Step 1: Create a Configuration Entry for the Machine

You need to create an entry in the domain configuration for each machine on which you have installed Managed Servers. To do this, do the following:

1. With the Administration Server running, invoke the Administration Console (if it isn't already running).
2. Select Machines in the left pane to display the machines table.
3. Select the Create a new Machine link (or Create a new UNIX Machine) at the top of the table.
4. Fill in the information for the machine and click Apply to create the new machine entry.

Step 2: Configure Node Manager on Each Machine

For each machine where you want to use Node Manager, modify the configuration entry for that machine accordingly:

1. In the Administration Console, select Machines→*machine_name*→Node Manager, where *machine_name* is the name of the machine on which the Node Manager is to run.

2. Fill in the fields on the Node Manager tab:
   - The Listen Address is either the host name or IP address where the Node Manager will be expecting requests from the Administration Server. This is the listen address you specify when Starting the Node Manager.
   - The Listen Port number must also match the port number you use when starting the Node Manager on that machine.
   - The certificate used by the Administration Server to talk to this Node Manager. The default certificate is `config/demo.crt`. It is recommended that you get a new certificate for a production environment. See Managing Security for information on how to do this.
   - The certificate password is not displayed because it is encrypted. If you change the certificate that is being used by Node Manager, you will need to change the password to match the password that was used to encrypt the private key in the new digital certificate.
   - The trustedCerts file contains the list of certificate authorities that are recognized. The default is `config/trusted.crt`. The certificate authority referred to in the digital certificate you are using must be listed in this file.
3. Click Apply.

Step 3: Configure Startup Information for Managed Servers

For Node Manager to start the WebLogic Managed Server, it must be provided with the startup parameters and options you want to use when starting that Managed Server. To set this up:

1. Invoke the Administration Console, if it isn't already running.
2. In the Administration Console, select *server_name*→Configuration→Remote Start, where *server_name* is the name of the Managed Server.

   There are five fields here that can be filled out to provide configuration information that the Administration Server will use when starting the target Managed Server:

   Note: If you don't specify values for these fields and attempt to start the target server from the Administration Console, the Node Manager will attempt to start the target server with the values for these attributes that were used when starting the Node Manager. The Node Manager will be able to start

the Managed Server in that case if you specified the required values on the command line when starting the Node Manager.

- BEA Home

  You can specify the BEA Home directory. This is the root directory under which all BEA products and licenses were installed for the target Managed Server.

- Root Directory

  This is the root directory where the WebLogic software was installed.

- Class Path

  The classpath for starting the Managed Server.

  At a minimum you will need to specify the following values for the classpath option:

  `/weblogic/lib/weblogic_sp.jar`

  `/weblogic/lib/weblogic.jar`

  You may need to also include the path to the root directory where you installed the JDK that is used when starting the Managed Server. For more information about setting the classpath, see Starting and Stopping WebLogic Servers.

- Arguments

  In the Arguments field enter any other arguments you want passed to the startup command.

  For example, you may want to set the maximum and minimum Java heap memory. Using the `-ms64m` and `-mx64m` options would specify a default allocation of 64 megabytes of Java heap memory to the WebLogic Server, for example.

  Note: Do not specify server name, user name or password. Also, do not specify the address and port of the Administration Server.

- Security Policy File

  The JVM's security policy file is used by default. There is also a WebLogic security policy file available, located at `weblogic/lib/weblogic.policy`.

3. Click Apply.

Platform Support for Node Manager

The Node Manager is available for use only on Windows and UNIX platforms. Native libraries are available for running the Node Manager on Windows, Solaris, HP-UX, AIX and Red Hat Linux operating systems. For UNIX operating systems other than Solaris and HP UX, you will need to use the following argument on the `java` command line when starting the Node Manager:

```
-Dweblogic.nodemanager.nativeVersionEnabled=false
```

Note: If you wish to start the Node Manager on a UNIX operating system other than Solaris or HP UX, you cannot have any white space characters in any of the parameters that will be passed to the `java` command line in starting the Node Manager. For example, if you try to use the parameter

```
-Dweblogic.Name=big iron
```

this will not work due to the space character in the name `big iron`.

Starting the Node Manager from the Command Line

There are two ways to start the Node Manager. You can start the Node Manager from the `java` command line, or you can use Node Manager start scripts. For information about using scripts, see Starting the Node Manager Using Start Scripts. Node Manager can also be set up as a Windows service. If Node Manager is a Windows service, it will be automatically restarted whenever Windows reboots. For information on setting up Node Manager as a Windows service, see Setting Up Node Manager as a Windows Service.

Setting Up the Environment

Before starting the Node Manager, there are a number of environment variables that need to be set. One way to set the environment variables would be to run the scripts provided with the WebLogic Server software. The script is called `setEnv.sh` on UNIX and `setEnv.cmd` on Windows. This script is located in the directory *`install_dir`*`/config/`*`domain_name`*, where *`install_dir`* is the directory where you installed WebLogic and *`domain_name`* is the name of the domain.

Note: If you use the Node Manager start scripts (`startNodeManager.cmd` on Windows, `startNodeManager.sh` on UNIX) to start the Node Manager, you do not need to set the environment variables as these are set by the Node Manager start script. For more information, see Starting the Node Manager Using Start Scripts.

Setting the Environment Variables on Windows

Make sure that the `JAVA_HOME` environment variable points to the root directory where you installed the JDK that you are using for the Node Manager. For example:

```
set JAVA_HOME=D:\bea\jdk131
```

Node Manager has the same JDK version requirements as the WebLogic Server.

You also need to set the `WL_HOME` environment variable. For example:

```
set WL_HOME=D:\bea\wlserver6.1
```

In addition, you need to set your `PATH` environment variable to access the Node Manager classes and the `java` executable. For example:

```
set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%
```

Setting the Environment Variables on UNIX

Assuming you have set the `WL_HOME` environment variable to point to the directory where you installed WebLogic, the following is an example of setting the `PATH` variable to point to the WebLogic and JDK software:

```
PATH=$WL_HOME/bin;$JAVA_HOME/jre/bin:$JAVA_HOME/bin:$PATH
```

In the above example it is assumed that the `JAVA_HOME` variable points to the root directory of the JDK installation.

You also need to set the path to the native UNIX libraries that be used by the Node Manager. The following is an example on Solaris:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$WL_HOME/lib/solaris:$WL_HOME/lib/solaris/oci8
16_8
```

The following is an example on HP UX:

```
SHLIB_PATH=$SHLIB_PATH:$WL_HOME/lib/hpux11:$WL_HOME/lib/hpux11/oci816_8
```

Setting the Classpath

The classpath can be set either as an option on the `java` command line or as an environment variable. The following is an example (on Windows NT) of setting the classpath as an environment variable:

```
set CLASSPATH=.;.\lib\weblogic_sp.jar;.\lib\weblogic.jar
```

Starting the Node Manager

If you don't use a start script to start the Node Manager, be sure to start the Node Manager in the root installation directory where you installed the WebLogic Server software. This is the directory that contains the `\config` subdirectory.

The command to start the Node Manager is:

```
java weblogic.nodemanager.NodeManager
```

Command-Line Arguments

The listen address of the Node Manager needs to be set when it is started. You can specify the address on which the Node Manager will listen for requests from the Administration Server with the following parameter:

`-Dweblogic.nodemanager.listenAddress=`*`host`*

where *`host`* is the DNS name or IP address of the machine where the Node Manager is started.

The default port on which the Node Manager will listen for requests from the Administration Server is 5555. You can change this with the following startup parameter:

`-Dweblogic.nodemanager.listenPort=`*`port`*

The Node Manager creates logs for each Managed Server that it is responsible for. By default, these are subdirectories under a directory `NodeManagerLogs`. You can change the location of this directory with the following startup parameter:

`-Dweblogic.nodemanager.savedLogsDirectory=`*`path`*

The Node Manager uses Secure Socket Layer for its communication with the Administration Server. For this reason, you must specify a digital certificate when starting the Node Manager. You can specify the location of the certificate with the following startup parameter:

`-Dweblogic.nodemanager.certificateFile=`*`path_to_cert`*

The default certificate type used with Node Manager is RSA. If you want to specify a different certificate type (such as DSA), use the following argument on the command line:

`–Dweblogic.nodemanager.certificateType=`*`type`*

where *`type`* is either `RSA` or `DSA`.

To pass the private key password, used to access the encrypted private key, use the following argument on the command line:

`–Dweblogic.nodemanager.certificatePassword=`*`pkpassword`*

where `pkpassword` is the private key password.

The certificate authority, or chain of authority, used to certify the user's identity is contained in a trusted certificate authorities file. By default this is `config/demo.crt`. You can specify another trusted certificate authorities file by using the following argument on the command line:

`–Dweblogic.nodemanager.trustedCerts=`*`path`*

where *`path`* is the location of the trusted certificate authorities file.

You also need to specify the location of the BEA home directory — the root directory under which all BEA products and licenses are installed. You can specify the BEA home directory with the following command-line argument:

`-Dbea.home=`*`directory`*

If you used DNS host names rather than IP addresses in the trusted hosts file, then you must also include the following startup parameter:

`-Dweblogic.nodemanager.reverseDnsEnabled=true`

By default, reverse DNS is disabled.

You can also specify the name of the file that contains the list of trusted hosts with the following startup parameter:

`-Dweblogic.nodemanager.trustedHosts=`*`path`*

where *`path`* specifies the location of the trusted hosts file. By default this file is located in the `/config` directory.

The default location of the WebLogic security policy file is `weblogic/lib/weblogic.policy`. To specify a different location for this file, use the following argument on the command line:

`-Djava.security.policy==`*`policy_file`*

where *`policy_file`* specifies the location of the WebLogic policy file.

By default Node Manager does not do SSL host name verification. If you want to turn on host name verification, use the following argument on the command line:

`–Dweblogic.nodemanager.sslHostNameVerificationEnabled=true`

Classpath Option

Node Manager also requires some of the same Java classes that are used by WebLogic Server. When starting the Node Manager, the following must be included as values in the `-classpath` option on the `java` command line:

- */weblogic/*`lib/weblogic_sp.jar`
- */weblogic/*`lib/weblogic.jar`

Starting the Node Manager Using Start Scripts

Sample start scripts are provided for use in starting Node Manager. These scripts are located in the `/config` directory where you have installed the WebLogic Server software. The start script for Windows is named `startNodeManager.cmd`. The start script for UNIX machines is named `startNodeManager.sh`.

Edit the start script for Node Manager to correctly specify the Node Manager listen address. Set the listen address by including the following argument in the startup command:

`-Dweblogic.nodemanager.listenAddress=`*`host`*

where *`host`* is the DNS name or IP address of the machine where the Node Manager will run.

Before invoking Node Manager the start script run `setEnv` script in the domain directory, or export the PATH and CLASSPATH variables in both the `startNodeManager` and `startWebLogic` scripts. Otherwise you may encounter class not found exceptions when using Node Manager.

Remote Starting and Killing of Managed Servers

If you have the Node Managers running on machines where you have Managed Servers configured, you can start the Managed Server as follows:

1. Invoke the Administration Console (if it isn't already running).
2. Right click on the name of the server in the navigation tree (left pane).
3. Select **Start this server...**.

When you start the Managed Server, the messages that are usually printed to `STDOUT` or `STDERROR` when starting a WebLogic Server are displayed in the right pane of the Administration Console. These messages are also written to the Node Manager log file for that server.

You can stop the Managed Server in the same way:

1. Right click on the name of the Managed Server in the left pane.
2. Select **Kill this server...**.

The **Kill this server...** option instructs the Node Manager on the machine where the target Managed Server is running to kill the target WebLogic Server process.

Note: The **Kill this server...** option cannot be used to stop the Administration Server.

The Distinction Between Stopping and Killing a Managed Server

If you right click on the name of a server in the left pane of the Administration Console, one of the options is **Stop this server...**. This option does not use the Node Manager to stop the selected server. If you select the **Stop this server...** option, the Administration Server sends an administrative shutdown request to the selected server. The Node Manager is not used in that case. Unlike the **Kill this server...** option, the **Stop this server...** option can be used to shut down the Administration Server.

Because the **Stop this server...** option uses the administrative capability of a Managed Server to initiate a shutdown, it can only be used if the server is alive and responding to administrative requests. The **Kill this server...** option would be typically used in situations where the target server is hung or not responding to administrative requests from the Administration Server.

The same pop-up menu gives you access to the `StdOut` and `StdErr` output generated by the Managed Server. Select the **Get StdOut for this server** option to view the StdOut output or select **Get StdErr for this server** to view the StdErr output.

Starting and Killing Domains and Clusters

You can also start or kill all of the Managed Servers in the active domain:

1. Right click on the name of the active domain in the left panel.
2. Select **Kill this domain...** or **Start this domain...**

If you start the entire domain from the Administration Console, the results displayed in the right pane will consist of a series of links to the results for each Managed Server that was configured for that domain.

You can also start or kill all of the Managed Servers in a selected cluster in a single action in a similar manner.

1. Right click on the name of the cluster in the left panel.
2. Select **Kill this cluster...** or **Start this cluster...**

Note: You cannot start or kill the Administration Server using the Node Manager.

Setting Up Node Manager as a Windows Service

The directory *`install_dir`*`/config/`*`mydomain`* (where *`install_dir`* is the root directory of the WebLogic Server installation and *`mydomain`* is the default configuration directory name specified during installation) contains scripts for installing and uninstalling the WebLogic Server as a Windows service. The script `installNtService.cmd` is used to install WebLogic Server as a Windows service; the script `uninstallNtService.cmd` is used to uninstall WebLogic Server as a Windows service. You can copy and modify these scripts to install or remove Node Manager as a Windows service.

In the following procedure it is assumed that `mydomain` is the default configuration directory specified during installation. If you specified a different name for the default configuration directory during installation, substitute it for `mydomain` throughout.

To install Node Manager as a Windows Service, do the following:

1. Make a copy of the script `installNtService.cmd` from the *`install_dir`*`/config/mydomain` directory (*`install_dir`* is the root of the WebLogic software installation) and rename it `installNMNtService.cmd`.
2. Make a copy of the script `uninstallNtService.cmd` from the *`install_dir`*`/config/mydomain` directory (*`install_dir`* is the root of the WebLogic software installation) and rename it `uninstallNMNtService.cmd`.
3. Modify the script `installNMNtService.cmd` to include the command line instruction you want to use to start the Node Manager and make sure that you modify the startup command to change the target startup class from `weblogic.Server` to `weblogic.nodemanager.NodeManager`. For information on command-line options, see Starting the Node Manager from the Command Line.
4. Rename the service to an appropriate name such as `nodemanager`.
5. Modify the script `uninstallNMNtService.cmd` so that the target service name is the one you use in the `installNMNtService.cmd` script for starting the Node Manager as a Windows service.
6. Make sure that the installNMNtService.cmd script is started in the root directory of the WebLogic software installation. For example: `c:\bea\wlserver6.1`.
7. Install Node Manager as a Windows service by invoking the script `installNMNtService.cmd`.
8. Start Node Manager as a Windows service from:
   Start→Settings→Control Panel→Administrative Tools→Services

Removing Node Manager as a Windows Service

To uninstall Node Manager as a Windows service, invoke the script `uninstallNMNtService.cmd`.

4 Configuring WebLogic Servers and Clusters

The following sections discuss how to set up WebLogic Servers and WebLogic Server clusters:

- Overview of Server and Cluster Configuration
- Role of the Administration Server
- Starting the Administration Console
- How Dynamic Configuration Works
- Planning a Cluster Configuration
- Server Configuration Tasks
- Cluster Configuration Tasks

Overview of Server and Cluster Configuration

The persistent configuration for a domain of WebLogic Servers and clusters is stored in an XML configuration file. You can modify this file in three ways:

- Through the Administration Console, BEA's graphical user interface (GUI) for managing and monitoring a domain configuration. This is intended as the main way to modify or monitor the domain configuration.
- By writing a program to modify the configuration attributes, based on the configuration application programmatic interface (API) provided with WebLogic Server.
- By running the WebLogic Server command-line utility for accessing configuration attributes of domain resources. This is provided for those who want to create scripts to automate domain management.

Role of the Administration Server

Whichever method you choose, the Administration Server must be running when you modify your domain configuration.

The Administration Server is the WebLogic Server on which the Administration Service runs. The Administration Service provides the functionality for WebLogic Server, and manages the configuration for an entire domain.

By default; an instance of WebLogic Server is treated as an Administration Server. When the Administration Server starts, it loads the configuration files, which are stored, by default, in a directory called `config` under the *`WEBLOGIC_HOME`* directory. The `config` directory has a subdirectory for each domain that is available to the Administration Server. The actual configuration file resides inside the domain-specific directory and is called `config.xml`. By default, when an Administration Server starts, it looks for the configuration file (`config.xml`) under the default domain directory (which is specified during installation of WebLogic Server software).

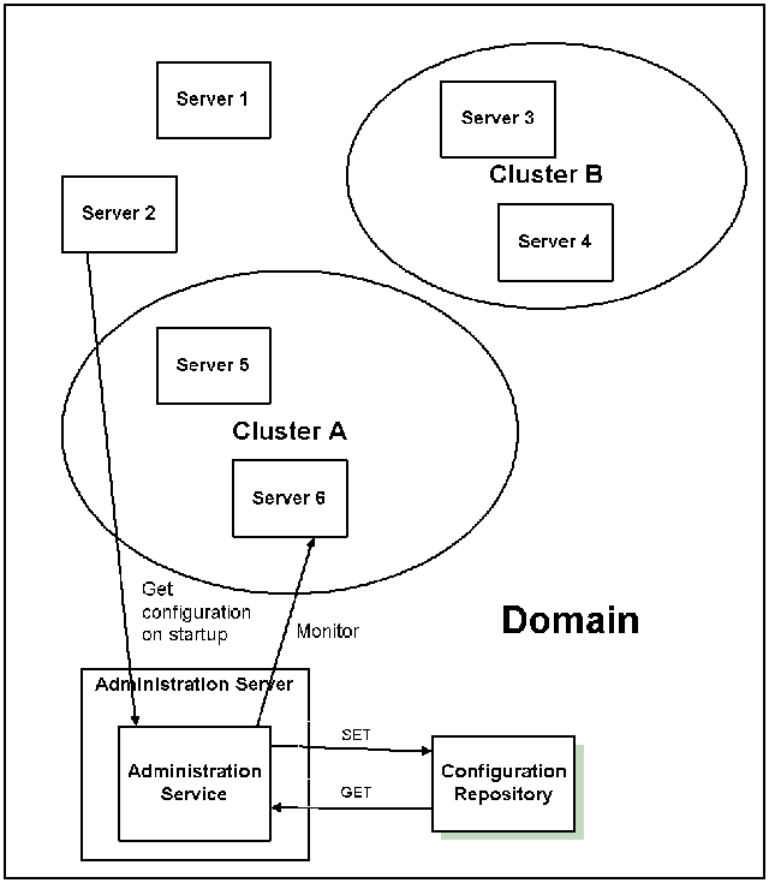
Each time the Administration Server is successfully started, a backup configuration file named `config.xml.booted` is created in the domain specific directory.In the unlikely event that the `config.xml` file should become corrupted during the lifetime of the server, it is possible to revert to this previously known, good configuration.

A domain could consist of only one WebLogic Server. But in that case, that WebLogic Server would be an Administration Server because each domain must have at least (and at most) one Administration Server.

Figure 4-1 shows a typical production environment that contains an Administration Server and multiple WebLogic Servers. When you start the servers in such a domain, the Administration Server is started first. As each additional server is started, it is instructed to contact the Administration Server for its configuration information. In this way, the Administration Server operates as the central control entity for the configuration of the entire domain. No more than one Administration Server can be active in a domain. Only the Administration Server can modify the configuration files when it is running.

Note: Do not use a shared filesystem and a single installation to run multiple WebLogic Server instances on separate machines. Using a shared filesystem introduces a single point of contention. All servers must compete to access the filesystem (and possibly to write individual log files). Moreover, should the shared filesystem fail, you may be unable to start managed or clustered servers.

Figure 4-1 WebLogic Server Configuration



Starting the Administration Console

The main point of access to the Administration Server is through the Administration Console. To open the Administration Console:

1. Enter the following URL:
   http://*host*:*port*/console

   where *host* is the host name or IP address of the machine on which the Administration Server is running and *port* is the address of the port at which the Administration Server is listening for requests (by default, 7001).

2. The system prompts you to enter a user ID and password. Enter your UserID and password. The system performs an authentication and authorization check: it verifies the user ID and password against the user database.

   If you are authorized to work with the console, then the console is displayed in the access mode that the system administrator originally assigned to you: either ReadOnly or Read/Write

How Dynamic Configuration Works

WebLogic Server allows you to change the configuration attributes of domain resources dynamically, that is, while servers are running. In most cases you do not need to restart WebLogic Server for your changes to take effect. When an attribute is reconfigured, the new value is immediately reflected in both the current run-time value of the attribute and the persistent value stored in the XML configuration file.

There are exceptions, however. If, for example, you change a WebLogic Server's listen port, the new address will not be used until the next time you start the affected server. In that case, if you modify the value, you are changing the persistent value stored in the XML file and the current run-time configuration value for the attribute may differ from that persistently stored value. The Administration Console indicates if the persistent and runtime values for a configuration attribute are not the same using an icon which changes to an alert

when the server needs to be restarted for changes to take effect.

The console does a validation check on each attribute that users change. The errors that are supported are out-of-range errors and datatype mismatch errors. In both cases, an error dialog box displays telling the user that an error has occurred.

Once the Administration Console has been started, if another process captures the Listen Port assigned to the Administration Server, you should remove the process that has captured the server. If you are not able to remove the process that has captured the Listen Port assigned to the Administration Server, you must edit the `Config.XML` file to change the assigned Listen Port. For information about editing the `Config.XML` file, please see the *Configuration Reference*.

Planning a Cluster Configuration

When planning a cluster configuration, keep in mind the following constraints on the networking environment and the cluster configuration.

1. The machine(s) you will be using as WebLogic hosts for the cluster must have permanently assigned, static IP addresses. You cannot use dynamically-assigned IP addresses in a clustering environment. If the servers are behind a firewall and the clients are in front of the firewall, each server must have a *public* static IP address that can be reached by the clients.
2. All WebLogic Servers in a cluster must be located on the same local area network (LAN) and must be reachable via IP multicast.
3. All servers in a cluster must be running the same version of WebLogic Server.

Configure the servers in your cluster to support the particular mix of services that you are offering.

- For EJBs that are using JDBC connections, all the servers that deploy a particular EJB must have the same deployment and persistence configuration. This means configuring the same JDBC connection pool on each server.
- Every machine that hosts servlets must maintain the same list of servlets with identical ACLs (access control lists).
- If your client application uses JDBC connection pools directly, you must create identical connection pools (with identical ACLs) on each WebLogic Server. This means that it must be possible to create any connection pool in use on all machines in the cluster. If, for example, you configure a pool of connections to a Microsoft SQL Server database on a Windows NT server running WebLogic, you cannot use this connection pool in a cluster that

contains any non-Windows machines (that is, any machines that cannot support a Microsoft SQL Server connection).

- Other configuration details may differ for various members in the cluster. You might, for example, configure a Solaris server to process more login requests than a small Windows NT workstation. Such differences are acceptable. Thus, in the example given here, the *performance-specific* attributes of individual cluster members may be configured with different values, so long as the service configuration for all members is identical. In practice, this often results in WebLogic Servers in the cluster being identically configured *in all areas to do with WebLogic services, class files, and external resources* (such as databases).

Server Configuration Tasks

Server configuration tasks that can be accomplished from the Administration Console include:

- Configuring an individual server using the Server node of the Administration Console. The attributes that can be changed using this node include the Server Name, the ListenPort, and the IP Address.
- Cloning an individual server using the Server node of the Administration Console. The individual server is cloned, maintaining the attribute values in the original server and the name of the new server is set on the Configuration portion of the Server node.
- Deleting a server using the Server node of the Administration Console. Click the delete icon for the server you want to delete. A dialog box will appear asking you to confirm the deletion of the server. Click Yes to confirm your decision to delete the server.
- Viewing a server log using the Server node of the Administration Console. Click the server you want to monitor. Select the Monitoring tab. Click the View Server Log link and monitor the server log in the right pane of the Administration Console.
- Viewing a server JNDI tree using the Server node of the Administration Console. Click the server you want to monitor. Select the Monitoring tab.

Click the View JNDI Tree link and view the tree in the right pane of the Administration Console.

- Viewing server execute queues using the Server node of the Administration Console. Click the server you want to monitor. Click the Execute Queues link and view the table in the right pane of the Administration Console.
- Viewing server execute threads using the Server node of the Administration Console. Click the server you want to monitor. Click the Execute Queues link and view the table in the right pane of the Administration Console.
- Viewing server sockets using the Server node of the Administration Console. Click the server you want to monitor. Click the View Sockets link and view the table in the right pane of the Administration Console.
- Viewing server connections using the Server node of the Administration Console. Click the server you want to monitor. Click the View Connections link and view the table in the right pane of the Administration Console.
- Forcing garbage collection on a server using the Server node of the Administration Console. Click the server you want to monitor. Select the JVM tab. Click the Force Garbage Collection link. A dialog box will appear to confirm that garbage collection has taken place.
- Monitoring server security using the Server node of the Administration Console. Click the server you want to monitor. Select the Monitoring tab. Select the Security tab. The security information will be displayed.
- Viewing the server version using the Server node of the Administration Console. Click the server you want to monitor. Select the Version tab. The version data for this server will be displayed.
- Monitoring server clusters using the Server node of the Administration Console. Click the server you want to monitor. Select the Cluster tab. The cluster data for this server will be displayed.
- Deploying EJBs on a server using the Server node of the Administration Console. Click the server on which you want to deploy EJBs. Click the EJB you want to deploy and use the move control to move it to the Chosen column. Click Apply to save your selections.
- Monitoring all EJB deployments on a server using the Server node of the Administration Console. Click the server on which you want to monitor EJBs.

Click the Monitor All EJB Deployments link to display the EJB Deployments table.

- Deploying Web Application components on a server using the Server node of the Administration Console. Click the server on which you want to deploy Web Applications. Click the Web Application you want to deploy and use the move control to move it to the Chosen column. Click Apply to save your selections.
- Monitoring all Web Application components on a server using the Server node of the Administration Console. Click the server on which you want to monitor Web Applications. Click the Monitor All Web Applications link to display the Web Application Deployments table.
- Deploy startup and shutdown classes on a server using the Server node of the Administration Console. Click the server on which you want to deploy startup classes. Click the startup class you want to deploy and use the move control to move it to the Chosen column. Click Apply to save your selections. Use the same process to deploy shutdown classes using the Shutdown Class control.
- Assigning JDBC connection pools to a server using the Server node of the Administration Console. Click a server for Web-server assignment. Click one or more JDBC connection pools in the Available column that you want to assign to the server and use the mover control to move the JDBC connection pools you selected to the Chosen column. Click Apply to save your assignments.
- Assigning WLEC connection pools to a server using the Server node of the Administration Console. Click a server for WLEC connection-pool assignment. Click one or more WLEC connection pools in the Available column that you want to assign to the server and use the mover control to move the WLEC connection pools you selected to the Chosen column.
- Monitoring all WLEC connection pools on a server using the Server node of the Administration Console. Click a server for WLEC connection-pool monitoring. Click the Monitor All WLEC Connection Pools on This Server text link on the WLEC tab. The WLEC Connection Pools table displays in the right pane showing all the connection pools assigned to this server.
- Assigning XML registries to a server using the Server node of the Administration Console. Click a sever for XML registry assignment. Click a registry from the XML Registry drop-down list box. Click Apply to save your selection.

- Assigning mail sessions to a server using the Server node of the Administration Console. Click a server for mail-session assignment. Click one or more mail sessions in the Available column that you want to assign to the server. Use the mover control to move the mail sessions you selected to the Chosen column. Click Apply to save your selections.
- Assigning File T3s to a server using the Server node of the Administration Console. Click a server for file T3 assignment. Click one or more file T3s in the Available column that you want to assign to the server. Use the mover control to move the file T3s you selected to the Chosen column. Click Apply to save your selections.

Cluster Configuration Tasks

Cluster configuration tasks that can be accomplished from the Administration Console include:

- Configuring a cluster of servers using the Cluster node of the Administration Console. The attributes that can be changed using this node include the Cluster Name, the Cluster ListenPort, and the names of the servers in the cluster.
- Cloning a cluster of servers using the Cluster node of the Administration Console. The cluster is cloned, maintaining the attribute values and individual servers in the original cluster and the name of the new cluster is set on the Configuration portion of the Server node.
- Monitoring servers in a cluster using the Cluster node of the Administration Console. Click a cluster for server monitoring. Click the Monitor Server Participation in This Cluster text link. The server table displays in the right pane showing all the servers assigned to this cluster.
- Assigning servers to a cluster using the Cluster node of the Administration Console. Click a cluster for server assignment. Click one or more servers in the Available column that you want to assign to the cluster. Use the mover control to move the servers you selected to the Chosen column. Click Apply to save your selections.
- Deleting a cluster using the Cluster node of the Administration Console. Click the Delete icon in the row of the cluster you want to delete. A dialog box

displays in the right pane asking you to confirm your deletion request. Click Yes to confirm your decision to delete the cluster.

5 Monitoring a WebLogic Server Domain

The following sections explain how to monitor your WebLogic Server domain:

- Overview of Monitoring
- Monitoring Servers
- Monitoring JDBC Connection Pools

Overview of Monitoring

The tool for monitoring the health and performance of your WebLogic Server domain is the Administration Console. The Administration Console allows you to view status and statistics for WebLogic Server resources such as servers, HTTP, the JTA subsystem, JNDI, security, CORBA connection pools, EJB, JDBC, and JMS.

Monitoring information is presented in the right pane of the Administration Console. You access a page by selecting a container or subsystem, or a particular entity under a container, on the hierarchical domain tree, in the left pane.

The Administration Console provides three types of page that contain monitoring information:

- Monitoring tab pages for a particular entity (such as an instance of a JDBC Connection Pool or a particular server's performance)

- Tables of data about all entities of a particular type (such as the WebLogic Servers table)
- Views of the domain log and of the local server logs. For information about log messages, see Using Log Messages to Manage WebLogic Servers.

The Administration Console obtains information about domain resources from the Administration Server. The Administration Server, in turn, is populated with Management Beans (MBeans), based on Sun's Java Management Extension (JMX) standard, which provides the scheme for management access to domain resources.

The Administration Server contains both configuration MBeans, which control the domain's configuration, and run-time MBeans. Run-time MBeans provide a snapshot of information about domain resources, such as JVM memory usage or the status of WebLogic Servers. When a particular resource in the domain (such as a Web application) is instantiated, an MBean instance is created which collects information about that particular resource.

When you access a monitoring page for particular resources in the Administration Console, the Administration Server performs a GET operation to retrieve the current attribute values.

The following sections describe some of the monitoring pages that are useful for managing a WebLogic Server domain. These pages have been selected simply to illustrate the facilities provided by the Administration Console.

Monitoring Servers

The servers table and the monitoring tab pages for individual servers enable you to monitor WebLogic Servers. The servers table provides a summary of the status of all servers in your domain. If only a small subset of the log messages from the server are forwarded to the domain log, accessing the local server log may be useful for troubleshooting or researching events.

For more information about the log files and the logging subsystem, see Using Log Messages to Manage WebLogic Servers.

You can access monitoring data for each WebLogic server from the monitoring tabs for that server. The Logging tab provides access to the local log for the server (that is, the log on the machine where the server is running).

The Monitoring→General tab page indicates the current status and provides access to the Active Queues table, the Active Sockets table, and the Connections table. The Active Execute Queues table provides performance information such as the oldest pending request and the queue throughput.

Performance

The Monitoring→Performance tab graphs real-time data on JVM memory heap usage and request throughput. This tab page also enables you to force the JVM to perform garbage collection on the memory heap.

The Java heap is a repository for Java objects (live and dead). Normally you do not need to perform garbage collection manually because the JVM does this automatically. When the JVM begins to run out of memory, it halts all execution and uses a garbage collection algorithm to free up space no longer used by Java applications.

On the other hand, developers debugging applications may have occasion to force garbage collection manually. Manual garbage collection may be useful, for example, if they are testing for memory leaks that rapidly consume JVM memory.

Server Security

The Monitoring→Security tab provides statistics about invalid login attempts and locked and unlocked users.

JMS

The Monitoring→JMS tab provides statistics on JMS servers and connections. This page also provides links to the tables of active JMS connections and active JMS servers, which monitor such attributes as total current sessions.

JTA

The Monitoring→JTA tab provides statistics on the Java Transactions subsystem such as total transactions and total rollbacks. The page provides links to tables that list transactions by resource and name, and a table of in-flight transactions.

Monitoring JDBC Connection Pools

Java Database Connectivity (JDBC) subsystem resources can also be monitored via the Administration Console. The Monitoring tab for a JDBC connection pool allows you to access a table listing statistics for the instances of that pool. As with other entity tables in the Administration Console, you can customize the table to select which attributes you want to be displayed.

A number of these attributes provide important information for managing client database access.

The Waiters High field indicates the highest number of clients waiting for a connection at one time. The Waiters field tells you how many clients are currently waiting for a connection. The Connections High field indicates the highest number of connections that have occurred at one time. The Wait Seconds High field tells you the longest duration a client has had to wait for a database connection. These attributes allow you to gauge the effectiveness of the current configuration is in responding to client requests.

If the Connections High field value is close to the value of the Maximum Capacity field (set on the Configuration Connections tab), you might consider increasing the value of Maximum Capacity (the maximum number of concurrent connections). If the value in the Waiters High field indicates that clients are subject to a long wait for database access, then you might want to increase the size of the pool.

The value in the Shrink Period field is the length of time the JDBC subsystem waits before shrinking the pool from the maximum. When the subsystem shrinks the pool, database connections are destroyed. Creating a database connection consumes resources and can be time-consuming. If your system has intermittent bursts of client requests, a short shrink period might mean that database connections are being recreated continually, which may degrade performance.

6 Using Log Messages to Manage WebLogic Servers

The following sections describe the functions of the logging subsystem:

- Overview of Logging Subsystem
- Local Server Log Files
- Message Attributes
- Message Catalog
- Message Severity
- Browsing Log Files
- Creating Domain Log Filters

Overview of Logging Subsystem

Log messages are a useful tool for managing systems. They allow you to detect problems, track down the source of a fault, and track system performance. Log messages generated by the WebLogic Server software are stored in two locations:

- WebLogic Server component subsystems generate messages that are logged to a local file, that is, a file that resides on the machine where the server is running. If there are multiple servers on a machine, each server has its own log file. Applications deployed on your WebLogic Servers may also log messages to the server's local log file.
- In addition, a subset of messages logged locally are stored in a central domain-wide log file maintained by the Administration Server.

Java Management Extension (JMX) facilities, embedded in the WebLogic Server, are used to transmit log messages from WebLogic Servers to the Administration Server. A message forwarded to other entities on the initiative of a local WebLogic Server is called a *notification* in JMX terminology.

When a WebLogic server starts, the Administration Server's message handler registers with that server to receive log messages. At the time of registration, a user-modifiable filter is provided that is used by the local server to select the messages to be forwarded to the Administration Server. These messages are collected in the domain log.

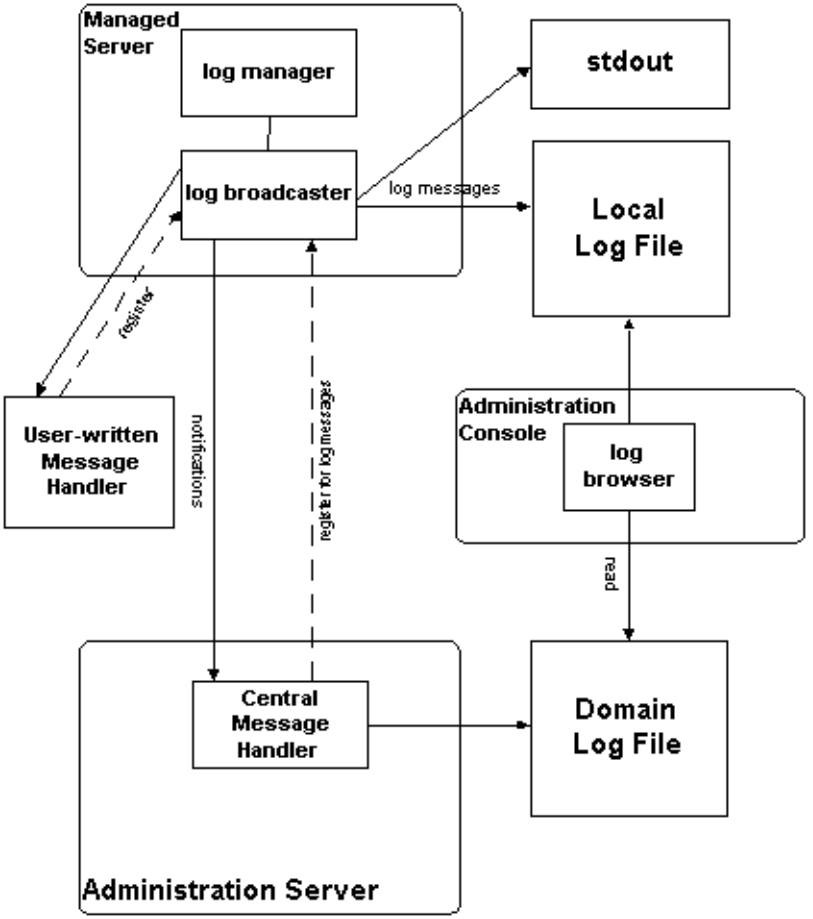
If the Administration Server is unavailable, Managed Servers continue to write messages to their local log files, but they do not keep track of which messages they generate while the Administration Server is unavailable. For example, if the Administration Server is unavailable for two hours and then is restored, the domain log will not contain any messages that were generated during the two hours.

By default, only the most important log messages (as determined by Message Severity) are forwarded from the local servers to the domain log. The domain log gives you an overall view of the entire domain while focusing on just the most critical messages.

If you want to modify the filter, to receive a different subset of logged messages from a local server, you can do so dynamically, using the Administration Console. You do not need to restart the local server for your changes to take effect. (See Creating Domain Log Filters.)

Developers can also build custom message handlers that can register with a WebLogic Server to receive log messages via JMX notifications.

Figure 6-1 WebLogic Server Logging Subsystem



Local Server Log Files

In versions of WebLogic Server prior to 6.0, a new log file is created once the log file reaches a maximum log file size. This type of automatic log file creation is called *log rotation*. In the current release, you have the option of basing log file rotation either on size or on time. To configure rotation, open the Administration Console and do the following:

1. In the left pane, select a server.
2. In the right pane, select Logging→Rotation.
3. In the Rotation Type field, select `By Size` or `By Date`.
   - `By Size`: Rotates the log when it exceeds the value entered into the File Min Size parameter.
   - `By Date`: Rotates the log after the number of minutes specified with the File Time Span parameter.
4. If you have selected `Size` as the Rotation Type, in the File Min Size field specify the file size (1 - 65535 kilobytes) that triggers the server to move log messages to a separate file.

   After the log file reaches the specified size, the next time the server checks the file size, it will rename the current log file and create a new one to store subsequent messages.
5. If you have selected Date as the Rotation Type, set the Rotation Time to the first date when you want the log file to rotate.

   Use the following format: *hh*:*mm*, where *hh* is the hour in a 24-hour format and *mm* is the minute.

   If the time that you specify has already past, then the server starts its file rotation immediately.
6. If you have selected Date as the Rotation Type, set the File Time Span to the number of hours after which the log file will rotate.
7. To include a time or date stamp in the file name when the log file is rotated, in the File Name field, add `java.text.SimpleDateFormat` variables to the file name. Surround each variable with percentage (`%`) characters.

For example, if you enter the following value in the File Name field:
`myserver_%yyyy%_%MM%_%dd%_%hh%_%mm%.log`

the server's log file will be named:
`myserver_yyyy_MM_dd_hh_mm.log`

When the server instance rotates the log file, the rotated file name contains the date stamp. For example, if the server instance rotates its local log file on 2 April, 2003 at 10:05 AM, the log file that contains the old log messages will be named:
`myserver_2003_04_02_10_05.log`

If you do not include a time and date stamp, the rotated log files are numbered in order of creation *`filename`*`nnnnn`, where *`filename`* is the name configured for the log file. For example: `myserver.log00007`.

By default, the local server log file is called *`servername`*`.log` (where *`servername`* is the name of the server) and is created in the directory where you started the WebLogic Server. You can set the file name also on the Configuration→Logging page for the server.

You can specify the maximum number of rotated files that can accumulate by setting an appropriate value for the `File Count` field. Once the number of log files reaches this number, the oldest log file is deleted each time a log file rotation occurs.

The local server log always has all the messages that have been logged.

Configuring logging by the local server also includes the ability to specify which messages are logged to `stdout`. You can exclude messages of lower severity by specifying the lowest severity to be logged. You can also enable or disable logging of debug messages to `stdout`.

Note: It is recommended that you not modify the server log files by manually editing them. Log rotation by time is based on the timestamp of the files. If you modify the file, this changes the timestamp and log rotation may become confused. If you edit the file manually this may lock the file and prevent updating of the file by the server.

Client Logging

Java clients that use the WebLogic logging facility may also generate log messages. However, messages logged by clients are not forwarded to the domain log. You configure logging properties of a client by entering the appropriate argument on the command line:

`-Dweblogic.log.`*`attribute`*`=`*`value`*

where *`attribute`* is any `LogMBean` attribute.

By default, logging to a log file is turned off for clients and messages are logged to `stdout`. You can turn on logging to a file and set the file name for the log if you include the following argument on the command line:

`-Dweblogic.log.FileName=`*`logfilename`*

where *`logfilename`* is the name that you want to use for the client log file.

The following command line arguments can also be used for client logging:

`-Dweblogic.StdoutEnabled=`*`boolean`*

`-Dweblogic.StdoutDebugEnabled=`*`boolean`*

`-Dweblogic.StdoutSeverityLevel = [64 | 32 | 16 | 8 | 4 | 2 | 1 ]`

where *`boolean`* is either `true` or `false` and the numeric values for `StdoutSeverityLevel` correspond to the following severity levels: `INFO(64)` `WARNING(32)`, `ERROR(16)`, `NOTICE(8)`, `CRITICAL(4)`, `ALERT(2)` and `EMERGENCY(1)`.

Log File Format

The first line of each message in a log file begins with #### followed by the message header. The message header provides the run-time context of the message. Each attribute of the message is contained between angle brackets.

Lines following the message body are only present for messages logging an exception and display the stack trace for the exception. If a message is not logged within the context of a transaction, the angle brackets (separators) for Transaction ID are present even though no Transaction ID is present.

The following is an example of a log message:

```
####<Jun 2, 2000 10:23:02 AM PDT> <Info> <SSL> <bigbox> <myServer>
    <SSLListenThread> <harry> <> <004500> <Using exportable strength SSL>
```

In this example, the message attributes are: Timestamp, Severity, Subsystem, Machine Name, Server Name, Thread ID, User ID, Transaction ID, Message ID, and Message Text.

Note: Log messages logged by clients do not have the attributes Server Name or Thread ID.

Note: The character encoding used in writing the log files is the default character encoding of the host system.

Message Attributes

Each log message saved in a server log file the attributes listed in the following table may be defined. The Message Id may also associate the message with additional attributes (such as Probable Cause and Recommended Action) contained in the Message Catalog.

Attribute	Description
Timestamp	The time and date when the message originated, in a format that is specific to the locale.
Severity	Indicates the degree of impact or seriousness of the event reported by the message. See Message Severity.
Subsystem	This attribute denotes the particular subsystem of WebLogic Server that was the source of the message. For example, EJB, RMI, JMS.
Server Name Machine Name Thread ID Transaction ID	These four attributes identify the origins of the message. Transaction ID is present only for messages logged within the context of a transaction. **Note:** Server Name and Thread ID are not present in log messages generated by a Java client and logged to a client log.

Attribute	Description
User ID	The user from the security context when the message was generated.
Message ID	A unique six-digit identifier. Message IDs through 499999 are reserved for WebLogic Server system messages.
Message Text	For WebLogic Server messages, this contains the Short Description as defined in the system message catalog. (See Message Catalog.) For other messages, this is text defined by the developer of the program.

Message Catalog

In addition to the information contained in a log message, messages generated by WebLogic Server system components (or possibly by user-written code) include additional pre-defined or canned information that is stored in a message catalog. The additional attributes stored in the message catalog are described below.

Attribute	Description
Message Body	This is a short textual description of the condition being reported. This is the same as Message Text in the message.
Message Detail	A more detailed description of the condition that the message is reporting.
Probable Cause	An explanation as to why the message was logged. The probable cause of the condition the message is reporting.
Recommended Action	A recommendation for action by the administrator to resolve or avoid the condition reported in the message.

You can access these additional message attributes from log views in the Administration Console.

Message Severity

WebLogic Server log messages have an attribute called **severity** which reflects the importance or potential impact on users of the event or condition reported in the message.

Defined severities are described below. Severities are listed in order of severity with `Emergency` being the highest severity.

Severity	Forwarded to Domain Log by Default?	Meaning
Informational	No	Used for reporting normal operations.
Warning	No	A suspicious operation or configuration has occurred but it may not have an impact on normal operation.
Error	Yes	A user error has occurred. The system or application is able to handle the error with no interruption, and limited degradation, of service.
Notice	Yes	A warning message: A suspicious operation or configuration has occurred which may not affect the normal operation of the server.
Critical	Yes	A system or service error has occurred. The system is able to recover but there might be a momentary loss, or permanent degradation, of service.
Alert	Yes	A particular service is in an unusable state while other parts of the system continue to function. Automatic recovery is not possible; the immediate attention of the administrator is needed to resolve the problem.
Emergency	Yes	The server is in an unusable state. This severity indicates a severe system failure or panic.

Debug Messages

Messages with a severity of debug are a special case. Debug messages are not forwarded to the domain log. Debug messages may contain detailed information about an application or the server. These messages should only occur when the application is running in debug mode.

Browsing Log Files

The log browsing capabilities of the Administration Console allow you to do the following:

- View the local log file of any server.
- View the domain-wide log file.

When viewing either the domain log or the local server log, you can:

- Select log messages to be viewed based on the time of occurrence, user ID, subsystem, message severity, or the message short description.
- View messages as they are logged, or search for past log messages.
- Select the log message attributes to be displayed in the Administration Console and the order in which the attributes are displayed.

Viewing the Logs

You can access both the domain log and the local server log files from the Administration Console. How to do these tasks is discussed in the Console Online Help:

- Viewing the Domain Log
- Viewing the Local Server Log

Creating Domain Log Filters

The log messages forwarded by WebLogic Servers to the domain log are, by default, a subset of messages logged locally. You can configure a log filter that selects log messages for forwarding based on message severity, subsystem, or user ID. (Debug messages are a special case and are not forwarded to the domain log.) You can create or modify domain log filters from the domain log filters table. The domain log filters table is accessible from the domain monitoring tab page. See the Administration Console online help for more information on creating domain log filers.

7 Deploying Applications

The following sections discuss installation and deployment of applications and application components on WebLogic Server:

- Supported Formats for Deployment
- Using the Administration Console to Deploy Applications
- Updating Deployed Applications at Startup
- Auto-Deployment

Supported Formats for Deployment

J2EE applications can be deployed on WebLogic Servers either as an Enterprise Application Archive (EAR) file or in exploded directory format.

However, if an application is deployed in exploded format, we recommend that no component other than the Web application component should be in exploded format. If the application is deployed in archived format, then we recommend that all of the components of the application also be in archived format.

An archived component is either packaged as an EJB archive (JAR) file, a Web Application Archive (WAR) file, or a Resource Adaptor Archive (RAR) file.

For more information about Web applications see Configuring WebLogic Server Web Components.

For more information about Resource Adaptor components, see Managing the WebLogic J2EE Connector Architecture.

Using the Administration Console to Deploy Applications

You can use the Administration Console to install and deploy an application or application components (such as EJB JAR files) and deploy instances of application components on target WebLogic Servers. There are several steps to carry out this task:

Step 1: Configure and Deploy the Application.

To do this, do the following:

1. Select Deployments→Applications to invoke the applications table.
2. Click on the link Configure a new Application to invoke the Create a new Application page.
3. Fill in the fields for this configuration entry for the application:
   - Give the application entry a name
   - Indicate the path to the application (EAR file)
   - Indicate whether this application is to be deployed.
4. Click Create to create the new entry.

Installing an application (or application component) via the Administration Console also creates entries for that application and application components in the configuration file for the domain (`/config/`*`domain_name`*`/config.xml`). The Administration Server also generates JMX Management Beans (MBeans) that enable configuration and monitoring of the application and application components.

Step 2: Deploying Application Components.

There are three types of component you can deploy: Web application components, EJBs or resource connector components.

Note: If you deploy application components (such as EJBs or WAR or RAR files) to Managed Servers in a cluster, you must ensure that the same application components are deployed on all servers in the cluster. To do this, you would select the cluster as the target for the deployment.

Deploying Web Application Components

To deploy a Web application component on a Managed Server, do the following:

1. Select Deployments→Web Applications to invoke the Web Application table.
2. Click on the link Configure a new Web Application which invokes the Create a new WebApp Component configuration page.
3. Fill in the fields appropriately:
   - A name for the configuration entry for this component
   - The Uniform Resource Identifier (URI) pointing to this component (or, if the component is in exploded format, its root directory)
   - The path to the WAR file, or a path to a directory if the Web application is in exploded format
   - Select a deployment order. This determines the order in which Web applications are deployed when the server starts up. (See Deployment Order.)
   - Indicate whether this component is to be deployed.
4. Click Create to create the new component entry.
5. You can choose to deploy the component on either target Managed Servers or target clusters. Click on Targets→Servers for deploying the component on target Managed Servers. Click on the Targets→Clusters for deploying the component on target clusters.

6. The Available field lists Managed Servers (or Clusters if you selected Targets→Clusters). Select the Managed Servers (or clusters) on which this Web Application is to be deployed by using the arrow buttons to move them to the Chosen field. Click Apply for your change to take effect.

For more information about configuring Web applications, see Configuring WebLogic Server Web Components.

Deploying EJB Components

To deploy EJBs on a Managed Server, do the following:

1. Select Deployments→EJB to invoke the EJB Deployments table.
2. Click on the link Configure a new EJB which invokes the Create a new EJB Component page.
3. Fill in the fields appropriately:
   - A name for the configuration entry for this component
   - The Uniform Resource Identifier (URI) pointing to this component (or, if the component is in exploded format, its root directory)
   - The path to the JAR file
   - Select a deployment order. This determines the order in which EJBs are deployed when the server starts up. (See Deployment Order.)
   - Indicate whether this component is to be deployed.
4. Click Create to create the new component entry.
5. You can choose to deploy the component on either target Managed Servers or target clusters. Click on Targets→Servers for deploying the component on target Managed Servers. Click on the Targets→Clusters for deploying the component on target clusters.
6. The Available field lists Managed Servers (or Clusters if you selected Targets→Clusters). Select the Managed Servers (or clusters) on which this Web Application is to be deployed by using the arrow buttons to move them to the Chosen field. Click Apply for your change to take effect.

Deploying Resource Adapter Components

To deploy a resource connector component on a Managed Server, do the following:

1. Select Deployments→Connectors to invoke the Resource Connectors table.
2. Click on the link Configure a new Connector Component which invokes the Create a new Connector Component configuration page.
3. Fill in the fields appropriately:
   - A name for the configuration entry for this component
   - The Uniform Resource Identifier (URI) pointing to this component (or, if the component is in exploded format, its root directory)
   - The path to the RAR file
   - Select a deployment order. This determines the order in which resource connectors are deployed when the server starts up. (See Deployment Order.)
   - Indicate whether this component is to be deployed.
4. Click Create to create the new component entry.
5. You can choose to deploy the component on either target Managed Servers or target clusters. Click on Targets→Servers for deploying the component on target Managed Servers. Click on the Targets→Clusters for deploying the component on target clusters.
6. The Available field lists Managed Servers (or Clusters if you selected Targets→Clusters). Select the Managed Servers (or clusters) on which this Web Application is to be deployed by using the arrow buttons to move them to the Chosen field. Click Apply for your change to take effect.

For more information about resource connectors, see Managing the WebLogic J2EE Connector Architecture.

When an application or application component (such as an EAR or WAR file, or EJB JAR files) is deployed to a particular WebLogic Server, the files are copied to a directory `.wlnotdelete` under `/config/`*`domain_name`*`/applications` on the target WebLogic Server. The WebLogic Administration Service invokes a file distribution servlet to copy the files to the target server.

Deployment Order

Within components of the same type, such as EJBs, you can specify the order in which they are to be deployed at server startup. The integer that you indicate in the Deployment Order field when deploying the component indicates the priority in relation to other components of the same type, such as the order of deployment among EJBs. Components that have deployment order 0 are deployed first among components of that type.

However, WebLogic Server uses an ordering among types that is not affected by this user-defined ordering among components of the same type. When WebLogic Server starts, the following class-level order is used in deployment:

1. JDBC Connection Pools
2. JDBC Multi Pools
3. JDCB Data Sources
4. JDBC Tx Data Sources
5. JMS Connection Factories
6. JMS Servers
7. Connector Components
8. EJB Components
9. Web App Components

Updating Deployed Applications at Startup

By default the Administration Server copies an application to a Managed Server at startup only if both of the following conditions are true:

- The application has been modified and redeployed.

- The Managed Server does not have the latest version of the application.

The Administration Server maintains a `StagedTargets` list that specifies which Managed Servers in a domain have the latest version of an application. At startup, a Managed Server queries its Administration Server to determine if it (the Managed Server) has the most recent version of the application. If an updated version of the application is available, it is copied to the Managed Server, and the Administration Server adds that Managed Server to the StagedTargets list.

When an Administration Server goes down, all Managed Servers are removed from the `StagedTargets` list. When the Administration Server is rebooted, each Managed Server in the domain copies its deployed applications from the Administration Server. This assures that, in the event that an application has been updated while the Administration Server was down, all Managed Servers have the same version of the application.

Forcing Application Update at Startup

A startup option for Administration Servers, `-Dweblogic.management.forceApplicationCopy`, forces Managed Servers to obtain the latest version of deployed applications at startup. If an Administration Server is started with `-Dweblogic.management.forceApplicationCopy` set to `true`, when a Managed Server in the domain starts up, the applications deployed to that Managed Server are copied from the Administration Server to the Managed Server. The startup option can be specified either on the command line or in a startup script.

Auto-Deployment

Auto-deployment is a method for quickly deploying an application on the Administration Server. It is recommended that this method be used only in a development environment for testing an application. Use of auto-deployment in a production environment or for deployment of components on Managed Servers is not recommended. You can ensure that auto-deployment is turned off if you start the Administration Server with the following argument on the `java` command line:

```
-Dweblogic.ProductionModeEnabled=true
```

If auto-deployment is enabled for the target WebLogic Server domain, when an application is copied into the `/config/`*`domain_name`*`/applications` directory of the WebLogic Administration Server, the Administration Server detects the presence of the new application and deploys it automatically (if the Administration Server is running). (The subdirectory *`domain_name`* is the name of the WebLogic Server domain that was used when starting the Administration Server.) If WebLogic Server is not running when you copy the application to the `/applications` directory, the application is deployed the next time the WebLogic Server is started.

Note that applications already in the `/config/`*`domain_name`*`/applications` directory will be deployed automatically even when production mode is enabled. It is not advisable to store applications or files in the directory `/config/`*`domain_name`*`/applications` when switching to production mode.

If you use the Administration Console to make any changes to the configuration of an application that is auto-deployed, those changes are not stored persistently, that is, no changes are made to the configuration defined in the `config.xml` for the active domain. If you make changes to the configuration of an application that is auto-deployed, those changes will be gone if you restart the Administration Server.

Enabling or Disabling Auto-Deployment

By default, auto-deployment is enabled.

To determine whether you have auto-deployment enabled, invoke the Administration Console and go to the domain applications settings page (*domain_name*→Configuration→Applications) for the domain. This page allows you to enable or disable auto-deployment and to set the interval (in milliseconds) at which the WebLogic Server checks for new applications in the `/applications` subdirectory. By default, the Administration Server checks every three seconds for changes in the `/applications` directory when auto-deployment is enabled.

Auto-Deployment of Applications in Expanded Directory Format

An application or application component can be auto-deployed either in expanded directory format or as packaged in an Enterprise Application Archive (EAR) file, a Web Application Archive (WAR) file, or a Java Archive (JAR) file.

To dynamically deploy an application in exploded format, do the following:

1. Make sure the directory name created for the exploded application is the same as the Context Path of the application.
2. Copy this subdirectory under `/config/`*`domain_name`*`/applications`, where *`domain_name`* is the name of the target domain where the application is to be deployed. This will automatically deploy the application if auto-deploy is enabled.

Undeployment or Redeployment of Auto-Deployed Applications

An application or application component that was auto-deployed can be dynamically redeployed while the server is running. This may be useful if you want to update a deployed application or application component without stopping and restarting the WebLogic Administration Server. To dynamically redeploy a JAR, WAR or EAR file, simply copy the new version of the file over the existing file in the `/applications` directory.

This feature is useful for developers who can simply add the copy to the `/applications` directory as the last step in their makefile, and the server will then be updated.

Redeployment of Applications Auto-Deployed in Exploded Format

You can also dynamically redeploy applications or application components that have been auto-deployed in exploded format. When an application has been deployed in exploded format, the Administration Server periodically looks for a file named `REDEPLOY` in the `WEB-INF` directory. If the timestamp on this file changes, the Administration Server redeploys the exploded directory.

If you want to update files in an exploded application directory, do the following:

1. When you first deploy the exploded application, create an empty file named `REDEPLOY` in the `WEB-INF` directory.
2. To update the exploded application, copy the updated files over the existing files in that directory.
3. After copying the new files, touch the `REDEPLOY` file in the exploded directory to alter its timestamp.

When the Administration Server detects the changed timestamp, it redeploys the contents of the exploded directory.

CHAPTER

8 Configuring WebLogic Server Web Components

The following sections discuss how to configure WebLogic Server Web components:

Overview

In addition to its ability to host dynamic Java-based distributed applications, WebLogic Server is also a fully functional Web server that can handle high volume Web sites, serving static files such as HTML files and image files as well as servlets and JavaServer Pages (JSP). WebLogic Server supports the HTTP 1.1 standard.

HTTP Parameters

You can configure the HTTP operating parameters using the Administration Console for each Server or Virtual Host.

Attribute	Description	Range of Values	Default Value
FrontendHost	When WebLogic Server redirects a request, it sets the host name returned in the HTTP response header with the string specified with Default Server Name. Useful when using firewalls or load balancers and you want the redirected request from the browser to reference the same host name that was sent in the original request.	String	Null

Attribute	Description	Range of Values	Default Value
FrontendHTTPPort	The frontend HTTP Port is set when the Port information coming from the URL may be inaccurate due to the presence of a firewall or proxy. If this parameter is set, the HOST header is ignored and this value is always used.	Valid Listen Port	null
FrontendHTTPSPort	The frontend HTTPS Port is set when the Port information coming from the URL may be inaccurate due to the presence of a firewall or proxy. If this parameter is set, the HOST header is ignored and this value is always used.	Valid Listen Port	null
Enable Keepalives	This attribute sets whether or not HTTP keep-alive is enabled	Boolean True = enabled False = not enabled	True
Send Server Header	If set to false, the server name is not sent with the HTTP response. Useful for wireless applications where there is limited space for headers.	Boolean True = enabled False = not enabled	True
Duration (labeled Keep Alive Secs on the Virtual Host panel)	The number of seconds that WebLogic Server waits before closing an inactive HTTP connection.	Integer	30

Attribute	Description	Range of Values	Default Value
HTTPS Duration (labeled Https Keep Alive Secs on the Virtual Host panel)	The number of seconds that WebLogic Server waits before closing an inactive HTTPS connection.	Integer	60
WAP Enabled	When selected, the session ID no longer includes JVM information. This may be necessary when using URL rewriting with WAP devices that limit the size of the URL to 128 characters. Selecting WAP Enabled may affect the use of replicated sessions in a cluster.	Enabled Disabled	Disabled
Post Timeout Secs	This attribute sets the timeout (in seconds) that WebLogic Server waits between receiving chunks of data in an HTTP POST data. Used to prevent denial-of-service attacks that attempt to overload the server with POST data.	Integer	30
Max Post Time	This attribute sets the time (in seconds) that WebLogic Server waits for chunks of data in an HTTP POST data.	Integer	-1
Max Post Size	This attribute sets the size of the maximum chunks of data in an HTTP POST data.	Integer	-1

Attribute	Description	Range of Values	Default Value
External DNS Address	If your system includes an address translation firewall that sits between the clustered WebLogic Servers and a plug-in to a web server front-end, such as the Netscape (proxy) plug-in, set this attribute to the address used by the plug-in to talk to this server.		

Configuring the Listen Port

You can specify the port that each WebLogic Server listens on for HTTP requests. Although you can specify any valid port number, if you specify port `80`, you can omit the port number from the HTTP request used to access resources over HTTP. For example, if you define port `80` as the listen port, you can use the form `http://hostname/myfile.html` instead of `http://hostname:portnumber/myfile.html`.

You define a separate listen port for regular and secure (using SSL) requests. You define the regular listen port on the Servers node in the Administration Console, under the Configuration/General tab, and you define the SSL listen port under the Configuration/SSL tab.

Web Applications

HTTP and Web services are deployed according to the Servlet 2.3 specification from Sun Microsystems, which describes the use of *Web Applications* as a standardized way of grouping together the components of a Web-based application. These components include JSP pages, HTTP servlets, and static resources such as HTML pages or image

files. In addition, a Web Application can access external resources such as EJBs and JSP tag libraries. Each server can host any number of Web Applications. You normally use the name of the Web Application as part of the URI you use to request resources from the Web Application.

For more information, see Assembling and Configuring Web Applications at `http://e-docs.bea.com/wls/docs61/webapp/index.html`.

Web Applications and Clustering

Web Applications can be deployed in a cluster of WebLogic Servers. When a user requests a resource from a Web Application, the request is routed to one of the servers of the cluster that host the Web Application. If an application uses a session object, then sessions must be replicated across the nodes of the cluster. Several methods of replicating sessions are provided.

For more information, see Using WebLogic Server Clusters at http://e-docs.bea.com/wls/docs61/cluster/index.html.

Designating a Default Web Application

Every server and virtual host in your domain can declare a *default Web Application*. The default Web Application responds to any HTTP request that cannot be resolved to another deployed Web Application. In contrast to all other Web Applications, the default Web Application does *not* use the Web Application name as part of the URI. Any Web Application targeted to a server or virtual host can be declared as the default Web Application. (Targeting a Web Application is discussed later in this section. For more information about virtual hosts, see "Configuring Virtual Hosting" on page 8-8).

The default domain and the examples domain that are shipped with Weblogic Server each have a default Web Application already configured. The default Web Application in these domains is named `DefaultWebApp` and is located in the `applications` directory of each domain.

If you declare a default Web Application that fails to deploy correctly, an error is logged and users attempting to access the failed default Web Application receive an HTTP `400` error message.

For example, if your Web Application is called `shopping`, you would use the following URL to access a JSP called `cart.jsp` from the Web Application:

```
http://host:port/shopping/cart.jsp
```

If, however, you declared `shopping` as the default Web Application, you would access `cart.jsp` with the following URL:

```
http://host:port/cart.jsp
```

(Where `host` is the host name of the machine running WebLogic Server and `port` is the port number where the WebLogic Server is listening for requests.)

To designate a default Web Application for a server or virtual host, use the Administration Console:

1. In the left pane, expand the Web Application node
2. Select your Web Application
3. In the right pane, select the Targets tab.
4. Select the Servers tab and move the server (or virtual host) to the chosen column. (You can also target all the servers in a cluster by selecting the Clusters tab and moving the cluster to the Chosen column.)
5. Click Apply.
6. Expand the Servers (or virtual host) node in the left pane.
7. Select the appropriate server or virtual host.
8. In the right pane, select the General tab
9. Select the HTTP tab (If you are configuring a virtual host, select the General tab instead.)
10. Select a Web Application from the drop-down list labeled Default Web Application.
11. Click Apply.
12. If you are declaring a default Web Application for one or more managed servers, repeat these procedures for each managed server.

Configuring Virtual Hosting

Virtual hosting allows you to define host names that servers or clusters respond to. When you use virtual hosting you use DNS to specify one or more host names that map to the IP address of a WebLogic Server or cluster and you specify which Web Applications are served by the virtual host. When used in a cluster, load balancing allows the most efficient use of your hardware, even if one of the DNS host names processes more requests than the others.

For example, you can specify that a Web Application called `books` responds to requests for the virtual host name `www.books.com`, and that these requests are targeted to WebLogic Servers A,B and C, while a Web Application called `cars` responds to the virtual host name `www.autos.com` and these requests are targeted to WebLogic Servers D and E. You can configure a variety of combinations of virtual host, WebLogic Servers, clusters and Web Applications, depending on your application and Web server requirements.

For each virtual host that you define you can also separately define HTTP parameters and HTTP access logs. The HTTP parameters and access logs set for a *virtual host* override those set for a *server*. You may specify any number of virtual hosts.

You activate virtual hosting by targeting the virtual host to a server or cluster of servers. Virtual hosting targeted to a cluster will be applied to all servers in the cluster.

Virtual Hosting and the Default Web Application

You can also designate a *default Web Application* for each virtual host. The default Web Application for a virtual host responds to all requests that cannot be resolved to other Web Applications deployed on same server or cluster as the virtual host.

Unlike other Web Applications, a default Web Application does not use the Web Application name (also called the *context path*) as part of the URI used to access resources in the default Web Application.

For example, if you defined virtual host name `www.mystore.com` and targeted it to a server on which you deployed a Web Application called `shopping`, you would access a JSP called `cart.jsp` from the `shopping` Web Application with the following URI:

```
http://www.mystore.com/shopping/cart.jsp
```

If, however, you declared `shopping` as the default Web Application for the virtual host `www.mystore.com`, you would access `cart.jsp` with the following URI:

```
http://www.mystore.com/cart.jsp
```

For more information, see "How WebLogic Server Resolves HTTP Requests" on page 8-11.

Setting Up a Virtual Host

To define a virtual host, use the Administration Console to:

1. Create a new Virtual Host.
   a. Expand the Services node in the left pane. The node expands and displays a list of services.
   b. Click on the virtual hosts node. If any virtual hosts are defined, the node expands and displays a list of virtual hosts.
   c. Click on Create a New Virtual Host in the right pane.
   d. Enter a name to represent this virtual host.
   e. Enter the virtual host names, one per line. Only requests matching one of these virtual host names will be handled by the WebLogic Server or cluster targeted by this virtual host.
   f. (Optional) Assign a default Web Application to this virtual host.
   g. Click Create.
2. Define logging and HTTP parameters:
   a. (Optional) Click on the Logging tab and fill in HTTP access log attributes (For more information, see "Setting Up HTTP Access Logs" on page 8-14.)
   b. Select the HTTP tab and fill in the HTTP Parameters.
3. Define the servers that will respond to this virtual host.
   a. Select the Targets tab.

b. Select the Servers tab. You will see a list of available servers.

c. Select a server in the available column and use the right arrow button to move the server to the chosen column.

4. Define the clusters that will respond to this virtual host (optional). You must have previously defined a WebLogic Cluster. For more information, see Using WebLogic Server Clusters at `http://e-docs.bea.com/wls/docs61/cluster/index.html`.

   a. Select the Targets tab.

   b. Select the Clusters tab. You will see a list of available servers.

   c. Select a cluster in the available column and use the right arrow button to move the cluster to the chosen column. The virtual host is targeted on all of the servers of the cluster.

5. Target Web Applications to the virtual host.

   a. Click the Web Applications node in the left panel.

   b. Select the Web Application you want to target.

   c. Select the Targets tab in the right panel.

   d. Select the Virtual Hosts tab.

   e. Select Virtual Host in the available column and use the right arrow button to move the Virtual Host to the chosen column.

You must add a line naming the virtual host to the etc/hosts file on your server to ensure that the virtual host name can be resolved.

How WebLogic Server Resolves HTTP Requests

When WebLogic Server receives an HTTP request, it resolves the request by parsing the various parts of the URL and using that information to determine which Web Application and/or server should handle the request. The examples below demonstrate various combinations of requests for Web Applications, virtual hosts, servlets, JSPs, and static files and the resulting response.

Note: If you package your Web Application as part of an Enterprise Application, you can provide an alternate name for a Web Application that is used to resolve requests to the Web Application. For more information, see Deploying Web Applications as Part of an Enterprise Application at `http://e-docs.bea.com/wls/docs61/webapp/deployment.html#war-ear`.

The table below provides some sample URLs and the file that is served by WebLogic Server. The Index Directories Checked column refers to the Index Directories attribute that controls whether or not a directory listing is served if no file is specifically requested. You set Index Directories using the Administration Console, on the Web Applications node, under the Configuration →Files tab.

Table 8-1 Examples of How WebLogic Server Resolves URLs

URL	Index Directories Checked?	This file is served in response
`http://host:port/apples`	No	Welcome file* defined in the `apples` Web Application.
`http://host:port/apples`	Yes	Directory listing of the top level directory of the `apples` Web Application.

Table 8-1 Examples of How WebLogic Server Resolves URLs

URL	Index Directories Checked?	This file is served in response
`http://host:port/oranges/naval`	Does not matter	Servlet mapped with `<url-pattern>` of `/naval` in the `oranges` Web Application. There are additional considerations for servlet mappings. For more information, see Configuring Servlets at `http://e-docs.bea.com/wls/docs61/webapp/components.html#configuring-servlets`.
`http://host:port/naval`	Does not matter	Servlet mapped with `<url-pattern>` of `/naval` in the `oranges` Web Application and `oranges` is defined as the default Web Application. For more information, see Configuring Servlets at `http://e-docs.bea.com/wls/docs61/webapp/components.html#configuring-servlets`.
`http://host:port/apples/pie.jsp`	Does not matter	`pie.jsp`, from the top-level directory of the `apples` Web Application.
`http://host:port`	Yes	Directory listing of the top level directory of the *default* Web Application

Table 8-1 Examples of How WebLogic Server Resolves URLs

URL	Index Directories Checked?	This file is served in response
`http://host:port`	No	Welcome file* from the *default* Web Application.
`http://host:port/apples/myfile.html`	Does not matter	`myfile.html`, from the top level directory of the `apples` Web Application.
`http://host:port/myfile.html`	Does not matter	`myfile.html`, from the top level directory of the *default* Web Application.
`http://host:port/apples/images/red.gif`	Does not matter	`red.gif`, from the images subdirectory of the top-level directory of the `apples` Web Application.
`http://host:port/myFile.html` Where `myfile.html` does not exist in the `apples` Web Application and a *default servlet* has not been defined.	Does not matter	Error 404 For more information, see Customizing HTTP Error Responses at `http://e-docs.bea.com/wls/docs61/webapp/components.html#error-page`.
`http://www.fruit.com/`	No	Welcome file* from the default Web Application for a virtual host with a host name of `www.fruit.com`.
`http://www.fruit.com/`	Yes	Directory listing of the top level directory of the `default` Web Application for a virtual host with a host name of `www.fruit.com`.

Table 8-1 Examples of How WebLogic Server Resolves URLs

URL	Index Directories Checked?	This file is served in response
`http://www.fruit.com/oranges/myfile.html`	Does not matter	`myfile.html`, from the `oranges` Web Application that is targeted to a virtual host with host name `www.fruit.com`.

* For more information, see Configuring Welcome Pages at `http://e-docs.bea.com/wls/docs61/webapp/components.html#welcome_pages`.

Setting Up HTTP Access Logs

WebLogic Server can keep a log of all HTTP transactions in a text file, in either *common log format* or *extended log format*. Common log format is the default, and follows a standard convention. Extended log format allows you to customize the information that is recorded. You can set the attributes that define the behavior of HTTP access logs for each server or for each virtual host that you define.

Log Rotation

You can also choose to rotate the log file based on either the size of the file or after a specified amount of time has passed. When either one of these two criteria are met, the current access log file is closed and a new access log file is started. If you do not configure log rotation, the HTTP access log file grows indefinitely. The name of the access log file includes a numeric portion that is incremented upon each rotation. Separate HTTP Access logs are kept for each Web Server you have defined.

Setting Up HTTP Access Logs by Using the Administration Console

To set up HTTP access logs use the Administration Console at `http://e-docs.bea.com/wls/docs61/ConsoleHelp/virtualhost.html`.

1. If you have set up virtual hosting:
   a. Select the services node in the left pane.
   b. Select the virtual hosts node. The node expands and displays a list of virtual hosts.
   c. Select a virtual host.

   If you have *not* set up virtual hosting:
   a. Select the servers node in the left pane. The node expands and displays a list of servers.
   b. Select a server.
   c. Select the Logging tab.
   d. Select the HTTP tab.
2. Check the Enable Logging box.
3. Enter values for your Log File Name.
4. Select Common or Extended from the drop-down list labeled Format.
5. If you want the time stamps for HTTP log messages to be in Greenwich Mean Time (GMT) regardless of the local time zone that the host computer specifies, select Log Time in GMT.

   Use this method to comply with the W3C specification for Extended Format Log Files. The specification states that all time stamps for Extended Format log entries be in GMT.
6. In Log File BufferK Bytes, set the size (in kilobytes) of the HTTP message buffer. After the HTTP message buffer reaches this size, the server writes all current entries to the HTTP log file and flushes the buffer.

7. In Log Rotation Type, select `By Size` or `By Date`.
   - `By Size`: Rotates the log when it exceeds the value entered into the Log Buffer Size parameter.
   - `By Date`: Rotates the log after the number of minutes specified with the Rotation Period parameter.
8. If you have selected `Size` as the Rotation Type, in the Max Log File Size K Bytes field specify the file size (1 - 65535 kilobytes) that triggers the server to move log messages to a separate file.

   After the log file reaches the specified size, the next time the server checks the file size, it will rename the current log file as *`FileName.n`* and create a new one to store subsequent messages.
9. If you have selected Date as the Rotation Type, set the Rotation time to the first date when you want the log file to rotate.

   Use the following `java.text.SimpleDateFormat` format to specify a date and time: `MM-dd-yyyy-k:mm:ss`. For information about this format, refer to the J2EE Javadoc.

   If the time that you specify has already past, then the server starts its file rotation immediately.
10. If you have selected Date as the Rotation Type, set the Rotation Period to the number of minutes after which the log file will rotate.

Common Log Format

The default format for logged HTTP information is the common log format at `http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format`. This standard format follows the pattern:

```
host RFC931 auth_user [day/month/year:hour:minute:second
    UTC_offset] "request" status bytes
```

where:

`host`
: Either the DNS name or the IP number of the remote client

`RFC931`
: Any information returned by IDENTD for the remote client; WebLogic Server does not support user identification

auth_user
: If the remote client user sent a userid for authentication, the user name; otherwise "-"

day/month/year:hour:minute:second UTC_offset
: Day, calendar month, year and time of day (24-hour format) with the hours difference between local time and GMT, enclosed in square brackets

"request"
: First line of the HTTP request submitted by the remote client enclosed in double quotes

status
: HTTP status code returned by the server, if available; otherwise "-"

bytes
: Number of bytes listed as the content-length in the HTTP header, not including the HTTP header, if known; otherwise "-"

Setting Up HTTP Access Logs by Using Extended Log Format

WebLogic Server also supports extended log file format, version 1.0, as defined by the W3C. This is an emerging standard, and WebLogic Server follows the draft specification from W3C at `www.w3.org/TR/WD-logfile.html`. The current definitive reference may be found from the W3C Technical Reports and Publications page at `www.w3.org/pub/WWW/TR`.

The extended log format allows you to specify the type and order of information recorded about each HTTP communication. To enable the extended log format, set the Format attribute on the HTTP tab in the Administration Console to `Extended`. (See step 4. in "Setting Up HTTP Access Logs by Using the Administration Console" on page 8-15).

You specify what information should be recorded in the log file with directives, included in the actual log file itself. A directive begins on a new line and starts with a # sign. If the log file does not exist, a new log file is created with default directives. However, if the log file already exists when the server starts, it must contain legal directives at the head of the file.

Creating the Fields Directive

The first line of your log file must contain a directive stating the version number of the log file format. You must also include a `Fields` directive near the beginning of the file:

```
#Version: 1.0
#Fields: xxxx xxxx xxxx ...
```

Where each `xxxx` describes the data fields to be recorded. Field types are specified as either simple identifiers, or may take a prefix-identifier format, as defined in the W3C specification. Here is an example:

```
#Fields: date time cs-method cs-uri
```

This identifier instructs the server to record the date and time of the transaction, the request method that the client used, and the URI of the request for each HTTP access. Each field is separated by white space, and each record is written to a new line, appended to the log file.

Note: The `#Fields` directive must be followed by a new line in the log file, so that the first log message is not appended to the same line.

Supported Field identifiers

The following identifiers are supported, and do not require a prefix.

`date`
: Date at which transaction completed, field has type <date>, as defined in the W3C specification.

`time`
: Time at which transaction completed, field has type <time>, as defined in the W3C specification.

`time-taken`
: Time taken for transaction to complete in seconds, field has type <fixed>, as defined in the W3C specification.

`bytes`
: Number of bytes transferred, field has type <integer>.

Note that the `cached` field defined in the W3C specification is not supported in WebLogic Server.

The following identifiers require prefixes, and cannot be used alone. The supported prefix combinations are explained individually.

IP address related fields:

These fields give the IP address and port of either the requesting client, or the responding server. This field has type <address>, as defined in the W3C specification. The supported prefixes are:

`c-ip`
: The IP address of the client.

`s-ip`
: The IP address of the server.

DNS related fields

These fields give the domain names of the client or the server. This field has type <name>, as defined in the W3C specification. The supported prefixes are:

`c-dns`
: The domain name of the requesting client.

`s-dns`
: The domain name of the requested server.

`sc-status`
: Status code of the response, for example (404) indicating a “File not found” status. This field has type <integer>, as defined in the W3C specification.

`sc-comment`
: The comment returned with status code, for instance “File not found”. This field has type <text>.

`cs-method`
: The request method, for example GET or POST. This field has type <name>, as defined in the W3C specification.

`cs-uri`
: The full requested URI. This field has type <uri>, as defined in the W3C specification.

`cs-uri-stem`
: Only the stem portion of URI (omitting query). This field has type <uri>, as defined in the W3C specification.

`cs-uri-query`
: Only the query portion of the URI. This field has type <uri>, as defined in the W3C specification.

Creating Custom Field Identifiers

You can also create user-defined fields for inclusion in an HTTP access log file that uses the extended log format. To create a custom field you identify the field in the ELF log file using the `Fields` directive and then you create a matching Java class that generates the desired output. You can create a separate Java class for each field, or the Java class can output multiple fields. A sample of the Java source for such a class is included in this document. See "Java Class for Creating a Custom ELF Field" on page 8-24.

To create a custom field:

1. Include the field name in the `Fields` directive, using the form:

   `x-`*`myCustomField`*.

   Where *`myCustomField`* is a fully-qualified class name.

   For more information on the `Fields` directive, see "Creating the Fields Directive" on page 8-18.

2. Create a Java class with the same fully-qualified class name as the custom field you defined with the `Fields` directive (for example `myCustomField`). This class defines the information you want logged in your custom field. The Java class must implement the following interface:

   `weblogic.servlet.logging.CustomELFLogger`

   In your Java class, you must implement the `logField()` method, which takes a `HttpAccountingInfo` object and `FormatStringBuffer` object as its arguments:

   - Use the `HttpAccountingInfo` object to access HTTP request and response data that you can output in your custom field. Getter methods are provided to access this information. For a complete listing of these get methods, see "Get Methods of the HttpAccountingInfo Object" on page 8-21.
   - Use the `FormatStringBuffer` class to create the contents of your custom field. Methods are provided to create suitable output. For more information on these methods, see the Javadocs for FormatStringBuffer (see

`http://e-docs.bea.com/wls/docs61/javadocs/weblogic/servlet/logging/FormatStringBuffer.html`).

3. Compile the Java class and add the class to the `CLASSPATH` statement used to start WebLogic Server. You will probably need to modify the `CLASSPATH` statements in the scripts that you use to start WebLogic Server.

   Note: Do not place this class inside of a Web Application or Enterprise Application in exploded or jar format.

4. Configure WebLogic Server to use the extended log format. For more information, see "Setting Up HTTP Access Logs by Using Extended Log Format" on page 8-17.

Note: When writing the Java class that defines your custom field, you should not execute any code that is likely to slow down the system (For instance, accessing a DBMS or executing significant I/O or networking calls.) Remember, an HTTP access log file entry is created for *every* HTTP request.

Note: If you want to output more than one field, delimit the fields with a tab character. For more information on delimiting fields and other ELF formatting issues, see Extended Log Format at `http://www.w3.org/TR/WD-logfile-960221.html`.

Get Methods of the HttpAccountingInfo Object

The following methods return various data regarding the HTTP request. These methods are similar to various methods of `javax.servlet.ServletRequest`, `javax.servlet.http.Http.ServletRequest`, and `javax.servlet.http.HttpServletResponse`.

For details on these methods see the corresponding methods in the Java interfaces listed in the following table, or refer to the specific information contained in the table.

Table 8-2 Getter Methods of HttpAccountingInfo

`HttpAccountingInfo Methods`	**Where to find information on the methods**
`Object getAttribute(String name);`	javax.servlet.ServletRequest
`Enumeration getAttributeNames();`	javax.servlet.ServletRequest
`String getCharacterEncoding();`	javax.servlet.ServletRequest

Table 8-2 Getter Methods of HttpAccountingInfo

HttpAccountingInfo Methods	Where to find information on the methods
`int getResponseContentLength();`	`javax.servlet.ServletResponse.setContentLength()` This method *gets* the content length of the response, as set with the `setContentLength()` method.
`String getContentType();`	javax.servlet.ServletRequest
`Locale getLocale();`	javax.servlet.ServletRequest
`Enumeration getLocales();`	javax.servlet.ServletRequest
`String getParameter(String name);`	javax.servlet.ServletRequest
`Enumeration getParameterNames();`	javax.servlet.ServletRequest
`String[] getParameterValues(String name);`	javax.servlet.ServletRequest
`String getProtocol();`	javax.servlet.ServletRequest
`String getRemoteAddr();`	javax.servlet.ServletRequest
`String getRemoteHost();`	javax.servlet.ServletRequest
`String getScheme();`	javax.servlet.ServletRequest
`String getServerName();`	javax.servlet.ServletRequest
`int getServerPort();`	javax.servlet.ServletRequest
`boolean isSecure();`	javax.servlet.ServletRequest
`String getAuthType();`	`javax.servlet.http.Http.ServletRequest`
`String getContextPath();`	`javax.servlet.http.Http.ServletRequest`
`Cookie[] getCookies();`	`javax.servlet.http.Http.ServletRequest`
`long getDateHeader(String name);`	`javax.servlet.http.Http.ServletRequest`
`String getHeader(String name);`	`javax.servlet.http.Http.ServletRequest`
`Enumeration getHeaderNames();`	`javax.servlet.http.Http.ServletRequest`

Table 8-2 Getter Methods of HttpAccountingInfo

HttpAccountingInfo Methods	Where to find information on the methods
`Enumeration getHeaders(String name);`	`javax.servlet.http.Http.ServletRequest`
`int getIntHeader(String name);`	`javax.servlet.http.Http.ServletRequest`
`String getMethod();`	`javax.servlet.http.Http.ServletRequest`
`String getPathInfo();`	`javax.servlet.http.Http.ServletRequest`
`String getPathTranslated();`	`javax.servlet.http.Http.ServletRequest`
`String getQueryString();`	`javax.servlet.http.Http.ServletRequest`
`String getRemoteUser();`	`javax.servlet.http.Http.ServletRequest`
`String getRequestURI();`	`javax.servlet.http.Http.ServletRequest`
`String getRequestedSessionId();`	`javax.servlet.http.Http.ServletRequest`
`String getServletPath();`	`javax.servlet.http.Http.ServletRequest`
`Principal getUserPrincipal();`	`javax.servlet.http.Http.ServletRequest`
`boolean isRequestedSessionIdFromCookie();`	`javax.servlet.http.Http.ServletRequest`
`boolean isRequestedSessionIdFromURL();`	`javax.servlet.http.Http.ServletRequest`
`boolean isRequestedSessionIdFromUrl();`	`javax.servlet.http.Http.ServletRequest`
`boolean isRequestedSessionIdValid();`	`javax.servlet.http.Http.ServletRequest`
`String getFirstLine();`	Returns the first line of the HTTP request, for example: `GET /index.html HTTP/1.0`
`long getInvokeTime();`	Returns the length of time it took for the service method of a servlet to write data back to the client.
`int getResponseStatusCode();`	`javax.servlet.http.HttpServletResponse`
`String getResponseHeader(String name);`	`javax.servlet.http.HttpServletResponse`

Listing 8-1 Java Class for Creating a Custom ELF Field

```
import weblogic.servlet.logging.CustomELFLogger;
import weblogic.servlet.logging.FormatStringBuffer;
import weblogic.servlet.logging.HttpAccountingInfo;

/* This example outputs the User-Agent field into a
 custom field called MyCustomField
*/

public class MyCustomField implements CustomELFLogger{

public void logField(HttpAccountingInfo metrics,
  FormatStringBuffer buff) {
  buff.appendValueOrDash(metrics.getHeader("User-Agent"));
  }
}
```

Preventing POST Denial-of-Service Attacks

A Denial-of-Service attack is a malicious attempt to overload a server with phony requests. One common type of attack is to send huge amounts of data in an HTTP `POST` method. You can set three attributes in WebLogic Server that help prevent this type of attack. These attributes are set in the console, under *Servers* or *virtual hosts*. If you define these attributes for a virtual host, the values set for the virtual host override those set under *Servers*.

`PostTimeoutSecs`
: You can limit the amount of time that WebLogic Server waits between receiving chunks of data in an HTTP POST.

`MaxPostTimeSecs`
: Limits the total amount of time that WebLogic Server spends receiving post data. If this limit is triggered, a `PostTimeoutException` is thrown and the following message is sent to the server log:

  ```
  Post time exceeded MaxPostTimeSecs.
  ```

`MaxPostSize`

Limits the number of bytes of data received in a POST from a single request. If this limit is triggered, a `MaxPostSizeExceeded` exception is thrown and the following message is sent to the server log:

```
POST size exceeded the parameter MaxPostSize.
```

An HTTP error code 413 (Request Entity Too Large) is sent back to the client.

If the client is in listening mode, it gets these messages. If the client is not in listening mode, the connection is broken.

Setting Up WebLogic Server for HTTP Tunneling

HTTP tunneling provides a way to simulate a stateful socket connection between WebLogic Server and a Java client when your only option is to use the HTTP protocol. It is generally used to *tunnel* through an HTTP port in a security firewall. HTTP is a stateless protocol, but WebLogic Server provides tunneling functionality to make the connection appear to be a regular T3Connection. However, you can expect some performance loss in comparison to a normal socket connection.

Configuring the HTTP Tunneling Connection

Under the HTTP protocol, a client may only make a request, and then accept a reply from a server. The server may not voluntarily communicate with the client, and the protocol is stateless, meaning that a continuous two-way connection is not possible.

WebLogic HTTP tunneling simulates a T3Connection via the HTTP protocol, overcoming these limitations. There are two attributes that you can configure in the Administration Console to tune a tunneled connection for performance. You access these attributes in the Servers section, under the Tuning Tab located under the Configuration tab. It is advised that you leave them at their default settings unless you experience connection problems. These properties are used by the server to determine whether the client connection is still valid, or whether the client is still alive.

`Enable Tunneling`
: Enables or disables HTTP tunneling. HTTP tunneling is disabled by default.

`Tunneling Ping`
: When an HTTP tunnel connection is set up, the client automatically sends a request to the server, so that the server may volunteer a response to the client. The client may also include instructions in a request, but this behavior happens regardless of whether the client application needs to communicate with the server. If the server does not respond (as part of the application code) to the client request within the number of seconds set in this attribute, it does so anyway. The client accepts the response and automatically sends another request immediately.

  Default is 45 seconds; valid range is 20 to 900 seconds.

`Tunneling Timeout`
: If the number of seconds set in this attribute have elapsed since the client last sent a request to the server (in response to a reply), then the server regards the client as dead, and terminates the HTTP tunnel connection. The server checks the elapsed time at the interval specified by this attribute, when it would otherwise respond to the client's request.

  Default is 40 seconds; valid range is 10 to 900 seconds.

Connecting to WebLogic Server from the Client

When your client requests a connection with WebLogic Server, all you need to do in order to use HTTP tunneling is specify the HTTP protocol in the URL. For example:

```
Hashtable env = new Hashtable();
env.put(Context.PROVIDER_URL, "http://wlhost:80");
Context ctx = new InitialContext(env);
```

On the client side, a special tag is appended to the `http` protocol, so that WebLogic Server knows this is a tunneling connection, instead of a regular HTTP request. Your application code does not need to do any extra work to make this happen.

The client must specify the port in the URL, even if the port is 80. You can set up your WebLogic Server to listen for HTTP requests on any port, although the most common choice is port 80 since requests to port 80 are customarily allowed through a firewall.

You specify the listen port for WebLogic Server in the Administration Console under the "Servers" node, under the "Network" tab.

Using Native I/O for Serving Static Files (Windows Only)

When running WebLogic Server on Windows NT/2000 you can specify that WebLogic Server use the native operating system call `TransmitFile` instead of using Java methods to serve static files such as `HTML` files, text files, and image files. Using native I/O can provide performance improvements when serving larger static files.

To use native I/O, add two parameters to the `web.xml` deployment descriptor of a Web Application containing the files to be served using native I/O. The first parameter, `weblogic.http.nativeIOEnabled` should be set to `TRUE` to enable native I/O file serving. The second parameter, `weblogic.http.minimumNativeFileSize` sets the minimum file size for using native I/O. If the file being served is larger than this value, native I/O is used. If you do not specify this parameter, a value of `400` bytes is used.

Generally, native I/O provides greater performance gains when serving larger files; however, as the load on the machine running WebLogic Server increases, these gains diminish. You may need to experiment to find the correct value for `weblogic.http.minimumNativeFileSize`.

The following example shows the complete entries that should be added to the `web.xml` deployment descriptor. These entries must be placed in the `web.xml` file after the `<distributable>` element and before `<servlet>` element.

```
<context-param>
  <param-name>weblogic.http.nativeIOEnabled</param-name>
  <param-value>TRUE</param-value>
</context-param>

<context-param>
  <param-name>weblogic.http.minimumNativeFileSize</param-name>
  <param-value>500</param-value>
</context-param>
```

For more information on writing deployment descriptors, see Writing Web Application Deployment Descriptors at `http://e-docs.bea.com/wls/docs61/webapp/webappdeployment.html`.

9 Proxying Requests to Another HTTP Server

The following sections discuss how to proxy HTTP requests to another HTTP server:

Overview

When you use WebLogic Server as your primary Web server, you may also want to configure WebLogic Server to pass on, or proxy, certain requests to a secondary HTTP server, such as Netscape Enterprise Server, Apache, Microsoft Internet Information Server, or another instance of WebLogic Server. Any request that gets proxied is redirected to a specific URL.You can even proxy to another Web server on a different machine.You proxy requests based on the URL of the incoming request.

The `HttpProxyServlet` (provided as part of the distribution) takes an HTTP request, redirects it to the proxy URL, and sends the response to the client's browser back through the WebLogic Server instance that is redirecting requests. To use the proxy, you must configure it in a Web Application and deploy that Web Application on the WebLogic Server that is redirecting requests.

If you want to proxy requests to a cluster of WebLogic Servers, you can use the `HttpClusterServlet`. For more information, see "Proxying Requests to a WebLogic Cluster" on page 10-1.

New Version of the HttpProxyServlet

Service Pack 2 for WebLogic Server 6.1, contains a new version of the `HttpProxyServlet`. The older version of `HttpProxyServlet` is still available and functions as described in this document. Differences between the older version and the new version are noted where appropriate in this document. The older version is deprecated and will be removed from a future release.

The new version has the following features:

- Supports HTTP 1.1, including chunk-transfer and keep-alive.
- Uses connection pooling to improve performance.
- Uses a new set of parameters to define functionality. These parameters are the same as the parameters used for the Apache, Netscape, and Microsoft IIS plug-ins packaged with WebLogic Server.
- Has a new Java package name. The full class name is now: `weblogic.servlet.proxy.HttpProxyServlet`.

Setting Up a Proxy to a Secondary HTTP Server

To set up a proxy to a secondary HTTP server:

1. Register the `proxy` servlet in your Web Application deployment descriptor. The Web Application must be the default Web Application of the Server that is responding to requests.

If you are using the new version of `HttpProxyServlet`, see "Sample web.xml for use with NEW version of HttpProxyServlet" on page 9-4).

If you are using the older, deprecated version of `HttpProxyServlet`, see "Sample web.xml for use with DEPRECATED version of HttpProxyServlet" on page 9-5).

The class name for the new version of `HttpProxyServlet` is `weblogic.servlet.proxy.HttpProxyServlet`.

The class name for the older, deprecated version of `HttpProxyServlet` is `weblogic.t3.srvr.HttpProxyServlet`.)

For more information on deployment descriptors and Web Applications, see Assembling and Configuring Web Applications at `http://e-docs.bea.com/wls/docs61/webapp/index.html`.

2. Define the appropriate initialization parameters for the `HttpProxyServlet`. You define initialization parameters with the `<init-param> element` in the `web.xml` Web Application deployment descriptor.

   If you are using the **new** version of `HttpProxyServlet`, define the `WebLogicHost and WebLogicPort` parameters (these two parameters are required) using `<init-param> elements` in the `web.xml` Web Application deployment descriptor. Set `WebLogicHost` to the host name of the secondary HTTP server and set `WebLogicPort` to the port number on the secondary HTTP server that is listening for HTTP requests. You can also, where appropriate, define additional parameters as described in "Parameters for Web Server Plug-ins" on page D-1. For a sample deployment descriptor, see "Sample web.xml for use with NEW version of HttpProxyServlet" on page 9-4.

   If you are using the older, **deprecated** version of `HttpProxyServlet`, define the `redirectURL` parameter using the `<init-param> element` in the `web.xml` Web Application deployment descriptor. Set the value of this parameter to the URL of the secondary HTTP, including the port number. For example, `http://myHttpServer:7001`. For a sample deployment descriptor, see "Sample web.xml for use with DEPRECATED version of HttpProxyServlet" on page 9-5.

3. Map the `ProxyServlet` to a `<url-pattern>`. Specifically, map the file extensions you wish to proxy, for example `*.jsp`, or `*.html`. Use the `<servlet-mapping>` element in the `web.xml` Web Application deployment descriptor.

If you set the `<url-pattern>` to "`/`", then any request that cannot be resolved by WebLogic Server is proxied to the remote server. However, you must also specifically map the following extensions: `*.jsp`, `*.html`, and `*.html` if you want to proxy files ending with those extensions.

4. Deploy the Web Application on the WebLogic Server that redirects incoming requests.

Sample Deployment Descriptor for the Proxy Servlet

The following are samples of Web Application deployment descriptors for use with the `HttpProxyServlet`.

Listing 9-1 Sample web.xml for use with NEW version of HttpProxyServlet

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.
 //DTD Web Application 2.2//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

<servlet>
  <servlet-name>ProxyServlet</servlet-name>
  <servlet-class>weblogic.servlet.proxy.HttpProxyServlet</servlet
-class>

<init-param>
    <param-name>WebLogicHost</param-name>
    <param-value>serverName</param-value>
</init-param>

<init-param>
    <param-name>WebLogicPort</param-name>
    <param-value>serverPort</param-value>
</init-param> </servlet>

<servlet-mapping>
  <servlet-name>ProxyServlet</servlet-name>
```

```
  <url-pattern>/</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProxyServlet</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProxyServlet</servlet-name>
  <url-pattern>*.htm</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProxyServlet</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>

</web-app>
```

Note: You can define additional parameters by creating additional `<init-param>` blocks within the `<servlet>` block. For example,

```
<init-param>
    <param-name>ParameterName</param-name>
    <param-value>ParameterValue</param-value>
</init-param>
```

Where *`ParameterName`* is a parameter described in "Parameters for Web Server Plug-ins" on page D-1 and *`ParameterValue`* is the value you set for the parameter.

Listing 9-2 Sample web.xml for use with DEPRECATED version of HttpProxyServlet

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.
 //DTD Web Application 2.2//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

<servlet>
 <servlet-name>ProxyServlet</servlet-name>
 <servlet-class>weblogic.t3.srvr.HttpProxyServlet</servlet-class>
```

```
 <init-param>
    <param-name>redirectURL</param-name>
    <param-value>http://myServer:7001</param-value>
 </init-param>

</servlet>

<servlet-mapping>
  <servlet-name>ProxyServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProxyServlet</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProxyServlet</servlet-name>
  <url-pattern>*.htm</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>ProxyServlet</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>

</web-app>
```

CHAPTER

10 Proxying Requests to a WebLogic Cluster

The following sections discuss how to proxy HTTP requests to a cluster of WebLogic Servers:

Overview

The `HttpClusterServlet` provided with WebLogic Server proxies requests from a WebLogic Server acting as an HTTP server to server instances in a WebLogic Server cluster. The `HttpClusterServlet` provides load balancing and failover for the proxied HTTP requests. For additional information on servlets and clustering, see Understanding HTTP Session State Replication at `http://e-docs.bea.com/wls/docs61/cluster/servlet.html`.

To proxy requests to a single server instance rather than to a cluster, use the `HttpProxyServlet`. For more information, see "Proxying Requests to Another HTTP Server" on page 9-1.

New Version of the HttpClusterServlet

A new version of the `HttpClusterServlet` was introduced in WebLogic Server 6.1 SP02. The older version is still available, but it is deprecated and will be removed from a future release. Differences between the older version and the new version are described in this document.

The WebLogic Server 6.1 SP02 `HttpClusterServlet` has the following features:

- Supports HTTP 1.1, including chunk-transfer and keep-alive.
- Uses connection pooling to improve performance.
- Uses a new set of parameters to define functionality. These parameters are the same as the parameters used for the Apache, Netscape, and Microsoft IIS plug-ins packaged with WebLogic Server.
- Uses a new Java package name. The full class name is: `weblogic.servlet.proxy.HttpClusterServlet`.

Setting Up the HttpClusterServlet

To use the HTTP Cluster Servlet, configure it as the default web application on your proxy server machine, as described in the steps below. For an introduction to web applications, see "Overview of Web Applications" in *Assembling and Configuring Web Applications*.

1. If you have not already done so, configure a separate, non-clustered Managed Server to host the HTTP Cluster Servlet.
2. If you have not already done so, create a directory for the web application, and a WEB-INF subdirectory within the web application directory. For more information, see "Directory Structure" in *Assembling and Configuring Web Applications*.

3. Create the `web.xml` deployment descriptor file for the servlet, under the WEB-INF directory of the Web application. Use any text editor. Sample deployment descriptors for the new and deprecated versions of the proxy servlet are provided in "Sample Deployment Descriptors" on page 10-4. For comprehensive instructions on writing a `web.xml` file, see "Writing Web Application Deployment Descriptors" in *Assembling and Configuring Web Applications.*

   a. Define the name and class for the servlet in the `<servlet>` element in `web.xml`. The servlet name is `HttpClusterServlet` for both versions of the servlet.

      The new servlet class is `weblogic.servlet.proxy.HttpClusterServlet`. The deprecated servlet class is `weblogic.servlet.internal.HttpClusterServlet`.

   b. Identify the clustered server instances to which the proxy servlet will direct requests in the `<servlet>` element in `web.xml`.

      If you are using the new version of `HttpClusterServlet`, define the `WebLogicCluster` parameter.

      If you are using the deprecated version of `HttpClusterServlet`, define the `serverlist` parameter.

   c. Create `<servlet-mapping>` stanzas to specify the the requests that the servlet will proxy to the cluster, using the `<url-pattern>` element to identify specific file extensions, for example `*.jsp`, or `*.html`. Defining each pattern in a separate `<servlet-mapping>` stanza.

      You can set the `<url-pattern>` to "`/`" to proxy any request that cannot be resolved by WebLogic Server to the remote server instance. If you do so, you must also specifically map the following extensions: `*.jsp`, `*.html`, and `*.html`, to proxy files ending with those extensions. For an example, see "web.xml for HttpClusterServlet SP02" on page 10-5.

   d. Define, as appropriate, any additional parameters as described in "Parameters for Web Server Plug-ins" in *Using WebLogic Server with Plug-ins*. Follow the syntax instructions in "Syntax" on page 10-7. See also "Cluster Configuration and Proxy Plug-ins" on page 10-13.

   e. Define, as appropriate, cluster configuration parameters that affect the behavior of `HttpClusterServlet`. For more information, see "Cluster Configuration and Proxy Plug-ins" on page 10-13.

4. In the Administration Console, assign the servlet as the default Web Application for the Managed Server on your proxy server machine. For instructions, see "Designating a Default Web Application" on page 8-6.
5. In the Administration Console, deploy the servlet to the Managed Server on your proxy server machine. For instructions, see "Deploying Web Application Components" on page 7-3.

Sample Deployment Descriptors

The subsections that follow contain sample deployments descriptor files (`web.xml`) for use with the new and deprecated versions of `HttpClusterServlet`.

Each sample `web.xml` defines a set of parameters that specify the location and behavior of the HTTP Cluster Servlet.

In both versions of the servlet:

- The `DOCTYPE` stanza specifies the DTD used by WebLogic Server to validate `web.xml`.
- The `servlet` stanza:
  - Specifies the location of the proxy plug-in servlet class. The file is located in the `weblogic.jar` in your `WL_HOME/server/lib` directory. You do not have to specify the servlet's full directory path in `web.xml` because `weblogic.jar` is put in your CLASSPATH when you start WebLogic Server.
  - Identifies the host name and list port of each Managed Servers in the cluster, using the `WebLogicCluster` parameter in the SP02 and later version, and the `defaultServers` parameter for deprecated version.
- The three `servlet-mapping` stanzas specify that the servlet will proxy URLs that end in '/', 'htm', 'html', or 'jsp' to the cluster.

web.xml for HttpClusterServlet SP02

This listing before is a sample `web.xml` for the `HttpClusterServlet` provided with WebLogic Server 6.1 SP02 and later. For parameter definitions see “Proxy Servlet Deployment Parameters” on page 10-7.

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.
 //DTD Web Application 2.2//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

<servlet>
  <servlet-name>HttpClusterServlet</servlet-name>
    <servlet-class>
      weblogic.servlet.proxy.HttpClusterServlet
    </servlet-class>

  <init-param>
    <param-name>WebLogicCluster</param-name>
    <param-value>
       myserver1:7736|myserver2:7736|myserver:7736
    </param-value>
  </init-param>

</servlet>

<servlet-mapping>
  <servlet-name>HttpClusterServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>HttpClusterServlet</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>HttpClusterServlet</servlet-name>
  <url-pattern>*.htm</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>HttpClusterServlet</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>

</web-app>
```

web.xml for Deprecated HttpClusterServlet

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.
 //DTD Web Application 2.2//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

<servlet>
  <servlet-name>HttpClusterServlet</servlet-name>
    <servlet-class>
      weblogic.servlet.internal.HttpClusterServlet
    </servlet-class>

  <init-param>
    <param-name>defaultServers</param-name>
    <param-value>
       myserver1:7736:7737|myserver2:7736:7737|myserver:7736:7737
    </param-value>
  </init-param>

</servlet>

<servlet-mapping>
  <servlet-name>HttpClusterServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>HttpClusterServlet</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>HttpClusterServlet</servlet-name>
  <url-pattern>*.htm</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>HttpClusterServlet</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>

</web-app>
```

Proxy Servlet Deployment Parameters

Key parameters for configuring the behavior of the current version of the proxy servlet are listed in Table 10-1. Parameters for the deprecated proxy servlet are listed in Table 10-2.

Prior to WebLogic Server 6.1 SP02, the proxy servlet behavior was configured with its own parameter set. In SP02 and later, the parameters for the proxy servlet are renamed to match the those used to configure WebLogic Server plug-ins (for Apache, Microsoft, and Netscape web servers. For a complete list of parameters for configuring the proxy servlet and the plug-ins for third-party web servers see "Parameters for Web Server Plug-ins" on page D-1.

The "Deprecated Equivalent" column of Table 10-1 lists the (deprecated) parameter that used for the previous version of the proxy servlet.

Syntax

The syntax for specifying the parameters, and the file where they are specified, is different for the proxy servlet and for each of the plug-ins.

For the proxy servlet, specify the parameters in `web.xml`, each in its own `<init-param>` stanza within the `<servlet>` stanza of `web.xml`. For example:

```
<init-param>
    <param-name>ParameterName</param-name>
    <param-value>ParameterValue</param-value>
</init-param>
```

Table 10-1 WLS 6.1 SP02 Proxy Servlet Deployment Parameters

Parameter in WLS 6.1 SP02 and Later	Usage	Deprecated Equivalent

`WebLogicCluster`	`<init-param>`
	`<param-name>WebLogicCluster</param-name>`
	`<param-value>WLS1.com:`*`port`*`|WLS2.com:`*`port`*
	`</param-value>`
	Where `WLS1.com` and `WLS2.com` are the host names of servers in the cluster, and *`port`* is a port where the host is listening for HTTP requests.
	If you are using SSL between the plug-in and WebLogic Server, set the port number to the SSL listen port (see Configuring the Listen Port) and set the `SecureProxy` parameter to ON.
		`defaultServers`

`SecureProxy`	`<init-param>`
	`<param-name>SecureProxy</param-name>`
	`<param-value>`*`ParameterValue`*`</param-value>`
	`</init-param>`
	Valid values are ON and OFF.
	If you are using SSL between the plug-in and WebLogic Server, set the port number to the SSL listen port and set the `SecureProxy` parameter to ON.
		`secureProxy`

`DebugConfigInfo`	`<init-param>`
	`<param-name>Debug</param-name>`
	`<param-value>`*`ParameterValue`*`</param-value>`
	`</init-param>`
	Valid values are ON and OFF.
	If set to ON, you can query the `HttpClusterServlet` for debugging information by adding a request parameter of `?__WebLogicBridgeConfig` to any request. (Note: There are two underscore (_) characters after the ?.) For security reasons, it is recommended that you set the `DebugConfigInfo` parameter to OFF in a production environment.
		`DebugConfig Info`

Parameter in WLS 6.1 SP02 and Later	Usage	Deprecated Equivalent

`ConnectRetrySecs`	Interval in seconds that the the servlet will sleep between attempts to connect to a server instance. Assign a value less than `ConnectTimeoutSecs`.
	The number of connection attempts the servlet makes before returning an `HTTP 503/Service Unavailable` response to the client is `ConnectTimeoutSecs` divided by `ConnectRetrySecs`.
	Syntax:
	`<init-param>`
	`  <param-name>ConnectRetrySecs</param-name>`
	`  <param-value>`*`ParameterValue`*`</param-value>`
	`</init-param>`
		`numOfRetries`

`ConnectTimeoutSecs`	Maximum time in seconds that the servlet will attempt to connect to a server instance. Assign a value greater than `ConnectRetrySecs`.
	If `ConnectTimeoutSecs` expires before a successful connection, an `HTTP 503/Service Unavailable` response is sent to the client.
	Syntax:
	`<init-param>`
	`<param-name>ConnectTimeoutSecs</param-name>`
	`  <param-value>`*`ParameterValue`*`</param-value>`
	`</init-param>`
		`connection Timeout`

Parameter in WLS 6.1 SP02 and Later	Usage	Deprecated Equivalent

`PathTrim`	String trimmed by the plug-in from the beginning of the original URL, before the request is forwarded to the cluster.
	Syntax:
	`<init-param>`
	`<param-name>PathTrim</param-name>`
	`  <param-value>`*`ParameterValue`*`</param-value>`
	`</init-param>`
	Example:
	If the URL
	`http://myWeb.server.com/weblogic/foo`
	is passed to the plug-in for parsing and if `PathTrim` has been set to
	`/weblogic`
	the URL forwarded to WebLogic Server is:
	`http://myWeb.server.com:7001/foo`
		`pathTrim`

`TrimExt`	The file extension to be trimmed from the end of the URL.
	Syntax:
	`<init-param>`
	`<param-name>TrimExt</param-name>`
	`  <param-value>`*`ParameterValue`*`</param-value>`
	`</init-param>`
		`trimExt`

Parameter in WLS 6.1 SP02 and Later	Usage	Deprecated Equivalent
`clientCertProxy`	Specifies to trust client certificates in the `WL-Proxy-Client-Cert` header. Valid values are true and false. The default value is false. This setting is useful if user authentication is performed on the proxy server—setting `clientCertProxy` to true causes the proxy server to pass on the certs to the cluster in a special header, `WL-Proxy-Client-Cert`. The `WL-Proxy-Client-Cert` header can be used by any client with direct access to WebLogic Server. WebLogic Server takes the certificate information from that header, trusting that is came from a secure source (the plug-in) and uses that information to authenticate the user. If you set `clientCertProxy` to true, use a connection filter to ensure that WebLogic Server accepts connections only from the machine on which the plug-in is running. See "Filtering Network Connections" in *Programming WebLogic Security*.	

`PathPrepend`	String that the servlet prepends to the original URL, after `PathTrim` is trimmed, before forwarding the URL to the cluster.
	`<init-param>`
	`<param-name>PathPrepend</param-name>`
	`  <param-value>`*`ParameterValue`*`</param-value>`
	`</init-param>`
		`pathPrepend`

Table 10-2 Parameters for Deprecated Proxy Servlet

Parameter Name	Parameter Value	Default

`defaultServers`	List of host names and associated plain and SSL listen ports for the Managed Servers in the cluster, separated by the | character. For example: *`host1`*`:port:SSLport|` *`host2`*`:port:SSLport` If you set the `secureProxy` parameter to `ON`, the HTTPS port uses SSL between the proxy server and the clustered servers. You must define the SSL ports, even if `secureProxy` is `OFF`.	None
`secureProxy`	`ON/OFF`. If set to `ON`, enables SSL between the `HttpClusterServlet` and the member of a WebLogic Server cluster.	`OFF`
`DebugConfigInfo`	`ON/OFF`. If set to `ON`, you can query the `HttpClusterServlet` for debugging information by adding a request parameter of `?__WebLogicBridgeConfig` to any request. (Note: There are *two* underscore (_) characters after the `?`.) For security reasons, it is recommended that you set the `DebugConfigInfo` parameter to `OFF` in a production environment.	`OFF`
`connectionTimeout`	The amount of time, in seconds, that a socket waits in between reading chunks of data. If the timeout expires, a `java.io.InterruptedIOException` is thrown	0 = infinite timeout.
`numOfRetries`	Number of times `HttpClusterServlet` will attempt to retry a failed connection**.**	5
`pathTrim`	String to be trimmed from the beginning of the original URL.	None
`trimExt`	The file extension to be trimmed from the end of the URL.	None

`pathPrepend`	String to be prepended to the beginning of the original URL, after `pathTrim` has been trimmed, and before the request is forwarded to a WebLogic Server cluster member.	None

Cluster Configuration and Proxy Plug-ins

Two WebLogic Server configuration attributes can be set at the cluster level to control the behavior of the `HttpClusterServlet`.

- `WeblogicPluginEnabled`—If you set this attribute to `true` for a cluster that receives requests from the `HttpClusterServlet`, the servlet will respond to `getRemoteAddr` calls with the address of the browser client from the proprietary `WL-Proxy-Client-IP` header, instead of returning the web server address.
- `ClientCertProxy Enabled`—If you set this attribute to `true` for a cluster that receives requests from `HttpClusterServlet`, the plug-in sends client certs to the cluster in the special `WL-Proxy-Client-Cert` header, allowing user authentication to be performed on the proxy server.

For more information see help for the Cluster -->Configuration-->General page in *Administration Console Online Help*.

Verifying Your Configuration

To verify that your configuration of the `HttpClusterServlet` is functioning correctly:

1. Set the `DebugConfigInfo` parameter in `web.xml` to `ON`.
2. Use a Web browser to access the following URL:

   `http://`*`myServer`*`:`*`port/placeholder.jsp`*`?__WebLogicBridgeConfig`

   Where:

   `myServer` is the Managed Server on the proxy machine where `HttpClusterServlet` runs,

port is the port number on that server that is listening for HTTP requests, and

placeholder.jsp is a file that does not exist on the server.

The plug-in gathers configuration information and run-time statistics and returns the information to the browser. For more information, see "DebugConfigInfo" on page D-7.

11 Installing and Configuring the Apache HTTP Server Plug-In

The following sections describe how to install and configure the Apache HTTP Server Plug-In:

Overview

The Apache HTTP Server Plug-In allows requests to be proxied from an Apache HTTP Server to WebLogic Server. The plug-in enhances an Apache installation by allowing WebLogic Server to handle those requests that require the dynamic functionality of WebLogic Server.

The plug-in is intended for use in an environment where an Apache Server serves static pages, and another part of the document tree (dynamic pages best generated by HTTP Servlets or JavaServer Pages) is delegated to WebLogic Server, which may be operating in a different process, possibly on a different host. To the end user—the browser—the HTTP requests delegated to WebLogic Server still appear to be coming from the same source.

The HTTP-tunneling can also operate through the plug-in, providing non-browser clients access to WebLogic Server services.

The Apache HTTP Server Plug-In operates as an Apache module within an Apache HTTP Server. An Apache module is loaded by Apache Server at startup, and then certain HTTP requests are delegated to it. Apache modules are similar to HTTP servlets, except that an Apache module is written in code native to the platform.

As of WebLogic Server 6.1 SP6, 7.0 SP5, 8.1 SP2, the WebLogic Server plug-ins are now certified to proxy to any version of WebLogic Server, including 5.1.

Keep-Alive Connections in Apache Version 1.3.x

The Apache HTTP Server Plug-In creates a socket for each request and closes the socket after reading the response. Because Apache HTTP Server is multiprocessed, connection pooling and keep-alive connections between WebLogic Server and theApache HTTP Server Plug-In cannot be supported.

Keep-Alive Connections in Apache Version 2.x

The Apache HTTP Server Plug-In improves performance by using a reusable pool of connections from the plug-in to WebLogic Server. The plug-in implements HTTP 1.1 keep-alive connections between the plug-in and WebLogic Server by reusing the same connection in the pool for subsequent requests from the same client. If the connection is inactive for more than 30 seconds, (or a user-defined amount of time) the connection is closed and returned to the pool. You can disable this feature if desired. For more information, see "KeepAliveEnabled" on page -10.

Proxying Requests

The plug-in proxies requests to WebLogic Server based on a configuration that you specify. You can proxy requests either based on the URL of the request (or a portion of the URL). This is called proxying by *path*. You can also proxy request based on the *MIME type* of the requested file. You can also use a combination of both methods. If a request matches both criteria, the request is proxied by path. You can also specify additional parameters for each of these types of requests that define additional behavior of the plug-in. For more information, see "Configuring the Apache HTTP Server Plug-In" on page 11-10.

Certifications

The Apache HTTP Server Plug-In is supported on the Linux, Solaris, AIX, Windows, and HPUX11 platforms. Plug-ins are not supported on all operating systems for all releases. For information on platform support for specific versions of Apache, see Platform Support for WebLogic Server Plug-ins and Web Servers in *Supported Configurations for WebLogic Server 6.1*.

Installing the Apache HTTP Server Plug-In

You install the Apache HTTP Server Plug-In as an Apache module in your Apache HTTP Server installation. The module is installed either as a Dynamic Shared Object (DSO) or as a statically linked module. (Installation as a statically linked module is only available for Apache version 1.3.x). There are separate instructions in this section for each type of installation (DSO, or statically linked module).

Installing as a Dynamic Shared Object

To install the Apache HTTP Server Plug-In as a dynamic shared object:

1. Locate the shared object file for your platform.

   The Apache plug-in is distributed as a shared object (`.so`) for use on Solaris, Linux, Windows, and HPUX11 platforms. Each shared object file is distributed as separate versions, depending on the platform, whether or not SSL is to be used between the client and Apache, and the encryption strength for SSL (regular or 128 bit— 128 bit versions are only installed if you install the 128 bit version of WebLogic Server). The shared object files are located in the following directories of your WebLogic Server installation:

Table 1: Locations of Shared Object Files

Platform	Location of Shared Object File
Solaris	`lib/solaris`
Linux	`lib/linux`
Windows (Apache 2.0 only)	`bin/apache20`

Table 1: Locations of Shared Object Files

Platform	Location of Shared Object File

HPUX11	`lib/hpux11`
	
	Note: If you are running Apache 2.0.x server on HP-UX11, set the environment variables specified below before you build the Apache server. Because of a problem with the order in which linked libraries are loaded on HP-UX11, a core dump can result if the load order is not preset as an environment variable before building. Set the following environment variables:
	
	`export EXTRA_LDFLAGS="-lstd -lstream -lCsup -lm -lcl -ldld -lpthread"`
	
	Proceed with the `configure`, `make`, and `make install` steps:
	
	`./configure --prefix=$INSTALLATION_DIRECTORY --enable-so --with-mpm=worker`
	
	`make`
	
	`make install`
	
	See the Apache HTTP Server documentation for information about building and configuring your Apache server.

Choose the appropriate shared object from the following table (note that on the Compaq OpenVMS platform, the file suffix that is `.so` on other platforms is `.exe`):

Apache Version	Regular Strength Encryption	128-bit Encryption
Standard Apache Version 1.x	`mod_wl.so`	`mod_wl128.so`
Apache w/ SSL/EAPI Version 1.x (Stronghold, modssl etc.)	`mod_wl_ssl.so`	`mod_wl128_ssl.so`

Apache Version	Regular Strength Encryption	128-bit Encryption
Apache + Raven Version 1.x Required because Raven applies frontpage patches that makes the plug-in incompatible with the standard shared object	`mod_wl_ssl_raven.so`	`mod_wl128_ssl_raven.so`
Standard Apache Version 2.x	`mod_wl_20.so`	`mod_wl28_20.so`

If you are using Compaq OpenVMS, skip to step 5.

2. Enable the shared object.

   The Apache HTTP Server Plug-In will be installed as an Apache Dynamic Shared Object (DSO).

   DSO support in Apache is based on a module named `mod_so.c` that must be enabled before `mod_wl.so` is loaded. If you installed Apache using the supplied script, `mod_so.c` should already be enabled. To verify that `mod_so.c` is enabled, execute the following command:

   `APACHE_HOME``/bin/httpd -l`

   (Where *`APACHE_HOME`* is the directory containing your Apache HTTP Server installation.)

   This command lists all of the enabled modules. If `mod_so.c` is not listed, build your Apache HTTP Server from the source code, making sure that the following options are configured:

   ```
   ...
   --enable-module=so
   --enable-rule=SHARED_CORE
   ...
   ```

3. You install the Apache HTTP Server Plug-In with a support program called `apxs` (APache eXtenSion) that builds DSO-based modules outside of the Apache source tree, and adds the following line to the `httpd.conf` file:

   ```
   AddModule mod_so.c
   ```

4. In your WebLogic Server installation, use a command shell to navigate to the directory that contains the shared object for your platform and activate the `weblogic_module` by issuing this command (note that you must have Perl installed to run this Perl script):

   ```
   perl APACHE_HOME\bin\apxs -i -a -n weblogic mod_wl.so
   ```

   This command copies the `mod_wl.so` file to the *`APACHE_HOME`*`\libexec` directory. It also adds two lines of instructions for `weblogic_module` to the `httpd.conf` file and activates the module. Make sure that the following lines were added to your *`APACHE_HOME`*`/conf/httpd.conf` file in your Apache server installation:

   ```
   LoadModule weblogic_module         libexec/mod_wl.so
   AddModule mod_weblogic.c
   ```

5. If you are installing on the Compaq OpenVMS platform, be aware that its Apache modules use the extension `.exe` as opposed to .so used by other platforms. Thus the module libraries for OpenVMS are:

   - `mod_wl.exe`: Apache Plugin Module (NonSSL)
   - `mod_wl_ssl.exe`: Apache Plugin Module (SSL)

   As a result of this different suffix on OpenVMS you must manually add the `LoadModule` entries to `APACHE$ROOT:[CONF]httpd.conf` based on where the Apache plugin modules are copied. For example, if the modules are in `APACHE$ROOT:[000000]` directory, add the following entry to the `APACHE$ROOT:[CONF]httpd.conf` file:

   - For NonSSL configuration:

   ```
   LoadModule weblogic_module /apache$root/000000/mod_wl.exe
   ```

   - For SSL configuration:

   ```
   LoadModule weblogic_module /apache$root/000000/mod_wl_ssl.exe
   ```

   The following is a complete stanza added to the `httpd.conf` file in the section headed Dynamic Shared Object (DSO) Support:

   ```
   LoadModule weblogic_module modules/mod_wl.exe

   AddModule mod_weblogic.c
   ```

```
<IfModule mod_weblogic.c>
WebLogicHost [hostname]
WebLogicPort 7001
PathTrim /weblogic
</IfModule>
<Location /weblogic>
 SetHandler weblogic-handler
</Location>
```

For more information on editing the httpd.conf file, see "Configuring the Apache HTTP Server Plug-In" on page 11-10.

6. Configure any additional parameters in the Apache `httpd.conf` configuration file as described in the section "Configuring the Apache HTTP Server Plug-In" on page 11-10. The `httpd.conf` file allows you to customize the behavior of the Apache HTTP Server Plug-In.
7. Verify the syntax of the *`APACHE_HOME`*`\conf\httpd.conf` file with one of the following commands:

   For Apache 1.x, `APACHE_HOME\bin\apachect1 configtest`

   For Apache 2.x, `APACHE_HOME\bin\Apache -t`

The output of this command indicates any errors in your `httpd.conf` file.

8. Restart Weblogic Server.
9. Start (or restart if you have changed the configuration) Apache HTTP Server.
10. Test the Apache plug-in by opening a browser and setting the URL to the Apache Server + "`/weblogic/`", which should bring up the default WebLogic Server HTML page, welcome file, or default servlet, as defined for the default Web Application on WebLogic Server. For example:

    `http://myApacheserver.com/weblogic/`

Installing as a Statically Linked Module

To install the Apache HTTP Server Plug-In as a statically linked module:

1. Locate the linked library file for your platform.

   Each library file is distributed as separate versions, depending on the platform and the encryption strength for SSL (regular or 128-bit—128-bit versions are only installed if you install the 128-bit version of WebLogic Server). The library files are located in the following directories of your WebLogic Server installation:

   Solaris
   `lib/solaris`

   Linux
   `lib/linux`

   HPUX11
   `lib/hpux11`

   Choose the appropriate shared object from the following stable.

Apache Version	Regular Strength Encryption	128-bit Encryption
Standard Apache Version 1.3.x	`libweblogic.a`	`libweblogic128.a`

   If you are using the Gnu C Compiler (gcc), gcc 2.95.x is the recommended version.

2. Unpack the Apache distribution using the following command:

   ```
   tar -xvf apache_1.3.x.tar
   ```

3. Within the unpacked distribution switch to the `src/modules` directory.

4. Create a directory called `weblogic`.

5. Copy `Makefile.libdir, Makefile.tmpl` from the `lib` directory of your WebLogic Server installation to `src/modules/weblogic`.
6. Copy `libweblogic.a`. (Use `libweblogic128.a` instead, if you are using 128 bit security.) from the same directory containing the linked library file (see step 1.) to `src/modules/weblogic`.
7. If you are using regular strength encryption, execute the following command from the Apache 1.3 home directory:

   ```
   configure --activate-module=src/modules/weblogic/libweblogic.a
   ```

8. If you are using 128 bit encryption, execute the following command (on a single line):

   ```
   configure--activate-module=
    src/modules/weblogic/libweblogic128.a
   ```

9. Build the server. Invoke the compiler by executing the following command:

   ```
   make
   ```

10. Execute the following command:

    ```
    make install
    ```

11. Continue with step 7. in "Installing as a Dynamic Shared Object".

Configuring the Apache HTTP Server Plug-In

After you install the plug-in (see "Installing the Apache HTTP Server Plug-In" on page 11-4), edit the `httpd.conf` file to configure the Apache plug-in. Editing the `httpd.conf` file informs the Apache Web server that it should load the native library for the plug-in as an Apache module and also describes which requests should be handled by the module.

Editing the httpd.conf File

To edit the `httpd.conf` file to configure the Apache HTTP Server Plug-In:

1. Open the `httpd.conf` file. The file is located at *`APACHE_HOME`*`/conf/httpd.conf` (where *`APACHE_HOME`* is the root directory of your Apache installation).
2. Ensure that the `httpd.conf` LoadModule stanza will load the correct module by verifying that the following two lines were added to the `httpd.conf` file when you ran the `apxs` utility:

   ```
   LoadModule weblogic_module    libexec/mod_wl.so
   AddModule mod_weblogic.c
   ```

3. Add an `IfModule` block that defines one of the following:

   For a *non-clustered* WebLogic Server:

   The `WebLogicHost` and `WebLogicPort` parameters.

   For a *cluster* of WebLogic Servers:

   The `WebLogicCluster` parameter.

   For example:

   ```
   <IfModule mod_weblogic.c>
     WebLogicHost myweblogic.server.com
     WebLogicPort 7001
   </IfModule>
   ```

4. If you want to proxy requests by MIME type, also add a `MatchExpression` line to the `IfModule` block. (You can also proxy requests by path. Proxying by path takes precedence over proxying by MIME type. If you only want to proxy requests by path, skip to step 5.)

   For example, the following `IfModule` block for a non-clustered WebLogic Server specifies that all files with MIME type `.jsp` are proxied:

   ```
   <IfModule mod_weblogic.c>
     WebLogicHost myweblogic.server.com
     WebLogicPort 7001
     MatchExpression *.jsp
   </IfModule>
   ```

   You can also use multiple `MatchExpressions`, for example:

```
<IfModule mod_weblogic.c>
  WebLogicHost myweblogic.server.com
  WebLogicPort 7001
  MatchExpression *.jsp
  MatchExpression *.xyz
</IfModule>
```

If you are proxying requests by MIME type to a cluster of WebLogic Servers, use the `WebLogicCluster` parameter instead of the `WebLogicHost` and `WebLogicPort` parameters. For example:

```
<IfModule mod_weblogic.c>
  WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
  MatchExpression *.jsp
  MatchExpression *.xyz
</IfModule>
```

5. If you want to proxy requests by path, use the `Location` block and the `SetHandler` statement. `SetHandler` specifies the handler for the Apache HTTP Server Plug-In module. For example the following `Location` block proxies all requests containing the `/weblogic` in the URL:

```
<Location /weblogic>
  SetHandler weblogic-handler
</Location>
```

If you want to proxy requests to multiple clusters by path, you can use multiple `Location` blocks and `SetHandler` statements.

For example:

```
<Location /x>
SetHandler weblogic-handler
WebLogicCluster cluster1
</Location>
<Location /y>
SetHandler weblogic-handler
WebLogicCluster cluster2
</Location>
<Location /z>
SetHandler weblogic-handler
WebLogicCluster cluster3
```

```
</Location>
```

An alternate way of proxying by path to multiple clusters would be:

```
MatchExpression /x

WebLogicCluster=server1:port,server2:port,server3:port,server4:
port|PathTrim=/x

MatchExpression /y

WebLogicCluster=server1:port,server2:port,server3:port,server4:
port|PathTrim=/y

MatchExpression /z

WebLogicCluster=server1:port,server2:port,server3:port,server4:
port|PathTrim=/z
```

Where the general syntax is

```
MatchExpression exp name=value|name=value

where exp=Mime type(*.jsp) or exp=/x(path)
```

and the next argument in the list is a pipe(|) delimited list of name=value such as

WebLogicHost, WebLogicPort, WebLogicCluster, PathTrim, PathPrepend..

6. Define any additional parameters for the Apache HTTP Server Plug-In.

   The Apache HTTP Server Plug-In recognizes the parameters listed in "General Parameters for Web Server Plug-Ins" on page -2. To modify the behavior of your Apache HTTP Server Plug-In, define these parameters either:

   - In a `Location` block, for parameters that apply to proxying by *path*, or
   - In an `IfModule` block, for parameters that apply to proxying by *MIME type*.

Notes on Editing the httpd.conf File

- As an alternative to the procedure in "Editing the httpd.conf File" on page 11-11, you can define parameters in a separate file called `weblogic.conf` file that is *included* in the `IfModule` block. Using this included file may help modularize your configuration. For example:

  ```
  <IfModule mod_weblogic.c>
    # Config file for WebLogic Server that defines the parameters
  ```

```
  Include conf/weblogic.conf
</IfModule>
```

Note: Defining parameters in an *included* file is not supported when using SSL between Apache HTTP Server Plug-In and WebLogic Server.

- Each parameter should be entered on a new line. Do not put an '=' between the parameter and its value. For example:

```
PARAM_1 value1
PARAM_2 value2
PARAM_3 value3
```

- If a request matches both a MIME type specified in a `MatchExpression` in an `IfModule` block and a path specified in a `Location` block, the behavior specified by the `Location` block takes precedence.
- If you define the `CookieName` parameter, you must define it in an `IfModule` block.

Using SSL with the Apache Plug-In

You can use the Secure Sockets Layer (SSL) protocol to protect the connection between the Apache HTTP Server Plug-In and WebLogic Server. The SSL protocol provides confidentiality and integrity to the data passed between the Apache HTTP Server Plug-In and WebLogic Server. In addition, the SSL protocol allows the plug-in to authenticate itself to WebLogic Server to ensure that information is passed to a trusted principal.

The Apache HTTP Server Plug-In does *not* use the transport protocol (`http` or `https`) specified in the HTTP request (usually by the browser) to determine whether or not the SSL protocol is used to protect the connection between the Apache HTTP Server Plug-In and WebLogic Server.

Note: You cannot configure a 2-way SSL between the Apache HTTP Server and WebLogic Server. The SSL protocol is a point-to-point connection, cyptographically sealed end-to-end. Therefore, any type of proxy or firewall cannot see into the SSL socket. The Apache HTTP Server acts as the server end-point in the SSL connection. The configuration is:

```
client-->2-way SSL-->Apache<--1-way SSL<--WebLogic Server
```

The Apache HTTP Server cannot use the digital certificate from the first SSL connection in the second SSL connection because it cannot use the client's private key.

Configuring SSL Between the Apache HTTP Server Plug-In and WebLogic Server

To use the SSL protocol between Apache HTTP Server Plug-In and WebLogic Server:

1. Configure WebLogic Server for SSL. For more information, see "Configuring the SSL Protocol" on page 14-46.
2. Configure the WebLogic Server SSL listen port. For more information, see "Configuring the Listen Port" on page 8-5.
3. Set the `WebLogicPort` parameter in the `httpd.conf` file to the listen port configured in step 2.
4. Set the `SecureProxy` parameter in the `httpd.conf` file to `ON`.
5. Set any additional parameters in the `httpd.conf` file that define information about the SSL connection. For a complete list of parameters, see "SSL Parameters for Web Server Plug-Ins" on page -13.

Issues with SSL-Apache Configuration

Three known issues arise when you configure the Apache plug-in to use SSL:

- The `PathTrim` (see page D-4) parameter must be configured inside the `<Location>` tag.

  The following configuration is **incorrect**:

  ```
  <Location /weblogic>
   SetHandler weblogic-handler
  </Location>

  <IfModule mod_weblogic.c>
   WebLogicHost localhost
  ```

```
 WebLogicPort 7001
 PathTrim /weblogic
</IfModule>
```

The following configuration is the **correct** setup:

```
<Location /weblogic>
  SetHandler weblogic-handler
  PathTrim /weblogic
</Location>
```

- The `Include` directive does not work with Apache SSL. You must configure all parameters directly in the `httpd.conf` file. Do not use the following configuration when using SSL:

```
<IfModule mod_weblogic.c>
  MatchExpression *.jsp
  Include weblogic.conf
</IfModule>
```

- The current implementation of the WebLogic Server Apache plug-in does not support the use of multiple certificate files.

Specifying Trust of the WL-Proxy-Client-Cert Header

The plug-in can encode users' identity certifications in the `WL-Proxy-Client-Cert` header and pass the header to WebLogic Server instances (see Proxying Requests to Another HTTP Server in the WebLogic Server Administration Guide). A WebLogic Server instance uses the certificate information from that header, trusting that it comes from a secure source (the Plug-In), to authenticate the user. In previous releases of WebLogic Server, the default behavior was to always trust the `WL-Proxy-Client-Cert` header. Beginning with WebLogic Server 6.1 SP2, you need to explicitly define trust of the `WL-Proxy-Client-Cert` header. A new parameter, `clientCertProxy`, allows WebLogic Server to determine whether to trust the certificate header. For an additional level of security, use a connection filter to limit all connections into WebLogic Server (therefore allowing WebLogic Server to only accept connections from the machine on which the plug-in is running).

The `clientCertProxy` parameter has been added to the `HTTPClusterServlet` and Web applications.

For the `HTTPClusterServlet`, add the parameter to the `web.xml` file as follows:

```
<context-param>
```

```
        <param-name>clientCertProxy</param-name>
        <param-value>true</param-value>
</context-param>
```

For Web applications, add the parameter to the `web.xml` file as follows:

```
ServletRequestImpl context-param
<context-param>
        <param-name>weblogic.httpd.clientCertProxy</param-name>
        <param-value>true</param-value>
</context-param>
```

You can also use this parameter in a cluster as follows:

```
<Cluster ClusterAddress="127.0.0.1" Name="MyCluster"
        ClientCertProxyHeader="true"/>
```

Connection Errors and Clustering Failover

When the Apache HTTP Server Plug-In attempts to connect to WebLogic Server, the plug-in uses several configuration parameters to determine how long to wait for connections to the WebLogic Server host and, after a connection is established, how long the plug-in waits for a response. If the plug-in cannot connect or does not receive a response, the plug-in will attempt to connect and send the request to other WebLogic Servers in the cluster. If the connection fails or there is no response from any WebLogic Server in the cluster, an error message is sent.

Figure 11-1 “Connection Failover” on page 11-20 demonstrates how the plug-in handles failover.

Connection Failures

Failure of the host to respond to a connection request could indicate possible problems with the host machine, networking problems, or other server failures.

Failure of WebLogic Server to respond, could indicate that WebLogic Server is not running or is unavailable, a hung server, a database problem, or other application failure.

Failover with a Single, Non-Clustered WebLogic Server

If you are running only a single WebLogic Server the same logic described here applies, except that the plug-in only attempts to connect to the server defined with the `WebLogicHost` parameter. If the attempt fails, an `HTTP 503` error message is returned. The plug-in continues trying to connect to WebLogic Server until `ConnectTimeoutSecs` is exceeded.

The Dynamic Server List

When you specify a list of WebLogic Servers in the `WebLogicCluster` parameter, the plug-in uses that list as a starting point for load balancing among the members of the cluster. After the first request is routed to one of these servers, a dynamic server list is returned containing an updated list of servers in the cluster. The updated list adds any new servers in the cluster and deletes any that are no longer part of the cluster or that have failed to respond to requests. This list is updated automatically with the HTTP response when a change in the cluster occurs.

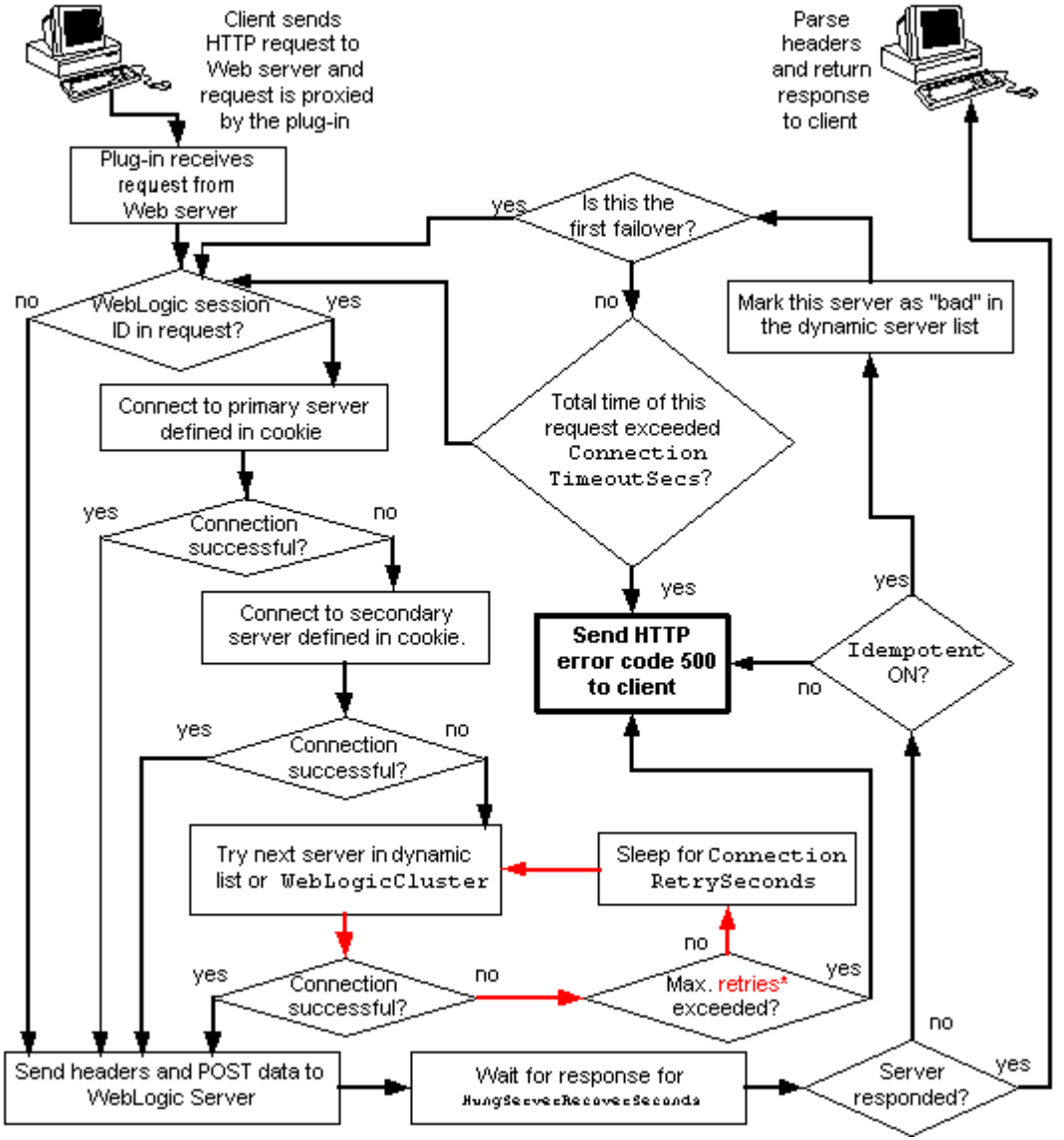
To avoid traffic on a new server you need to test, wait until the newly added server is fully tested, target it to the cluster and it will become a node of the cluster. This node will get automatically begin to receive traffic from the proxy.

Failover, Cookies, and HTTP Sessions

When a request contains a session information stored in a cookie, in the POST data, or by URL encoding, the session ID contains a reference to the specific server in which the session was originally established (called the *primary* server) and a reference to an additional server where the original session is replicated (called the *secondary* server). A request containing a cookie attempts to connect to the primary server. If that attempt fails, the request is routed to the secondary server. If both the primary and secondary servers fail, the session is lost and the plug-in attempts to make a fresh connection to another server in the dynamic cluster list. For more information, see Figure 11-1 “Connection Failover” on page 11-20.

Note: If the POST data is larger than 64K, the plugin will not parse the POST data to obtain the session ID. Therefore, if you store the session ID in the POST data, the plugin cannot route the request to the correct primary or secondary server, resulting in possible loss of session data.

Figure 11-1 Connection Failover



*The Maximum number of retries allowed in the red loop is equal to `ConnectTimeoutSecs` ÷ `ConnectRetrySecs`.

Template for the httpd.conf File

This section contains a sample `httpd.conf` file. You can use this sample as a template that you can modify to suit your environment and server. Lines beginning with # are comments. Note that Apache HTTP Server is not case sensitive, and that the `LoadModule` and `AddModule` lines are automatically added by the `apxs` utility.

```
#####################################################

APACHE-HOME/conf/httpd.conf file
#####################################################

LoadModule weblogic_module   libexec/mod_wl.so

AddModule mod_weblogic.c

<Location /weblogic>
  SetHandler weblogic-handler
  PathTrim /weblogic
  ErrorPage http://myerrorpage1.mydomain.com
</Location>

<Location /servletimages>
  SetHandler weblogic-handler
  PathTrim /something
  ErrorPage http://myerrorpage1.mydomain.com
</Location>

<IfModule mod_weblogic.c>
  MatchExpression *.jsp
  WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
  ErrorPage http://myerrorpage.mydomain.com
</IfModule>
```

Sample Configuration Files

Instead of defining parameters in the `location` block of your `httpd.conf` file, if you prefer, you can use a `weblogic.conf` file that is loaded by the `IfModule` in the `httpd.conf` file. The following examples may be used as templates that you can modify to suit your environment and server. Lines beginning with # are comments.

Example Using WebLogic Clusters

```
# These parameters are common for all URLs which are
# directed to the current module. If you want to override
# these parameters for each URL, you can set them again in
# the <Location> or <Files> blocks. (Except WebLogicHost,
# WebLogicPort, WebLogicCluster, and CookieName.)

<IfModule mod_weblogic.c>
  WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
  ErrorPage http://myerrorpage.mydomain.com
  MatchExpression *.jsp
</IfModule>
#####################################################
```

Example Using Multiple WebLogic Clusters

```
# These parameters are common for all URLs which are
# directed to the current module. If you want to override
# these parameters for each URL, you can set them again in
# the <Location> or <Files> blocks (Except WebLogicHost,
# WebLogicPort, WebLogicCluster, and CookieName.)

<IfModule mod_weblogic.c>
  MatchExpression *.jsp WebLogicHost=myHost|WebLogicPort=7001|Debug=ON
  MatchExpression *.html WebLogicCluster=myHost1:7282,myHost2:7283|ErrorPage=
    http://www.xyz.com/error.html
</IfModule>
```

Example Without WebLogic Clusters

```
# These parameters are common for all URLs which are
# directed to the current module. If you want to override
# these parameters for each URL, you can set them again in
# the <Location> or <Files> blocks (Except WebLogicHost,
# WebLogicPort, WebLogicCluster, and CookieName.)

<IfModule mod_weblogic.c>
  WebLogicHost myweblogic.server.com
  WebLogicPort 7001
  MatchExpression *.jsp
</IfModule>
```

Example Configuring IP-Based Virtual Hosting

```
NameVirtualHost 172.17.8.1
<VirtualHost goldengate.domain1.com>
WebLogicCluster tehama1:4736,tehama2:4736,tehama:4736
PathTrim /x1
ConnectTimeoutSecs 30
</VirtualHost>
<VirtualHost goldengate.domain2.com>
WeblogicCluster green1:4736,green2:4736,green3:4736
PathTrim /y1
ConnectTimeoutSecs 20
</VirtualHost>
```

Example Configuring Name-Based Virtual Hosting With a Single IP Address

```
<VirtualHost 162.99.55.208>
        ServerName myserver.mydomain.com
         <Location / >
               SetHandler weblogic-handler
               WebLogicCluster 162.99.55.71:7001,162.99.55.72:7001
               Idempotent ON
               Debug ON
               DebugConfigInfo ON
        </Location>
</VirtualHost>

<VirtualHost 162.99.55.208>
        ServerName myserver.mydomain.com
       <Location / >
                SetHandler weblogic-handler
                WebLogicHost russell
                WebLogicPort 7001
                Debug ON
                DebugConfigInfo ON
        </Location>
</VirtualHost>
```

12 Installing and Configuring the Microsoft Internet Information Server (ISAPI) Plug-In

The following sections describe how to install and configure the Microsoft Internet Information Server Plug-In.

Overview of the Microsoft Internet Information Server Plug-In

The Microsoft Internet Information Server Plug-In allows requests to be proxied from a Microsoft Internet Information Server (IIS) to WebLogic Server. The plug-in enhances an IIS installation by allowing WebLogic Server to handle those requests that require the dynamic functionality of WebLogic Server.

The Microsoft Internet Information Server Plug-In is intended for use in an environment where the Internet Information Server (IIS) serves static pages such as HTML pages, while dynamic pages such as HTTP Servlets or JavaServer Pages are served by WebLogic Server. The WebLogic Server may be operating in a different process, possibly on a different host. To the end user—the browser—the HTTP requests delegated to WebLogic Server still appear to be coming from IIS. The HTTP-tunneling facility of the WebLogic client-server protocol also operates through the plug-in, providing access to all WebLogic Server services.

As of WebLogic Server 6.1 SP6, 7.0 SP5, 8.1 SP2, the WebLogic Server plug-ins are now certified to proxy to any version of WebLogic Server, including 5.1.

Connection Pooling and Keep-Alive

The Microsoft Internet Information Server Plug-In improves performance by using a re-usable pool of connections from the plug-in to WebLogic Server. The plug-in implements HTTP 1.1 keep-alive connections between the plug-in and WebLogic Server by re-using the same connection for subsequent requests from the same client. If the connection is inactive for more than 30 seconds, (or a user-defined amount of time) the connection is closed. The connection with the client can be reused to connect to the same client at a later time if it has not timed out. You can disable this feature if desired. For more information, see “KeepAliveEnabled” on page -10.

Proxying Requests

The plug-in proxies requests to WebLogic Server based on a configuration that you specify. You can proxy requests either based on the URL of the request (or a portion of the URL). This is called proxying by *path*. You can also proxy a request based on the *MIME type* of the requested file, called proxying by file extension. You can also use a combination of both methods. If a request matches both criteria, the request is proxied by path. You can also specify additional parameters for each of these types of requests that define additional behavior of the plug-in. For more information, see "Installing the Microsoft Internet Information Server Plug-In" on page 12-3.

Platform Support

The Microsoft Internet Information Server Plug-In is supported on Windows. Plug-ins are not supported on all operating systems for all releases. For information on platform support for specific versions of Microsoft Internet Information Server Plug-In, see Platform Support for WebLogic Server Plug-ins and Web Servers in *Supported Configurations for WebLogic Server 6.1*.

Installing the Microsoft Internet Information Server Plug-In

To install the Microsoft Internet Information Server Plug-In:

1. Copy the `iisproxy.dll` file from the `/bin` directory of your WebLogic Server installation into a convenient directory that is accessible by IIS. This directory must also contain the `iisproxy.ini` file. If you need to use 128 bit security with the

Microsoft Internet Information Server plug-in, you must rename the `iisproxy128.dllfile to iisproxy.dll.` If you wish to keep both files you will need to change the name of the original `iisproxy.dll` file.

2. Start the IIS Internet Service Manager by selecting it from the Microsoft IIS Start menu.
3. In the left panel of the Service Manager, select your website (the default is "Default Web Site").
4. Click the "Play" arrow in the toolbar to start.
5. Open the properties for the selected website by holding the right mouse button down over the website selection in the left panel.
6. In the Properties panel, select the Home Directory tab, and click the Configuration button in the Applications Settings section.
7. Configure proxying by file extension:
   a. On the App Mappings tab, click the Add button to add file types and configure them to be proxied to WebLogic Server.
   b. In the dialog box, browse to find the "`iisproxy.dll`" file.
   c. Set the Extension to the type of file that you want to proxy to WebLogic Server.
   d. Deselect the "Check that file exists" check box.
   e. Set Execute Permissions to "Scripts and Executables".
   f. Set the Method exclusions as needed to create a secure installation.
   g. When you finish, click the OK button to save the configuration. Repeat this process for each file type you want to proxy to WebLogic.
   h. When you finish configuring file types, click the OK button to close the Properties panel.

   Note: Any path information you add to the URL after the server and port is passed directly to WebLogic Server. For example, if you request a file from IIS with the URL:

   `http://myiis.com/jspfiles/myfile.jsp`

   it is proxied to WebLogic Server with a URL such as

   `http://mywebLogic:7001/jspfiles/myfile.jsp`

Note: To avoid out-of-process errors, do not deselect the "Cache ISAPI Applications" check box.

8. Create the `iisproxy.ini` file.

   The `iisproxy.ini` file contains name=value pairs that define configuration parameters for the plug-in. The parameters are listed in "General Parameters for Web Server Plug-Ins" on page -2.

   Note: Changes in the parameters will not go into effect until you restart the "IIS Admin Service" (under *services*, in the control panel).

   BEA recommends that you locate the `iisproxy.ini` file in the same directory that contains the `iisproxy.dll` file. You can also use other locations. If you place the file elsewhere, note that WebLogic Server searches for `iisproxy.ini` in the following directories, in the following order:

   a. The same directory where `iisproxy.dll` is located.

   b. The home directory of the most recent version of WebLogic Server that is referenced in the Windows Registry. If WebLogic Server does not find the `iisproxy.ini` file there, it continues looking in the Windows Registry for older versions of WebLogic Server and looks for the `iisproxy.ini` file in the home directories of those installations.

   c. The directory `c:\weblogic`, if it exists.

9. Define the WebLogic Server host and port number to which the Microsoft Internet Information Server Plug-In proxies requests. Depending on your configuration, there are two ways to define the host and port:

   - If you are proxying requests to a single WebLogic Server, define the `WebLogicHost` and `WebLogicPort` parameters in the `iisproxy.ini` file. For example:

     ```
     WebLogicHost=localhost
     WebLogicPort=7001
     ```

   - If you are proxying requests to a cluster of WebLogic Servers, define the `WebLogicCluster` parameter in the `iisproxy.ini` file. For example:

     `WebLogicCluster=`*`myweblogic.com`*`:7001,`*`yourweblogic.com`*`:7001`

     Where *`myweblogic.com`* and `yourweblogic.com` are instances of Weblogic Server running in a cluster.

10. Configure proxying by path. In addition to proxying by file type, you can configure the Microsoft Internet Information Server Plug-In to serve files based on their *path* by specifying some additional parameters in the `iisproxy.ini` file. Proxying by path takes precedence over proxying by MIME type.

    You can also proxy multiple websites defined in IIS by path. For more information, see "Proxying Multiple Virtual Websites from IIS" on page 12-7.

    To configure proxying by path:

    a. Place the `iisforward.dll` file in the same directory as the `iisproxy.dll` file and add the `iisforward.dll` file as a filter service in IIS (WebSite *Properties* → ISAPI Filters tab → Add the `iisforward dll`).

    b. Register `.wlforward` as a special file type to be handled by `iisproxy.dll`.

    c. Define the property `WlForwardPath` in `iisproxy.ini`. `WlForwardPath` defines the path that is proxied to WebLogic Server, for example: `WlForwardPath=/weblogic`.

    d. Set the `PathTrim` parameter to trim off the `WlForwardPath` when necessary. For example, using

    ```
    WlForwardPath=/weblogic
    PathTrim=/weblogic
    ```

    trims a request from IIS to Weblogic Server. Therefore, `/weblogic/session` is changed to `/session`.

    e. If you want requests that do not contain extra path information (in other words, requests containing only a host name), set the `DefaultFileName` parameter to the name of the welcome page of the Web Application to which the request is being proxied. The value of this parameter is appended to the URL.

    f. If you need to debug your application, set the `Debug=ON` parameter in `iisproxy.ini`. A `c:\tmp\iisforward.log` is generated containing a log of the plug-in's activity that you can use for debugging purposes.

11. If you want to enable HTTP tunneling (optional), follow the instructions for proxying by path (see step 10 above), substituting the WebLogic Server host name and the WebLogic Server port number, or the name of a WebLogic Cluster that you wish to handle HTTP tunneling requests.

    ```
    WlForwardPath=*/HTTPClnt*
    ```

    You do not need to use the PathTrim parameter.

Note: The only time you need to use HTTP-tunneling is when you connect through an applet through IIS/NES to WebLogic Server and and use http as the protocol instead of t3. (For example, http:// as the protocol in the provider URL instead of t3://.)

12. Set any additional parameters in the `iisproxy.ini` file. A complete list of parameters is available in the appendix "General Parameters for Web Server Plug-Ins" on page -2.
13. If you are proxying servlets from IIS to WebLogic Server and you are not proxying by path, please read the section "Proxying Servlets from IIS to WebLogic Server" on page 12-12.

Proxying Multiple Virtual Websites from IIS

To proxy multiple Websites (defined as virtual servers in IIS) to WebLogic Server:

1. Create a new directory for each virtual server. This directory will contain `dll` and `ini` files used to define the proxy.
2. Copy `iisforward.dll` to each of the directories you created in step1.
3. Register the `iisforward.dll` for each Website with IIS.
4. Create a file called `iisforward.ini`. Place this file in the same directory that contains `iisforward.dll`. This file should contain the following entry for each virtual website defined in IIS:

   `vhost`*`N`*`=`*`websiteName`*`:`*`port`*
   `websiteName``:`*`port`*`=`*`dll_directory`*`/iisproxy.ini`

   Where:

   - *`N`* is an integer representing the virtual website. The first virtual website you define should use the integer 1 and each subsequent website should increment this number by 1.
   - *`websiteName`* is the name of the virtual website as registered with IIS.

- *port* is the port number where IIS listens for HTTP requests.
- *dll_directory* is the path to the directory you created in step 1.

For example:

```
vhost1=strawberry.com:7001
strawberry.com:7001=c:\strawberry\iisproxy.ini
vhost2=blueberry.com:7001
blueberry.com:7001=c:\blueberry\iisproxy.ini
...
```

5. Create a separate `iisproxy.ini` file for each virtual Website, as described in step 8. in "Proxying Requests". For each virtual Website, copy this `iispoxy.ini` file to the directory you created in step 1.
6. Copy `iisproxy.dll` to each directory you created in step 1.
7. In IIS, set the value for the Application Protection option to high (isolated). If the Application Protection option is set to Medium(pooled), the iisproxy.dll that registered as the first website will always be invoked. In this event, all the requests will be proxied to the same WLS instances defined in the iisproxy.ini of the first website.

Creating ACLs through IIS

ACLs will not work through the Microsoft Internet Information Server Plug-In if the Authorization header is not passed by IIS. Use the following information to ensure that the Authorization header is passed by IIS.

When using Basic Authentication, the user is logged on with local log-on rights. To enable the use of Basic Authentication, grant each user account the *Log On Locally* user right on the IIS server. Note the following two problems that may result from Basic Authentication's use of local logon.

- If the user does not have local log-on rights, Basic Authentication will not work even if the FrontPage, IIS, and Windows NT configurations appear to be correct.
- A user who has local log-on rights and who can obtain physical access to the host computer running IIS will be permitted to start an interactive session at the console.

To enable Basic Authentication, in the Directory Security tab of the console, ensure that the Allow Anonymous option is "on" and all other options are "off".

Sample iisproxy.ini File

Here is a sample `iisproxy.ini` file for use with a single, non-clustered WebLogic Server. Comment lines are denoted with the "#" character.

```
# This file contains initialization name/value pairs
# for the IIS/WebLogic plug-in.

WebLogicHost=localhost
WebLogicPort=7001
ConnectTimeoutSecs=20
ConnectRetrySecs=2
```

Here is a sample `iisproxy.ini` file with clustered WebLogic Servers. Comment lines are denoted with the "#" character.

```
# This file contains initialization name/value pairs
# for the IIS/WebLogic plug-in.

WebLogicCluster=myweblogic.com:7001,yourweblogic.com:7001
ConnectTimeoutSecs=20
ConnectRetrySecs=2
```

Note: If you are using SSL between the plug-in and WebLogic Server, the port number should be defined as the SSL listen port.

Using SSL with the Microsoft Internet Information Server Plug-In

You can use the Secure Sockets Layer (SSL) protocol to protect the connection between the WebLogic Server proxy plug-in and the Microsoft Internet Information Server. The SSL protocol provides confidentiality and integrity to the data passed

between the Microsoft Internet Information Server Plug-In and WebLogic Server. In addition, the SSL protocol allows the WebLogic Server proxy plug-in to authenticate itself to the Microsoft Internet Information Server to ensure that information is passed to a trusted principal.

The Microsoft Internet Information Server Plug-In does not use the transport protocol (`http` or `https`) to determine whether or not the SSL protocol will be used to protect the connection between the proxy plug-in and the Microsoft Internet Information Server. In order to use the SSL protocol with the Microsoft Internet Information Server Plug-In, configure the WebLogic Server receiving the proxied requests to use the SSL protocol. The port on the WebLogic Server that is configured for secure SSL communication is used by the WebLogic Server proxy plug-in to communicate with the Microsoft Internet Information Server.

Note: You cannot configure a 2-way SSL between the Microsoft Internet Information Server and WebLogic Server. The SSL protocol is a point-to-point connection, cyptographically sealed end-to-end. Therefore, any type of proxy or firewall cannot see into the SSL socket. The Microsoft Internet Information Server acts as the server end-point in the SSL connection. The configuration is:

```
client-->2-way SSL-->IIS<--1-way SSL<--WebLogic Server
```

The Microsoft Internet Information Server cannot use the digital certificate from the first SSL connection in the second SSL connection because it cannot use the client's private key.

Configuring SSL

To use the SSL protocol between Microsoft Internet Information Server Plug-In and WebLogic Server:

1. Configure WebLogic Server for SSL. For more information, see "Configuring the SSL Protocol" on page 14-46.
2. Configure the WebLogic Server SSL listen port. For more information, see "Configuring the Listen Port" on page 8-5.
3. Set the `WebLogicPort` parameter in the `iisproxy.ini` file to the listen port configured in step 2.

4. Set the `SecureProxy` parameter in the `iisproxy.ini` file to `ON`.
5. Set additional parameters in the `iisproxy.ini` file that define the SSL connection. For a complete list of parameters, see "SSL Parameters for Web Server Plug-Ins" on page -13.

For example:

```
WebLogicHost=myweblogic.com
WebLogicPort=7002
SecureProxy=ON
```

Specifying Trust of the WL-Proxy-Client-Cert Header

The plug-in can encode users' identity certifications in the `WL-Proxy-Client-Cert` header and pass the header to WebLogic Server instances (see Proxying Requests to Another HTTP Server in the WebLogic Server Administration Guide). A WebLogic Server instance uses the certificate information from that header, trusting that it comes from a secure source (the Plug-In), to authenticate the user. In previous releases of WebLogic Server, the default behavior was to always trust the `WL-Proxy-Client-Cert` header. Beginning with WebLogic Server 6.1 SP2, you need to explicitly define trust of the `WL-Proxy-Client-Cert` header. A new parameter, `clientCertProxy`, allows WebLogic Server to determine whether to trust the certificate header. For an additional level of security, use a connection filter to limit all connections into WebLogic Server (therefore allowing WebLogic Server to only accept connections from the machine on which the plug-in is running).

The `clientCertProxy` parameter has been added to the `HTTPClusterServlet` and Web applications.

For the `HTTPClusterServlet`, add the parameter to the `web.xml` file as follows:

```
<context-param>
        <param-name>clientCertProxy</param-name>
        <param-value>true</param-value>
</context-param>
```

For Web applications, add the parameter to the `web.xml` file as follows:

```
ServletRequestImpl context-param
```

```
<context-param>
        <param-name>weblogic.httpd.clientCertProxy</param-name>
        <param-value>true</param-value>
</context-param>
```

You can also use this parameter in a cluster as follows:

```
<Cluster ClusterAddress="127.0.0.1" Name="MyCluster"
        ClientCertProxyHeader="true"/>
```

Proxying Servlets from IIS to WebLogic Server

Servlets may be proxied by path if the `iisforward.dll` is registered as a filter. You would then invoke your servlet with a URL similar to the following:

```
http://IISserver/weblogic/myServlet
```

To proxy servlets if `iisforward.dll` is not registered as a filter, you must configure proxying by file type.To proxy servlets by file type:

1. Register an arbitrary file type (extension) with IIS to proxy the request to the WebLogic Server, as described in step 7. under "Installing the Microsoft Internet Information Server Plug-In" on page 12-3.
2. Register your servlet in the appropriate Web Application. For more information on registering servlets, see Configuring Servlets at `http://e-docs.bea.com/wls/docs61/webapp/components.html#configuring-servlets`.
3. Invoke your servlet with a URL formed according to this pattern:

   `http://www.myserver.com/`*`virtualName`*`/`*`anyfile`*`.`*`ext`*

   where *`virtualName`* is the URL pattern defined in the `<servlet-mapping>` element of the Web Application deployment descriptor (`web.xml`) for this servlet and *`ext`* is a file type (extension) registered with IIS for proxying to WebLogic Server. The *`anyfile`* part of the URL is ignored in this context.

Note:

- If the image links called from the servlet are part of the Web Application, you must also proxy the requests for the images to WebLogic Server by registering the appropriate file types (probably `.gif` and `.jpg`) with IIS. You can, however, choose to serve these images directly from IIS if desired.
- If the servlet being proxied has links that call other servlets, then these links must also be proxied to WebLogic Server, conforming to the pattern shown above.

Testing the Installation

After you install and configure the Microsoft Internet Information Server Plug-In, follow these steps for deployment and testing:

1. Make sure WebLogic Server and IIS are running.
2. Save a JSP file into the document root of the default Web Application.
3. Open a browser and set the URL to the IIS + `filename.jsp` as shown in this example:

   ```
   http://myii.server.com/filename.jsp
   ```

   If `filename.jsp` is displayed in your browser, the plug-in is functioning.

Connection Errors and Clustering Failover

When the Microsoft Internet Information Server Plug-In attempts to connect to WebLogic Server, the plug-in uses several configuration parameters to determine how long to wait for connections to the WebLogic Server host and, after a connection is established, how long the plug-in waits for a response. If the plug-in cannot connect or does not receive a response, the plug-in will attempt to connect and send the request to other WebLogic Servers in the cluster. If the connection fails or there is no response from any WebLogic Server in the cluster, an error message is sent.

Figure 12-1 “Connection Failover” on page 12-16 demonstrates how the plug-in handles failover.

Connection Failures

Failure of the host to respond to a connection request could indicate possible problems with the host machine, networking problems, or other server failures.

Failure of WebLogic Server to respond, could indicate that WebLogic Server is not running or is unavailable, a hung server, a database problem, or other application failure.

Failover with a Single, Non-Clustered WebLogic Server

If you are running only a single WebLogic Server the same logic described here applies, except that the plug-in only attempts to connect to the server defined with the `WebLogicHost` parameter. If the attempt fails, an `HTTP 503` error message is returned. The plug-in continues trying to connect to WebLogic Server until `ConnectTimeoutSecs` is exceeded.

The Dynamic Server List

When you specify a list of WebLogic Servers in the `WebLogicCluster` parameter, the plug-in uses that list as a starting point for load balancing among the members of the cluster. After the first request is routed to one of these servers, a dynamic server list is returned containing an updated list of servers in the cluster. The updated list adds any new servers in the cluster and deletes any that are no longer part of the cluster or that have failed to respond to requests. This list is updated automatically with the HTTP response when a change in the cluster occurs.

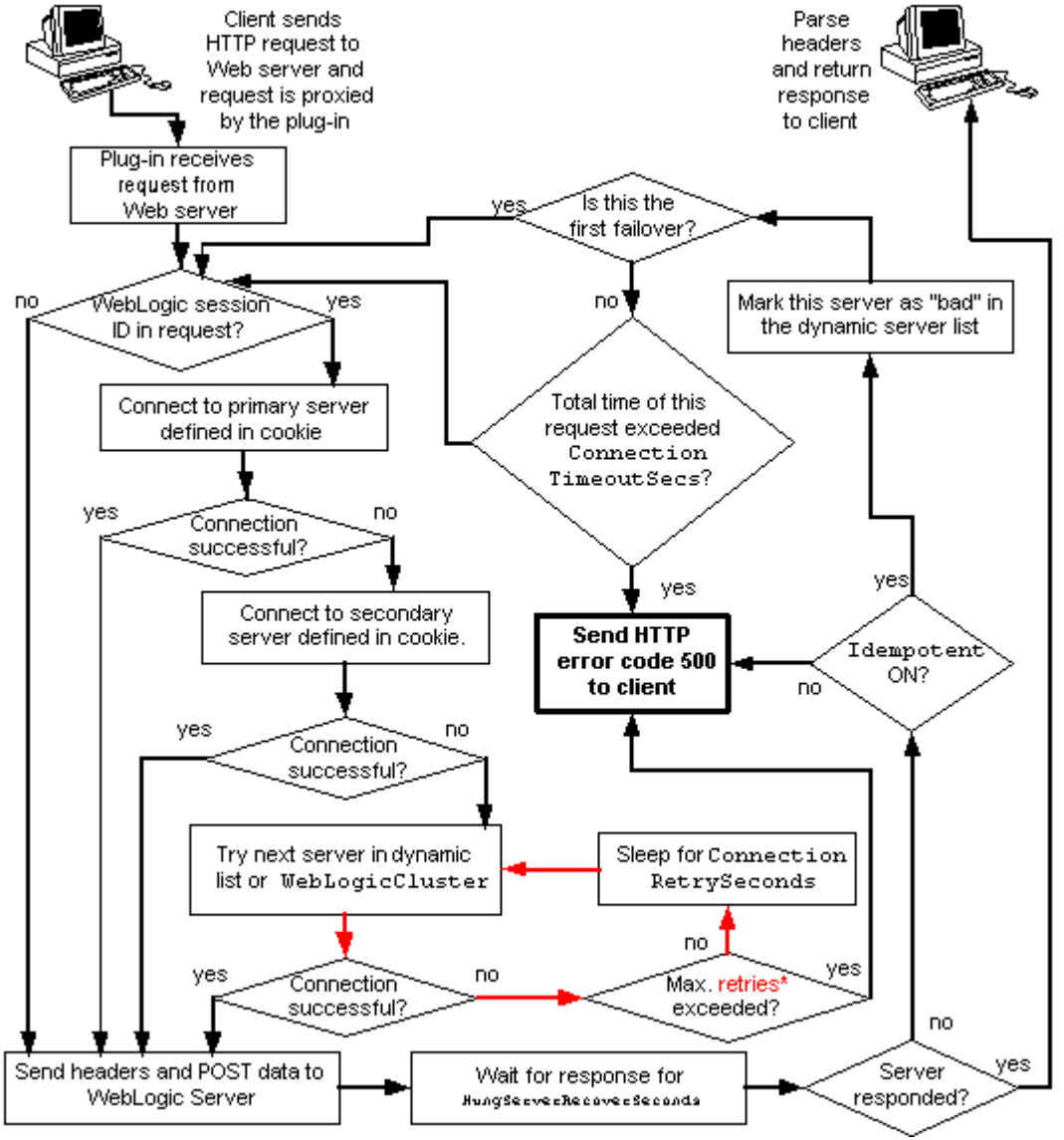
To avoid traffic on a new server you need to test, wait until the newly added server is fully tested, target it to the cluster and it will become a node of the cluster. This node will get automatically begin to receive traffic from the proxy.

Failover, Cookies, and HTTP Sessions

When a request contains a session information stored in a cookie, in the POST data, or by URL encoding, the session ID contains a reference to the specific server in which the session was originally established (called the *primary* server) and a reference to an additional server where the original session is replicated (called the *secondary* server). A request containing a cookie attempts to connect to the primary server If that attempt fails, the request is routed to the secondary server. If both the primary and secondary servers fail, the session is lost and the plug-in attempts to make a fresh connection to another server in the dynamic cluster list. For more information see Figure 12-1 “Connection Failover” on page 12-16.

Note: If the POST data is larger than 64K, the plugin will not parse the POST data to obtain the session ID. Therefore, if you store the session ID in the POST data, the plugin cannot route the request to the correct primary or secondary server, resulting in possible loss of session data.

Figure 12-1 Connection Failover



*The Maximum number of retries allowed in the red loop is equal to `ConnectTimeoutSecs` ÷ `ConnectRetrySecs`.

13 Installing and Configuring the Netscape Enterprise Server Plug-In (NSAPI)

The following sections describe how to install and configure the Netscape Enterprise Server Plug-In (NES) proxy plug-in:

Overview of the Netscape Enterprise Server Plug-In

The Netscape Enterprise Server Plug-In enables requests to be proxied from Netscape Enterprise Server (NES, also called iPlanet) to WebLogic Server. The plug-in enhances an NES installation by allowing WebLogic Server to handle those requests that require the dynamic functionality of WebLogic Server.

The Netscape Enterprise Server Plug-In is designed for an environment where Netscape Enterprise Server serves static pages, and a Weblogic Server (operating in a different process, possibly on a different host or hosts) is delegated to serve dynamic pages, such as JSPs or pages generated by HTTP Servlets. The connection between WebLogic Server and the Netscape Enterprise Server Plug-In is made using clear text or Secure Sockets Layer (SSL). To the end user—the browser—the HTTP requests delegated to WebLogic Server appear to come from the same source as the static pages. Additionally, the HTTP-tunneling facility of the WebLogic Server can operate through the Netscape Enterprise Server Plug-In, providing access to all WebLogic Server services (not just dynamic pages).

The Netscape Enterprise Server Plug-In operates as an NSAPI module (see `http://home.netscape.com/servers/index.html`) within a Netscape Enterprise Server. The NSAPI module is loaded by NES at startup, and then certain HTTP requests are delegated to it. NSAPI is similar to an HTTP (Java) servlet, except that a NSAPI module is written in code native to the platform.

For more information on supported versions of Netscape Enterprise Server and iPlanet servers, see the BEA WebLogic Server Platform Support Page.

As of WebLogic Server 6.1 SP6, 7.0 SP5, 8.1 SP2, the WebLogic Server plug-ins are now certified to proxy to any version of WebLogic Server, including 5.1.

Connection Pooling and Keep-Alive

The WebLogic Server NSAPI plug-in provides efficient performance by using a re-usable pool of connections from the plug-in to WebLogic Server. The NSAPI plug-in automatically implements “keep-alive” connections between the plug-in and

WebLogic Server. If a connection is inactive for more than 30 seconds or a user-defined amount of time, the connection is closed. You can disable this feature if desired. For more information, see "KeepAliveEnabled" on page -10.

Proxying Requests

The plug-in proxies requests to WebLogic Server are based on a configuration that you specify. You can proxy requests either based on the URL of the request (or a portion of the URL). This is called proxying by *path*. You can also proxy request based on the *MIME type* of the requested file. You can also use a combination of both methods. If a request matches both criteria, the request is proxied by path. You can also specify additional parameters for each of these types of requests that define additional behavior of the plug-in. For more information, see the next section.

Certifications

The Netscape Enterprise Server Plug-In is supported on Linux, Solaris, AIX, and Windows platforms. Plug-ins are not supported on all operating systems for all releases. For information on platform support for specific versions of Microsoft Internet Information Server Plug-In, see Platform Support for WebLogic Server Plug-ins and Web Servers in *Supported Configurations for WebLogic Server 6.1*.

Installing and Configuring the Netscape Enterprise Server Plug-In

To install and configure the Netscape Enterprise Server Plug-In:

1. Copy the library.

   The WebLogic NSAPI plug-in module is distributed as a shared object (`.so`) on UNIX platforms and as a dynamic-link library (`.dll`) on Windows. These files

are respectively located in the `/lib` or `/bin` directories of your WebLogic Server distribution. The modules are:

- Linux: `lib/linux/i686/libproxy.so`
- AIX: `lib/aix/libproxy4x.so` or `lib/aix/libproxy4x_128.so`
- Solaris: `lib/solaris/libproxy.so`
- Windows: `server/bin/proxy36.dll`

2. Modify the `obj.conf` file. The `obj.conf` file defines which requests are proxied to WebLogic Server and other configuration information. For details, see "Modifying the obj.conf File" on page 13-5.
3. If you are proxying requests by MIME type:
   a. Add the appropriate lines to the `obj.conf` file. For more information, see "Modifying the obj.conf File" on page 13-5.
   b. Add any new MIME types referenced in the `obj.conf` file to the `MIME.types` file. You can add MIME types by using the Netscape server console or by editing the `MIME.types` file directly.

      To directly edit the `MIME.types` file, open the file for edit and type the following line:

      ```
      type=text/jsp          exts=jsp
      ```

      Note: For NES 4.0 (iPlanet), instead of adding the MIME type for JSPs, change the existing MIME type from

      `magnus-internal/jsp`

      to

      `text/jsp.`

      To use the Netscape console, select Manage Preferences→ Mime Types, and make the additions or edits.
4. Deploy and test the Netscape Enterprise Server Plug-In
   a. Start WebLogic Server.
   b. Start Netscape Enterprise Server. If NES is already running, you must either restart it or apply the new settings from the console in order for the new settings to take effect.

c. To test the Netscape Enterprise Server Plug-In, open a browser and set the URL to the Enterprise Server + `/weblogic/`, which should bring up the default WebLogic Server HTML page, welcome file, or default servlet, as defined for the default Web Application on WebLogic Server, as shown in this example:

```
http://myenterprise.server.com/weblogic/
```

Modifying the obj.conf File

To use the Netscape Enterprise Server Plug-In, you must make several modifications to the NES `obj.conf` file. These modifications specify how requests are proxied to WebLogic Server. You can proxy requests by URL or by MIME type. The procedure for each is described later in this section.

The Netscape `obj.conf` file is very strict about the placement of text. To avoid problems, note the following regarding the `obj.conf` file:

- Eliminate extraneous leading and trailing white space. Extra white space can cause your Netscape server to fail.
- If you must enter more characters than you can fit on one line, place a backslash (\) at the end of that line and continue typing on the following line. The backslash directly appends the end of the first line to the beginning of the following line. If a space is necessary between the words that end the first line and begin the second line, be certain to use *one* space, either at the end of the first line (before the backslash), or at the beginning of the second line.
- Do not split attributes across multiple lines. (For example, all servers in a cluster must be listed in the same line, following `WebLogicCluster`.)
- If a required parameter is missing from the configuration, when the object is invoked it issues an HTML error that notes the missing parameter from the configuration.

To configure the `obj.conf` file:

1. Locate and open `obj.conf`.

   The `obj.conf` file for your NES instance is in the following location:

   `NETSCAPE_HOME``/https-`*`INSTANCE_NAME`*`/config/obj.conf`

Where *NETSCAPE_HOME* is the root directory of the NES installation, and *INSTANCE_NAME* is the particular "instance" or server configuration that you are using. For example, on a UNIX machine called myunixmachine, the obj.conf file would be found here:

```
/usr/local/netscape/enterprise-351/
 https-myunixmachine/config/obj.conf
```

2. Instruct NES to load the native library as an NSAPI module

   Add the following lines to the beginning of the magnus.conf file. These lines instruct NES to load the native library (the .so or .dll file) as an NSAPI module:

   ```
   Init fn="load-modules" funcs="wl_proxy,wl_init"\
   shlib=/usr/local/netscape/plugins/SHARED_LIBRARY
   Init fn="wl_init"
   ```

   Where *SHARED_LIBRARY* is the shared object or dll (for example libproxy.so) that you installed in step 1. under "Installing and Configuring the Netscape Enterprise Server Plug-In" on page 13-3. The function "load-modules" tags the shared library for loading when NES starts up. The values "wl_proxy" and "wl_init" identify the functions that the Netscape Enterprise Server Plug-In executes.

3. If you want to proxy requests by URL, (also called proxying by *path*.) create a separate <Object> tag for each URL that you want to proxy and define the PathTrim parameter. (To proxy requests by MIME type, see step 4.) Proxying by path supersedes proxying by MIME type. The following is an example of an <Object> tag that proxies a request containing the string */weblogic/*.

   ```
   <Object name="weblogic" ppath="*/weblogic/*">
   Service fn=wl_proxy WebLogicHost=myserver.com\
    WebLogicPort=7001 PathTrim="/weblogic"
   </Object>
   ```

   To create an <Object> tag to proxy requests by URL:

   a. Specify a name for this object (optional) inside the opening <Object> tag using the name attribute. The name attribute is informational only and is not used by the Netscape Enterprise Server Plug-In. For example:

      ```
      <Object name=myObject ...>
      ```

   b. Specify the URL to be proxied within the <Object> tag, using the ppath attribute. For example:

      ```
      <Object name=myObject ppath="*/weblogic/*>
      ```

The value of the `ppath` attribute can be any string that identifies requests intended for Weblogic Server. When you use a `ppath`, every request that contains that path is redirected. For example, a `ppath` of "`*/weblogic/*`" redirects every request that begins "`http://enterprise.com/weblogic`" to the Netscape Enterprise Server Plug-In, which sends the request to the specified Weblogic host or cluster.

c. Add the `Service` directive within the `<Object>` and `</Object>` tags. In the `Service` directive you can specify any valid parameters as name=value pairs. Separate multiple name=value pairs with *one and only one* space. For example:

```
Service fn=wl_proxy WebLogicHost=myserver.com\
 WebLogicPort=7001 PathTrim="/weblogic"
```

For a complete list of parameters, see "General Parameters for Web Server Plug-Ins" on page -2. You *must* specify the following parameters:

For a *non-clustered* WebLogic Server:
The `WebLogicHost` and `WebLogicPort` parameters.

For a *cluster* of WebLogic Server:
The `WebLogicCluster` parameter.

The `Service` directive should always begin with Service `fn=wl_proxy`, followed by valid name=value pairs of parameters.

Here is an example of the object definitions for two separate `ppaths` that identify requests to be sent to different instances of WebLogic Server:

```
<Object name="weblogic" ppath="*/weblogic/*">
Service fn=wl_proxy WebLogicHost=myserver.com\
 WebLogicPort=7001 PathTrim="/weblogic"
</Object>
```

```
<Object name="si" ppath="*/servletimages/*">
Service fn=wl_proxy WebLogicHost=otherserver.com\
 WebLogicPort=7008
</Object>
```

Note: Parameters that are not required, such as `PathTrim`, can be used to further configure the way the `ppath` is passed through the Netscape Enterprise Server Plug-In. For a complete list of plug-in parameters, see "General Parameters for Web Server Plug-Ins" on page -2.

4. If you want to proxy by MIME type, the MIME type must be listed in the `MIME.types` file. For instructions on modifying this file, see step 3. under "Installing and Configuring the Netscape Enterprise Server Plug-In" on page 13-3.

   All requests with a designated MIME type extension (for example, `.jsp`) can be proxied to the WebLogic Server, regardless of the URL.

   To proxy all requests of a certain file type to WebLogic Server:

   a. Add a `Service` directive to the existing `default Object` definition. (`<Object name=default ...>`)

      For example, to proxy all JSPs to a WebLogic Server, the following `Service` directive should be added *after* the last line that begins with:

      ```
      NameTrans fn=....
      ```

      and *before* the line that begins with:

      ```
      PathCheck.
      ```

      ```
      Service method="(GET|HEAD|POST|PUT)" type=text/jsp
      fn=wl_proxy\
       WebLogicHost=192.1.1.4 WebLogicPort=7001
      PathPrepend=/jspfiles
      ```

      This `Service` directive proxies all files with the `.jsp` extension to the designated WebLogic Server, where they are served with a URL like this:

      ```
      http://WebLogic:7001/jspfiles/myfile.jsp
      ```

      The value of the `PathPrepend` parameter should correspond to the context root of a Web Application that is deployed on the WebLogic Server or cluster to which requests are proxied.

      After adding entries for the Netscape Enterprise Server Plug-In, the default `Object` definition will be similar to the following example, with the additions shown in **`bold`**:

      ```
      <Object name=default>
      NameTrans fn=pfx2dir from=/ns-icons\
       dir="c:/Netscape/SuiteSpot/ns-icons"
      NameTrans fn=pfx2dir from=/mc-icons\
       dir="c:/Netscape/SuiteSpot/ns-icons"
      NameTrans fn="pfx2dir" from="/help" dir=\
       "c:/Netscape/SuiteSpot/manual/https/ug"
      NameTrans fn=document-root root="c:/Netscape/SuiteSpot/docs"
      Service method="(GET|HEAD|POST|PUT)" type=text/jsp\
      ```

```
 fn=wl_proxy WebLogicHost=localhost WebLogicPort=7001\
 PathPrepend=/jspfiles
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD) type=magnus-internal/imagemap\
 fn=imagemap
Service method=(GET|HEAD) \
 type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD) \
 type=*~magnus-internal/* fn=send-file
AddLog fn=flex-log name="access"
</Object>
```

b. Add a similar `Service` statement to the default object definition for all other MIME types that you want to proxy to WebLogic Server.

5. If you want to enable HTTP-tunneling (optional):

   Add the following object definition to the `obj.conf` file, substituting the WebLogic Server host name and the WebLogic Server port number, or the name of a WebLogic Cluster that you wish to handle HTTP tunneling requests.

```
<Object name="tunnel" ppath="*/HTTPClnt*">
Service fn=wl_proxy WebLogicHost=192.192.1.4\
 WebLogicPort=7001
</Object>
```

Using SSL with the NSAPI Plug-In

Use the Secure Sockets Layer (SSL) protocol to protect the connection between the Netscape Enterprise Server Plug-In, and WebLogic Server. The SSL protocol provides confidentiality and integrity to the data passed between the Netscape Enterprise Server Plug-In and WebLogic Server. In addition, the SSL protocol allows the WebLogic Server proxy plug-in to authenticate itself to the Netscape Enterprise Server to ensure that information is passed to a trusted principal.

The WebLogic Server proxy plug-in does *not* use the transport protocol (`http` or `https`) specified in the HTTP request (usually by the browser) to determine whether or not the SSL protocol will be used to protect the connection between the Netscape Enterprise Server Plug-In and WebLogic Server.

Note: You cannot configure a 2-way SSL between the Netscape Enterprise Server and WebLogic Server. The SSL protocol is a point-to-point connection, cyptographically sealed end-to-end. Therefore, any type of proxy or firewall cannot see into the SSL socket. The Netscape Enterprise Server acts as the server end-point in the SSL connection. The configuration is:

```
client-->2-way SSL-->NSAPI<--1-way SSL<--WebLogic Server
```

The Netscape Enterprise Server cannot use the digital certificate from the first SSL connection in the second SSL connection because it cannot use the client's private key.

Configuring SSL

To use the SSL protocol between Netscape Enterprise Server Plug-In and WebLogic Server:

1. Configure WebLogic Server for SSL. For more information, see "Configuring the SSL Protocol" on page 14-46.
2. Configure the WebLogic Server SSL listen port. For more information, see "Configuring the Listen Port" on page 8-5.
3. Set the `WebLogicPort` parameter in the `Service` directive in the `obj.conf` file to the listen port configured in step 2.
4. Set the `SecureProxy` parameter in the `Service` directive in the `obj.conf` file file to `ON`.
5. Set additional parameters in the `Service` directive in the `obj.conf` file that define information about the SSL connection. For a complete list of parameters, see "SSL Parameters for Web Server Plug-Ins" on page -13.

Specifying Trust of the WL-Proxy-Client-Cert Header

The plug-in can encode users' identity certifications in the `WL-Proxy-Client-Cert` header and pass the header to WebLogic Server instances (see Proxying Requests to Another HTTP Server in the WebLogic Server Administration Guide). A WebLogic Server instance uses the certificate information from that header, trusting that it comes from a secure source (the Plug-In), to authenticate the user. In previous releases of WebLogic Server, the default behavior was to always trust the `WL-Proxy-Client-Cert` header. Beginning with WebLogic Server 6.1 SP2, you need to explicitly define trust of the `WL-Proxy-Client-Cert` header. A new parameter, `clientCertProxy`, allows WebLogic Server to determine whether to trust the certificate header. For an additional level of security, use a connection filter to limit all connections into WebLogic Server (therefore allowing WebLogic Server to only accept connections from the machine on which the plug-in is running).

The `clientCertProxy` parameter has been added to the `HTTPClusterServlet` and Web applications.

For the `HTTPClusterServlet`, add the parameter to the `web.xml` file as follows:

```
<context-param>
        <param-name>clientCertProxy</param-name>
        <param-value>true</param-value>
</context-param>
```

For Web applications, add the parameter to the `web.xml` file as follows:

```
ServletRequestImpl context-param
<context-param>
        <param-name>weblogic.httpd.clientCertProxy</param-name>
        <param-value>true</param-value>
</context-param>
```

You can also use this parameter in a cluster as follows:

```
<Cluster ClusterAddress="127.0.0.1" Name="MyCluster"
        ClientCertProxyHeader="true"/>
```

Connection Errors and Clustering Failover

When the Netscape Enterprise Server Plug-In attempts to connect to WebLogic Server, the plug-in uses several configuration parameters to determine how long to wait for connections to the WebLogic Server host and, after a connection is established, how long the plug-in waits for a response. If the plug-in cannot connect or does not receive a response, the plug-in will attempt to connect and send the request to other WebLogic Servers in the cluster. If the connection fails or there is no response from any WebLogic Server in the cluster, an error message is sent.

Figure 13-1 “Connection Failover” on page 13-14 demonstrates how the plug-in handles failover.

Connection Failures

Failure of the host to respond to a connection request could indicate possible problems with the host machine, networking problems, or other server failures.

Failure of WebLogic Server to respond, could indicate that WebLogic Server is not running or is unavailable, a hung server, a database problem, or other application failure.

Failover with a Single, Non-Clustered WebLogic Server

If you are running only a single WebLogic Server the same logic described here applies, except that the plug-in only attempts to connect to the server defined with the `WebLogicHost` parameter. If the attempt fails, an `HTTP 503` error message is returned. The plug-in continues trying to connect to WebLogic Server until `ConnectTimeoutSecs` is exceeded.

The Dynamic Server List

When you specify a list of WebLogic Servers in the `WebLogicCluster` parameter, the plug-in uses that list as a starting point for load balancing among the members of the cluster. After the first request is routed to one of these servers, a dynamic server list is returned containing an updated list of servers in the cluster. The updated list adds any new servers in the cluster and deletes any that are no longer part of the cluster or that have failed to respond to requests. This list is updated automatically with the HTTP response when a change in the cluster occurs.

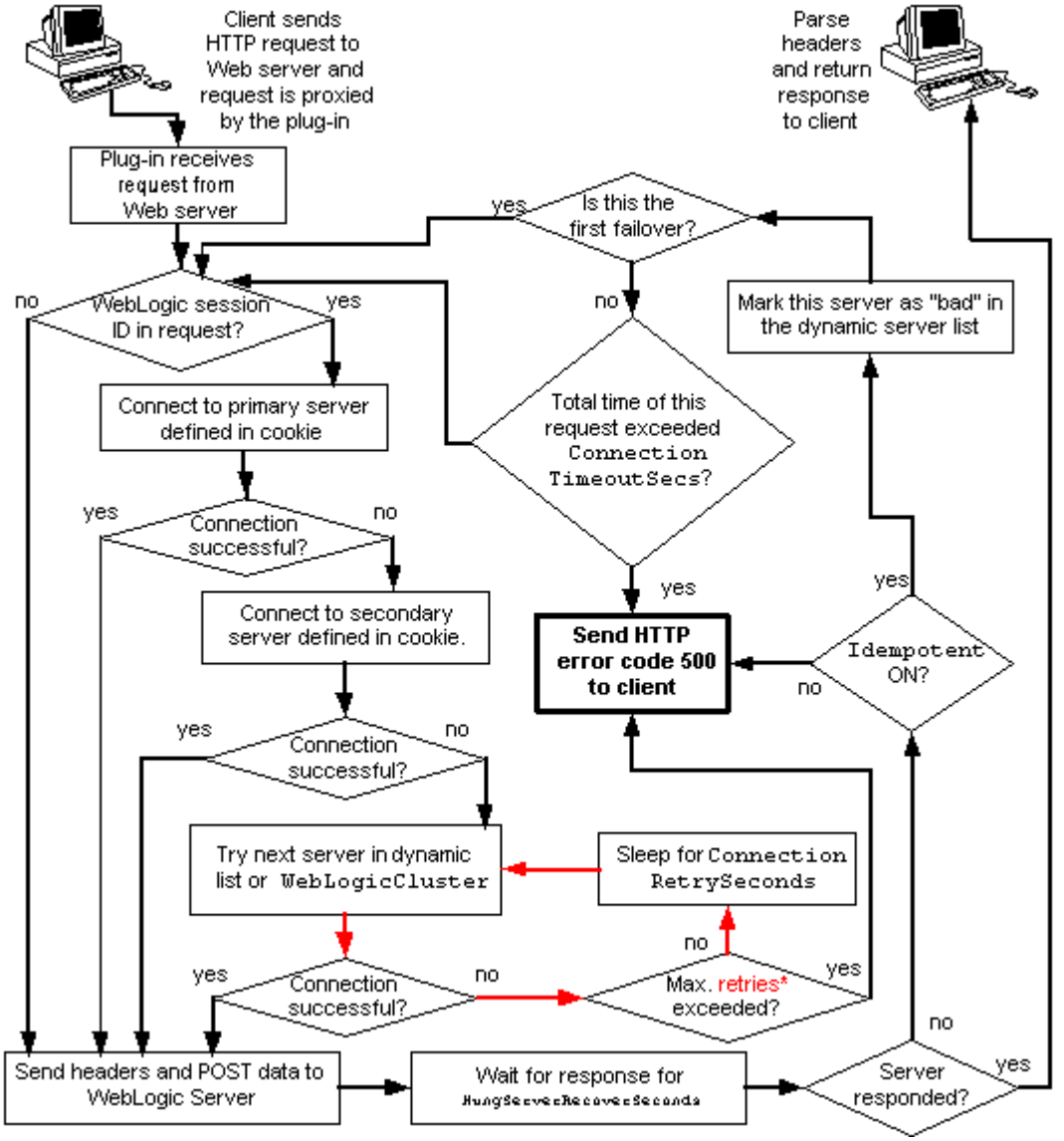
To avoid traffic on a new server you need to test, wait until the newly added server is fully tested, target it to the cluster and it will become a node of the cluster. This node will get automatically begin to receive traffic from the proxy.

Failover, Cookies, and HTTP Sessions

When a request contains session information stored in a cookie, in the POST data, or by URL encoding, the session ID contains a reference to the specific server in which the session was originally established (called the *primary* server) and a reference to an additional server where the original session is replicated (called the *secondary* server). A request containing a cookie attempts to connect to the primary server. If that attempt fails, the request is routed to the secondary server. If both the primary and secondary servers fail, the session is lost and the plug-in attempts to make a fresh connection to another server in the dynamic cluster list. For more information, see Figure 13-1 “Connection Failover” on page 13-14.

Note: If the POST data is larger than 64K, the plugin will not parse the POST data to obtain the session ID. Therefore, if you store the session ID in the POST data, the plugin cannot route the request to the correct primary or secondary server, resulting in possible loss of session data.

Figure 13-1 Connection Failover



*The Maximum number of retries allowed in the red loop is equal to `ConnectTimeoutSecs` ÷ `ConnectRetrySecs`.

Failover Behavior When Using Firewalls and Load Directors

In most configurations, the Netscape Enterprise Server Plug-In sends a request to the primary instance of a cluster. When that instance is unavailable, the request fails over to the secondary instance. However, in some configurations that use combinations of firewalls and load-directors, any one of the servers (firewall or load-directors) can accept the request and return a successful connection while the primary instance of WebLogic Server is unavailable. After attempting to direct the request to the primary instance of WebLogic Server (which is unavailable), the request is returned to the plug-in as “connection reset”.

Requests running through combinations of firewalls (with or without load-directors) are handled by WebLogic Server. In other words, responses of `connection reset` fail over to a secondary instance of WebLogic Server. Because responses of `connection reset` fail over in these configurations, servlets must be idempotent. Otherwise duplicate processing of transactions may result.

Sample obj.conf File (Not Using a WebLogic Cluster)

Below is an example of lines that should be added to the `obj.conf` file if you are not using a cluster. You can use this example as a template that you can modify to suit your environment and server. Lines beginning with # are comments.

Note: Make sure that you do not include any extraneous white space in the `obj.conf` file. Copying and pasting from the samples below sometimes adds extra white space, which can create problems when reading the file.

You can read the full documentation on Enterprise Server configuration files in the Netscape Enterprise Server Plug-In documentation.

```
## ------------ BEGIN SAMPLE OBJ.CONF CONFIGURATION  ---------
# (no cluster)

# The following line locates the NSAPI library for loading at
# startup, and identifies which functions within the library are
# NSAPI functions.  Verify the path to the library (the value
# of the shlib=<...> parameter) and that the file is
# readable, or the server fails to start.

Init fn="load-modules" funcs="wl_proxy,wl_init"\
 shlib=/usr/local/netscape/plugins/libproxy.so
Init fn="wl_init"

# Configure which types of HTTP requests should be handled by the
# NSAPI module (and, in turn, by WebLogic).  This is done
# with one or more "<Object>" tags as shown below.

# Here we configure the NSAPI module to pass requests for
# "/weblogic" to a WebLogic Server listening at port 7001 on
# the host myweblogic.server.com.

<Object name="weblogic" ppath="*/weblogic/*">
Service fn=wl_proxy WebLogicHost=myweblogic.server.com\
 WebLogicPort=7001 PathTrim="/weblogic"
</Object>

# Here we configure the plug-in so that requests that
# match "/servletimages/" is handled by the
# plug-in/WebLogic.
```

```
<Object name="si" ppath="*/servletimages/*">
Service fn=wl_proxy WebLogicHost=192.192.1.4 WebLogicPort=7001
</Object>

# This Object directive works by file extension rather than
# request path. To use this configuration, you must also add
# a line to the mime.types file:
#
# type=text/jsp             exts=jsp
#
# This configuration means that any file with the extension
# ".jsp" are proxied to WebLogic. Then you must add the
# Service line for this extension to the Object "default",
# which should already exist in your obj.conf file:

<Object name=default>
NameTrans fn=pfx2dir from=/ns-icons\
 dir="c:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons\
 dir="c:/Netscape/SuiteSpot/ns-icons"
NameTrans fn="pfx2dir" from="/help" dir=\
 "c:/Netscape/SuiteSpot/manual/https/ug"
NameTrans fn=document-root root="c:/Netscape/SuiteSpot/docs"
Service method="(GET|HEAD|POST|PUT)" type=text/jsp fn=wl_proxy\
 WebLogicHost=localhost WebLogicPort=7001 PathPrepend=/jspfiles
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD) type=magnus-internal/imagemap\
 fn=imagemap
Service method=(GET|HEAD) \
 type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD) type=*~magnus-internal/* fn=send-file
AddLog fn=flex-log name="access"
</Object>

# The following directive enables HTTP-tunneling of the
# WebLogic protocol through the NSAPI plug-in.

<Object name="tunnel" ppath="*/HTTPClnt*">
Service fn=wl_proxy WebLogicHost=192.192.1.4 WebLogicPort=7001
</Object>

#
## ------------- END SAMPLE OBJ.CONF CONFIGURATION ---------
```

Sample obj.conf File (Using a WebLogic Cluster)

Below is an example of lines that should be added to `obj.conf` if you are using a WebLogic Server cluster. You can use this example as a template that you can modify to suit your environment and server. Lines beginning with # are comments.

Note: Make sure that you do not include any extraneous white space in the `obj.conf` file. Copying and pasting from the samples below sometimes adds extra white space, which can create problems when reading the file.

Note: If you are proxying to more than one WebLogic Server cluster from a single Web server, each cluster must have a unique `CookieName` parameter, and each value should start with a unique string.

For more information, see the full documentation on Enterprise Server configuration files from Netscape.

```
## ------------- BEGIN SAMPLE OBJ.CONF CONFIGURATION ---------
# (using a WebLogic Cluster)
#
# The following line locates the NSAPI library for loading at
# startup, and identifies which functions within the library are
# NSAPI functions.  Verify the path to the library (the value
# of the shlib=<...> parameter) and that the file is
# readable, or the server fails to start.

Init fn="load-modules" funcs="wl_proxy,wl_init"\
 shlib=/usr/local/netscape/plugins/libproxy.so
Init fn="wl_init"

# Configure which types of HTTP requests should be handled by the
# NSAPI module (and, in turn, by WebLogic).  This is done
# with one or more "<Object>" tags as shown below.

# Here we configure the NSAPI module to pass requests for
# "/weblogic" to a cluster of WebLogic Servers.

<Object name="weblogic" ppath="*/weblogic/*">
Service fn=wl_proxy \
 WebLogicCluster="myweblogic.com:7001,yourweblogic.com:7001,\
```

```
 theirweblogic.com:7001" PathTrim="/weblogic"
</Object>

# Here we configure the plug-in so that requests that
# match "/servletimages/" are handled by the
# plug-in/WebLogic.

<Object name="si" ppath="*/servletimages/*">
Service fn=wl_proxy \
WebLogicCluster="myweblogic.com:7001,yourweblogic.com:7001,\
 theirweblogic.com:7001"
</Object>

# This Object directive works by file extension rather than
# request path. To use this configuration, you must also add
# a line to the mime.types file:
#
# type=text/jsp             exts=jsp
#
# This configuration means that any file with the extension
# ".jsp" is proxied to WebLogic. Then you must add the
# Service line for this extension to the Object "default",
# which should already exist in your obj.conf file:

<Object name=default>
NameTrans fn=pfx2dir from=/ns-icons\
 dir="c:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons\
 dir="c:/Netscape/SuiteSpot/ns-icons"
NameTrans fn="pfx2dir" from="/help" dir=\
 "c:/Netscape/SuiteSpot/manual/https/ug"
NameTrans fn=document-root root="c:/Netscape/SuiteSpot/docs"
Service method="(GET|HEAD|POST|PUT)" type=text/jsp fn=wl_proxy\
 WebLogicCluster="myweblogic.com:7001,yourweblogic.com:7001,\
 theirweblogic.com:7001",PathPrepend=/jspfiles
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD) type=magnus-internal/imagemap\
 fn=imagemap
Service method=(GET|HEAD) \
 type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD) type=*~magnus-internal/* fn=send-file
AddLog fn=flex-log name="access"
</Object>
```

```
# The following directive enables HTTP-tunneling of the
# WebLogic protocol through the NSAPI plug-in.

<Object name="tunnel" ppath="*/HTTPClnt*">
Service fn=wl_proxy WebLogicCluster="myweblogic.com:7001,\
 yourweblogic.com:7001,theirweblogic.com:7001"
</Object>

#
## ------------- END SAMPLE OBJ.CONF CONFIGURATION ---------
```

CHAPTER

14 Managing Security

The following sections describe how to implement security in WebLogic Server:

- Steps for Configuring Security
- Changing the System Password
- Specifying a Security Realm
- Defining Users
- Defining Groups
- Defining ACLs
- Configuring the SSL Protocol
- Configuring Mutual Authentication
- Configuring RMI over IIOP with SSL
- Protecting Passwords
- Installing an Audit Provider
- Installing a Connection Filter
- Setting Up the Java Security Manager
- Modifying the weblogic.policy File for Third Party or User-Written Classes
- Using the Recording Security Manager Utility
- Configuring Security Context Propagation
- SSL Certificate Validation

Steps for Configuring Security

Implementing security in a WebLogic Server deployment largely consists of configuring attributes that define the security policy for that deployment. WebLogic Server provides an Administration Console to help you define the security policy for your deployment. Using the Administration Console, you specify security-specific values for the following elements of your deployment:

- Security realms
- Users and Groups
- Access Control Lists (ACLs) and permissions for WebLogic Server resources
- SSL protocol
- Mutual authentication
- Host Name verification
- Audit providers
- Custom filters
- Security context propagation

Because security features are closely related, it is difficult to determine where to start when configuring security. In fact, defining security for your WebLogic Server deployment may be an iterative process. Although more than one sequence of steps may work, BEA Systems recommends the following procedure:

1. Change the password of the `system` User to protect your WebLogic Server deployment. See "Changing the System Password."
2. Specify a security realm. By default, WebLogic Server is installed with the File realm in place. However, you may prefer an alternate security realm or a custom security realm. See "Specifying a Security Realm."
3. Define Users for the security realm. You can organize Users further by implementing Groups in the security realm. See "Defining Users."
4. Define ACLs and permissions for the resources in your WebLogic Server deployment. See "Defining ACLs."

5. Protect the network connection between clients and WebLogic Server by implementing the SSL protocol. When SSL is implemented, WebLogic Server uses its digital certificate, issued by a trusted certificate authority, to authenticate clients. This step is optional but BEA recommends it. See "Configuring the SSL Protocol."

6. Further protect your WebLogic Server deployment by implementing mutual authentication. When mutual authentication is implemented, WebLogic Server must authenticate itself to the client and then the client in turn, must authenticate itself to WebLogic Server. Again, this step is an optional but BEA recommends it. See "Configuring Mutual Authentication."

For a complete description of WebLogic Server security features, see Introduction to WebLogic Security and Security Fundamentals.

Note: All configuration steps in this topic are based on the use of the Administration Console.

For information about assigning security roles to WebLogic EJBs, see WebLogic Server 6.1 Deployment Properties.

For information about security in WebLogic web applications, see Assembling and Configuring Web Applications.

Changing the System Password

During installation you specify a password for the `system` User. The specified password is associated with the `system` User in WebLogic Server and is stored in the `fileRealm.properties` file in the `\wlserver6.1\config\`*`domain`* directory where *`domain`* is the name specified as the WebLogic Administration domain name during installation. The specified password corresponds to the Administration Server for the domain and all the Managed Servers associated with that Administration Server.

Note: The `system` User is the only user account under which WebLogic Server can be started.

The password of the `system` User is encrypted and is further protected when WebLogic Server applies a hash to it. To improve security, BEA recommends frequently changing the system password that was set during installation. Each WebLogic Server deployment must have a unique password.

To change the system password, do the following:

1. In the Administration Console under the Security node, click Users to open the Users.
2. Under Change a User's Password, enter `system` in the Name attribute field.
3. Enter password you specified when installing WebLogic Server in the Old Password attribute field.
4. Enter a new password in the New Password attribute field.
5. Enter the new password again in the Confirm the Password attribute field.
6. Click Change.
7. Click the Changes You Have Made Must Be Saved to the Realm Implementation link.
8. Submit the change.
9. Reboot WebLogic Server.

When you use an Administration Server and Managed Servers in a domain, the Managed Server must always use the password for the Administration Server in the domain. Always change the password for the Administration Server through the Administration Console. After changing the password, restart all servers in the domain. The process is as follows:

1. Shutdown all the Managed Servers in the domain.
2. Change the system password on the Administration Server following the steps in this section.
3. Shutdown the Administration Server.
4. Reboot the Administration Server using the new system password.
5. Login into the Administration Console using the new system password.
6. Restart all the Managed Servers in the domain.

Maintaining the secrecy of WebLogic passwords is critical to keeping your WebLogic Server deployment and data secure. For your protection, BEA recommends keeping the password of WebLogic Server secret.

Specifying a Security Realm

This section describes configuring a security realm for your WebLogic Server deployment. For an introduction of security realms and how they are used in WebLogic Server, see Security Realms in *Programming WebLogic Security*. The following sections describe specifying a security realm:

- Configuring the File Realm
- Configuring the Caching Realm
- Configuring the LDAP Security Realm
- Configuring the Windows NT Security Realm
- Configuring the UNIX Security Realm
- Configuring the RDBMS Security Realm
- Installing a Custom Security Realm
- Migrating Security Realms

Configuring the File Realm

By default WebLogic Server is installed with the File realm in place. Before using the File realm, you need to define several attributes that govern the use of the File realm. You set these attributes on the Filerealm tab in the Security window of the Administration Console.

The following table describes each attribute on the Filerealm tab.

Table 14-1 File Realm Attributes

Attribute	Description
Caching Realm	Name of the Caching realm being used. ■ When using the File realm, this attribute should be set to None. ■ If you are using an alternate or custom security realm, set this attribute to the name of the Caching realm to be used. A list of configured Caching realms appears on the pull-down menu.
Max Users	Maximum number of Users to be used the File realm. The File realm is intended to be used with 10,000 or fewer Users. The minimum value for this attribute is 1 and the maximum value is 10,000. The default is 1,000.
Max Groups	Maximum number of Groups to be used with the File realm**.** The minimum value for this attribute is 1 and the maximum value is 10,000. The default is 1,000.
Max ACLs	Maximum number of ACLs to be used with the File realm. The minimum value for this attribute is 1 and the maximum value is 10,000. The default is 1,000.

Use the Manage Caching Realm button to clear the user, group, and ACL caches.

Caution: If the `fileRealm.properties` file becomes corrupted or is destroyed, you must reconfigure the security information for WebLogic Server. WebLogic Server cannot boot without a `fileRealm.properties` file.

The `fileRealm.properties` file contains default ACLs used to boot WebLogic Server. Even if you write a custom security realm, you still need a `fileRealm.properties` file to boot WebLogic Server since the custom security realm is not initially called during the start-up sequence.

Therefore, BEA recommends that you take the following steps:

Make a backup copy of the `fileRealm.properties` file and put it in a secure place.

Set the permissions on the `fileRealm.properties` file such that the administrator of the WebLogic Server deployment has write and read privileges and no other users have any privileges.

Note: Also make a backup copy of the `SerializedSystemIni.dat` file for the File realm. For more information about the `SerializedSystemIni.dat` file, see Protecting Passwords.

If, instead of the File realm, you want to use one of the alternate security realms provided by WebLogic Server or a custom security realm, set the attributes for the desired realm and reboot WebLogic Server.

Caution: If you use one of the alternate security realms, you must configure and enable the Caching realm; otherwise the alternate security realm will not work.

For more information about security realms in WebLogic Server, see Security Realms.

Configuring the Caching Realm

The Caching realm works with the File realm, alternate security realms, or custom security realms to fulfill client requests with the proper authentication and authorization. The Caching realm stores the results of both successful and unsuccessful realm lookups. It manages separate caches for Users, Groups, permissions, ACLs, and authentication requests. The Caching realm improves the performance of WebLogic Server by caching lookups, thereby reducing the number of calls into other security realms. For more information about security realms in WebLogic Server, see Security Realms.

The Caching realm is installed automatically when you install WebLogic Server: the cache is set up to delegate to the other security realms however caching is not enabled. You have to enable caching through the Administration Console.

Caution: If you use one of the alternate security realms, you must configure and enable the Caching realm; otherwise the alternate security realm will not work.

When you enable caching, the Caching realm saves the results of a realm lookup in its cache. Lookup results remain in the cache until either the specified number of seconds defined for the time-to-live (TTL) attributes has passed (the lookup result has expired) or the cache has filled. When the cache is full, new lookup results replace the oldest cached results. The TTL attributes determine how long a cached object is valid. The higher you set these attributes, the less often the Caching realm calls the secondary

security realm. Reducing the frequency of such calls improves the performance. The trade-off is that changes to the underlying security realm are not recognized until the cached object expires.

Note: When you obtain an object from a security realm, the object reflects a snapshot of the object. To update the object, call the object's `get()` method again. For example, the membership of a Group is set when the Group is retrieved from the security realm with a call to the `getGroup()` method. To update the members of the Group, you must call the `getGroup()` method again.

By default, the Caching realm operates on the assumption that the alternate security realm is case-sensitive. In a case-sensitive security realm, the owners of usernames `bill` and `Bill`, for example are treated as two distinct Users. The Windows NT Security realm and the LDAP Security realm are examples of security realms that are not case-sensitive. If you are using a security realm that is not case-sensitive, you must disable the CacheCaseSensitive attribute. When this attribute is set, the Caching realm converts usernames to lowercase so that WebLogic Server gives correct results for the security realm when it performs case-sensitive comparisons. When defining or referencing Users or Groups in a case-sensitive security realm, type usernames in lowercase.

To configure the Caching realm:

1. Go to the Security→Caching Realms node in the left pane of the Administration Console.
2. In the right pane of the Administration Console, click the Configure a New Caching Realm link.
3. Define the attributes on the General tab in the Caching Realm Configuration window.

   The following table describes the attributes you set on the General tab.

Table 14-2 Caching Realm Attributes on the General Tab

Attribute	Description
Name	Displays the active security realm as defined in the Administration Console. This attribute can not be changed.

Attribute	Description
Basic Realm	Name of the class for the alternate security realm or custom security realm to be used with the Caching realm. The names of the configured realms appear on the pull-down menu.
Case Sensitive Cache	Defines whether the specified security realm is case-sensitive. By default, this attribute is enabled: the realm is case-sensitive. To use a realm that is not case-sensitive (such as the Windows NT and LDAP security realms), you must disable this attribute.

4. Click Create.
5. Configure and enable the ACL cache by defining values for the attributes shown on the ACL tab in the Caching Realm Configuration window.

   The following table describes the attributes you set on the ACL tab.

Table 14-3 ACL Cache Attributes

Attribute	Description
Enable Cache	Option for enabling the ACL cache.
ACL Cache Size	Maximum number of ACL lookups to cache. This attribute should be a prime number for best lookup performance. The default is 211.
ACL Cache Positive TTL	Number of seconds to retain the results of a successful lookup. The default is 60 seconds.
ACL Cache Negative TTL	Number of seconds to retain the results of an unsuccessful lookup. The default is 10 seconds.

6. To save your changes, click the Apply button.

7. To enable and configure the Authentication cache, define values for the attributes shown on the Authentication tab in the Caching Realm Configuration window.

   The following table describes the attributes you set on the Authentication tab.

Table 14-4 Authentication Cache Attributes

Attribute	Description
Enable Authentication Cache	Option for enabling the Authentication cache.
Authentication Cache Size	Maximum number of Authenticate requests to cache. This attribute should be a prime number for best lookup performance. The default is 211.
Authentication Cache TTLPositive	Number of seconds to retain the results of a successful lookup. The default is 60 seconds.
Authentication Cache TTLNegative	Number of seconds to retain the results of an unsuccessful lookup. The default is 10 seconds.

8. To save your changes, click the Apply button.
9. To enable and configure the Group cache, define values for the attributes shown on the Groups tab in the Caching Realm Configuration window.

   The following table describes the attributes you set on the Group tab.

Table 14-5 Group Cache Attributes

Attribute	Description
Enable Group Cache	Option for enabling the Group cache.
Group Cache Size	Maximum number of Group lookups to cache. This attribute should be a prime number for best lookup performance. The default is 211.

Table 14-5 Group Cache Attributes

Attribute	Description
Group Cache TTLPositive	Number of seconds to retain the results of a successful lookup. The default is 60 seconds.
Group Cache TTLNegative	Number of seconds to retain the results of an unsuccessful lookup. The default is 10 seconds.
Group Membership Cache TTL	Number of seconds to store the members of a group before updating it. The default is 300 seconds.

10. To save your changes, click the Apply button.
11. To enable and configure the User cache, define values for the attributes shown on the User tab in the Caching Realm Configuration window.

    The following table describes the attributes you set on the User tab.

Table 14-6 User Cache Attributes

Attribute	Description
Enable User Cache	Option for enabling the User cache.
User Cache Size	Maximum number of User lookups to cache. This attribute should be a prime number for best lookup performance. The default is 211.
User Cache TTLPositive	Number of seconds to retain the results of a successful lookup. The default is 60 seconds.
User Cache TTLNegative	Number of seconds to retain the results of an unsuccessful lookup. The default is 10 seconds.

12. To save your changes, click the Apply button.

13. To enable and configure the Permission cache, define values for the attribute shown on the Permission tab in the Caching Realm Configuration window. T

    The following table describes each attribute on the Permission tab.

Table 14-7 Permission Cache Attributes

Attribute	Description
Enable Permission Cache	Option for enabling the Permission cache.
Permission Cache Size	Maximum number of Permission lookups to cache. This attribute should be a prime number for best lookup performance. The default is 211.
Permission Cache TTLPositive	Number of seconds to retain the results of a successful lookup. The default is 60 seconds.
Permission Cache TTLNegative	Number of seconds to retain the results of an unsuccessful lookup. The default is 10 seconds.

14. To save your changes, click the Apply button.
15. When you finish defining attributes for the Caching realm, reboot WebLogic Server.

Configuring the LDAP Security Realm

The LDAP security realm provides authentication through a Lightweight Directory Access Protocol (LDAP) server. This server allows you to manage all the users for your organization in one place: the LDAP directory. The LDAP security realm supports Open LDAP, Netscape iPlanet, Microsoft Site Server, and Novell NDS.

In this release of WebLogic Server, you can choose between two versions of the LDAP security realm:

- LDAP realm V1—The LDAP security realm packaged in previous releases of WebLogic Server. With the exception of Microsoft Site Server, the LDAP security realm V1 works with all the supported LDAP servers and is provided

for BEA customers that are currently using the LDAP security realm in an older release of WebLogic Server. However, the LDAP realm V1 is deprecated in this release and BEA recommends users upgrade to the LDAP realm V2.

- LPAP realm V2—An updated LDAP security realm with improved performance and configurability. This is the same LDAP security realm provided in WebLogic Server 6.0 Service Pack 1.0. LDAP realm V2 does not support `getUsers()` or `getGroups()` due to the fact that allocating memory to fulfill those requests can cause a denial of service vulernability. If you want to use those functions, BEA recommends using LDAP realm V1 When running Windows 2000, BEA recommends using LDAP realm V2 to authenticate against the Windows 2000 User and Group store.

Note: When using LDAP realm V1 you can view Users and members of a Group stored in the LDAP directory server through the Administration Console. However, when using LDAP realm V2, you can only view the Groups stored in the LDAP directory server through the Administration Console.

You need to use the administration tools available with the LDAP server to manage Users and Groups (for example, adding or deleting Users or Groups or adding members to Groups). If you make a change in the LDAP directory store, reset the User cache and the Group cache to immediately view your changes in the Administration Console.

The following suggestions are ways to improve the performance of the LDAP Security realm:

- Use the filters in the `ldaprealm.props` file to obtain smaller and more specific results sets from the LDAP server (supported for LDAP realm V2 only).
- Have the LDAP server index all of the attributes that you use as search keys in your LDAP realm search filters. Not indexing the attributes could cause linear search performance.
- Use the Caching realm carefully. Changes in the LDAP server's information will not be propagated to the LDAP Security realm until the cache is cleared.

Configuring the LDAP security realm involves defining attributes that enable the LDAP Security realm in WebLogic Server to communicate with the LDAP server and attributes that describe how Users and Groups are stored in the LDAP directory. The LDAP tree and schema is different for every LDAP server. Therefore, the LDAP realm V2 provides a set of templates that define default attributes for the supported LDAP servers.

Restrictions When Using the LDAP Security Realm

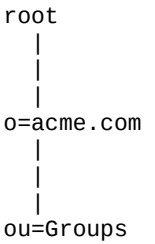
The LDAP security realm has the following restrictions:

- When the LDAP server in Microsoft Site Server is installed and the root of the LDAP directory is created, a number of organizational units are created by default. Under Groups there is a default organization unit called `NTGroups` with a default Group named `Administrators`, which is empty. By default, WebLogic Server also provides a Group called `Administrators` that contains a member `System` which is the User under which WebLogic Server is started. If you use the defaults in Microsoft Site Server and start creating your own Groups under the default organizational unit, WebLogic Server will not start. In order to start WebLogic Server with the LDAP security realm, you need to create your own unique organizational unit in the LDAP directory and create Groups for your WebLogic Server deployment under that organizational unit.
- If you have two Groups within the LDAP directory with the same name, WebLogic Server cannot properly authenticate the Users in the second Group that it locates. The LDAP security realm uses the Group's distinguished name (DN) to locate Groups in the LDAP directory. If you create more than one group with the same name, WebLogic Server only authenticates the Users in the first Group it locates. You must use unique Group names when using the LDAP security realm.
- The LDAP realm V2 does not provide the following functionality provided in LDAP realm V1:
  - Listing all Users
  - Listing the members of a Group
  - The authProtocol and userAuthentication mechanisms have been removed. You need to use the JNDI bind mechanism to pass security credentials to the LDAP server.
- The LDAP realm V2 has issues with the Open LDAP Server when running the `getGroups()` method for large numbers of groups (more than 300). This problem is due to caching bugs in Open LDAP.

Locating Users and Groups in the LDAP Directory

The LDAP security realm needs to know where the Users and Groups are stored in the LDAP directory used with the security realm. This is done by specifying the distinguished names (DNs) of the LDAP directories that contain the Users and Groups.

In LDAP, a DN starts with a leaf node and goes to the root node. For example:



The DN for this branch would be specified as `ou=Groups, o=acme.com`.

In LDAP realm V1, you specify DNs via the GroupDN and UserDN attributes when configuring the security realm. However, you must reverse the DNs. For example, the sample DN would be specified as `groupDN="o=acme.com, ou=Groups"`.

In LDAP realm V2, you specify DNs by adding `user.dn` and `group.dn` properties to the Configuration attribute of the CustomRealm MBean. Unlike LDAP realm V1, you do not have to reverse the DN. For example, the `user.dn` and `group.dn` properties for a LDAP realm V2 are specified as follows:

```
ConfigurationData="..., group.dn=ou=Groups, o=acme.com, ..."
```

A common error when switching between the LDAP realm V1 and LDAP realm V2 is copying over the reverse DNs thus causing the LDAP security realm to stop working. Check your DN specifications when migrating from LDAP realm V1 to LDAP realm V2.

Configuring an LDAP Realm V1

To use the LDAP Security realm V1 instead of the File realm:

1. Go to the Security→Realms node in the left pane of the Administration Console.
2. In the right pane of the Administration Console, click the Configure a New LDAP Realm V1 link.

The name of the class that implements the LDAP Security realm is displayed.

3. Click Create.
4. To enable communication between the LDAP server and WebLogic Server define values for the attributes shown on the LDAP Realm V1 tab in the LDAP Realm Create window.

   The following table describes the attributes you set on the LDAP Realm V1 tab.

Table 14-8 LDAP Security Realm Attributes on the LDAP Tab

Attribute	Description
LDAPURL	Location of the LDAP server. Change the URL to the name of the computer on which the LDAP server is running and the number of the port at which it is listening. For example: `ldap://ldapserver:385`. If you want WebLogic Server to connect to the LDAP server using the SSL protocol, use the LDAP server's SSL port in the URL.
Principal	Distinguished name (DN) of the LDAP User used by WebLogic Server to connect to the LDAP server. This user must be able to list LDAP Users and Groups.
Credential	Password that authenticates the LDAP User defined in the Principal attribute.
Enable SSL	Option for enabling the SSL protocol to protect communications between the LDAP server and WebLogic Server. Keep in mind the following guidelines: ■ Disable this attribute if the LDAP server is not configured to use the SSL protocol. ■ If you set the UserAuthentication attribute on the LDAP Users tab to `external`, this attribute must be enabled.

Table 14-8 LDAP Security Realm Attributes on the LDAP Tab

Attribute	Description
AuthProtocol	The type of authentication used to authenticate the LDAP server. Set this attribute to one of the following values: ■ `None` for no authentication ■ `Simple` for password authentication ■ `CRAM-MD5` for certificate authentication Netscape iPlanet supports `CRAM-MD5`. Microsoft Site Server, Netscape iPlanet, and OpenLDAP and Novell NDS support `Simple`.

5. To save your changes, click the Apply button.
6. To specify how Users are stored in the LDAP directory define the attributes shown on the Users tab in the LDAP Realm Create window.

   The following table describes the attributes you set on the Users tab.

Table 14-9 LDAP Security Realm Attributes on the Users Tab

Attribute	Description
User Authentication	Determines the method for authenticating Users. Set this attribute to one of the following values: ■ `Bind` specifies that the LDAP security realm retrieves user data, including the password for the LDAP server, and checks the password in WebLogic Server. ■ `External` specifies that the LDAP security realm authenticates a User by attempting to bind to the LDAP server with the username and password supplied by the WebLogic Server client. If you choose the `External` setting, you must also use the SSL protocol. ■ `Local` specifies that the LDAP security realm authenticates a User by looking up the UserPassword property in the LDAP directory and checking it against the passwords in WebLogic Server. If you are using Netscape iPlanet, set this attribute to `Bind`.
User Password Attribute	If the User Authentication attribute is set to Local, this attribute is used to find out what LDAP property contains the passwords for the LDAP users.
User DN	A list of attributes and their values that, when combined with the attributes in the `User Name Attribute` attribute, uniquely identifies an LDAP User.
User Name Attribute	The login name of the LDAP User. The value of this attribute can be the common name of an LDAP User but usually it is an abbreviated string, such as the common name.

7. To save your changes, click the Apply button.

8. To specify how Groups are stored in the LDAP directory, assign values to the attributes shown on the Groups tab in the LDAP Realm Create window.

   The following table describes the attributes you set on the Groups tab.

Table 14-10 LDAP Security Realm Attribute on the Groups Tab

Attribute	Description
Group DN	List of attributes and values that, combined with the Group Name Attribute attribute, uniquely identifies a Group in the LDAP directory. For example, `"o=acme.com, ou=Groups"`.
Group Name Attribute	Name of a Group in the LDAP directory. It is usually a common name.
Group Is Context	Boolean checkbox that specifies how Group membership is recorded in the LDAP directory. ■ Check this checkbox if each Group entry contains one User. By default, the box is selected. ■ Uncheck this checkbox if one Group entry contains an attribute for each Group member.
Group Username Attribute	Name of the LDAP attribute that contains a Group member in a Group entry.

9. To save your changes, click the Apply button.
10. When you have finished defining all the attributes, reboot WebLogic Server.
11. Configure the Caching realm. For more information, see "Configuring the Caching Realm."

    Note: When you use an LDAP Security realm, you must configure and enable the Caching realm; otherwise, the LDAP Security realm will not work.

    When configuring the Caching realm, select LDAP Realm from the pull-down menu for the Basic attribute on the General tab. The Basic attribute defines the association between the Caching realm and the alternate security realm (in this case, the LDAP Realm V1).

12. Go to the Security node.

13. Choose the Filerealm tab.
14. In the Caching Realm attribute, choose the name of the Caching Realm to be used with the LDAP Security realm. A list of configured Caching Realms appears on the pull-down menu.

    Note: When you use an LDAP Security realm, you must configure and enable the Caching realm; otherwise, the LDAP Security realm will not work.

15. Reboot WebLogic Server.

The Caching realm caches Users and Groups internally to avoid frequent lookups in the LDAP directory. Each object in the Users and Groups caches has a TTL attribute that you set when you configure the Caching realm. If you make changes in the LDAP directory, those changes are not reflected in the LDAP Security realm until the cached object expires or is flushed from the cache. The default TTL is 10 seconds for unsuccessful lookups and 60 seconds for successful lookups. Unless you change the TTL attributes for the User and Group caches, changes in the LDAP directory should be reflected in the LDAP Security realm in 60 seconds.

If some server-side code has performed a lookup in the LDAP Security realm, such as a `getUser()` call on the LDAP Security realm, the object returned by the realm cannot be released until the code releases it. Therefore, a User authenticated by WebLogic Server remains valid as long as the connection persists, even if you delete the user from the LDAP directory.

Configuring an LDAP Realm V2

Configuring the LDAP Realm V2 involves defining attributes that enable the security realm to communicate with the LDAP server and describe where users and groups are stored in the LDAP directory. The LDAP tree and schema is different for every LDAP server. WebLogic Server provides templates for the supported LDAP servers. These templates specify default configuration information used to represent Users and Groups in each of the supported LDAP servers. For more information, see “Supported LDAP Server Templates” on page 14-22.

To configure a LDAP security realm V2, you choose the template that corresponds to the LDAP server you want to use and modify it to specify information about your specific configuration.

To use a LDAP Security realm V2:

1. Go to the Security→Realms node in the left pane of the Administration Console.

2. Choose the LDAP server you want to use with WebLogic Server. The following options are available:
   - defaultLDAPRealmforOpenLDAPDirectoryServices
   - defaultLDAPRealmforNovellDirectoryServices
   - defaultLDAPRealmforMicrosoftSiteServer
   - defaultLDAPRealmforNetscapeDirectoryServer

   The configuration window for the chosen LDAP server appears.
3. Modify the following information in the Configuration Data box:
   - `server.host`—The host name of the LDAP server.
   - `server.port`—The port number on which the LDAP server listens.
   - `useSSL`—Specifies whether or not to use SSL to protect communications between the LDAP server and WebLogic Server. Set the value to `true` to enable the use of SSL.
   - `server.principal`—The LDAP user used by WebLogic Server to connect to the LDAP server.
   - `server.credential`—The password of the LDAP user used by WebLogic Server to connect to the LDAP server.
   - `user.dn`—The base DN of the tree in the LDAP directory that contains users.
   - `user.filter`—The LDAP search filter for finding a user given the name of the user.
   - `group.dn`—The base DN of the tree in the LDAP directory that contains groups.
   - `group.filter`—The LDAP search filter for finding a group given the name of the group.
   - `membership.filter`—The LDAP search filter for finding the members of a group given the name of the group.

   For more information, see “Supported LDAP Server Templates” on page 14-22.

Note: When using the LDAP v2 realm for Microsoft Site server, you must also specify `membership.search=true` and the following must be added to the `user.filter` value so that Microsoft Site server does not authenticate disabled users:

```
user.filter=(&(sAMAccountName=%u)(objectclassname=user)
(!userAccountControl:1.2.840.113556.1.4.803:=2))
```

4. To save your changes, click the Apply button.
5. Go to the Security node.
6. Choose the Filerealm tab.
7. Configure the Caching realm. For more information, see "Configuring the Caching Realm."

   Note: When you use an LDAP Security realm, you must configure and enable the Caching realm; otherwise, the LDAP Security realm will not work.

   When configuring the Caching realm, select the defaultLDAPRealmfor*LDAPserver* (for example, defaultLDAPRealmforOpenLDAPDirectoryServices) from the pull-down menu for the Basic attribute on the General tab. The Basic attribute defines the association between the Caching realm and the alternate security realm (in this case, the LDAP Realm).
8. Reboot WebLogic Server.

Supported LDAP Server Templates

Listing 14-1 through Listing 14-5 are templates used to configure LDAP servers supported by the LDAP V2 Realm.

Warning: Each line in the following code examples must appear on a single line. The code in the code examples has been formatted to fit the margins of this document and some lines have been broken to facilitate that formatting.

Listing 14-1 Default Netscape Directory Server Template

```
<CustomRealmName="defaultLDAPRealmForNetscapeDirectoryServer"
RealmClassName="weblogic.security.ldaprealmv2.LDAPRealm"
```

```
ConfigurationData=
"server.host=ldapserver.example.com;
server.port=700;
useSSL=true;
server.principal=uid=admin,
ou=Administrators,ou=TopologyManagement,o=NetscapeRoot;
server.credential=*secret*;
user.dn=ou=people,o=beasys.com;
user.filter=(&amp;(uid=%u)(objectclass=person));
group.dn=ou=groups,o=beasys.com;
group.filter=(&amp;(cn=%g)(objectclass=groupofuniquenames));
membership.filter=(&amp;(uniquemember=%M)
   (objectclass=groupofuniquenames));

"Notes="Before enabling the LDAP V2 security realm, edit the
configuration parameters for your environment."/>
```

Listing 14-2 Default Microsoft Site Server Template

```
<CustomRealmName="defaultLDAPRealmForMicrosoftSiteServer"
RealmClassName="weblogic.security.ldaprealmv2.LDAPRealm"
ConfigurationData=
"server.host=ldapserver.example.com;
server.port=700;
useSSL=true;
server.principal=cn=Administrator,ou=Members,
   o=ExampleMembershipDir;
server.credential=*secret*
user.dn=ou=Members, o=ExampleMembershipDir;
user.filter=(&amp;(cn=%u)(objectclass=member)
   (!userAccountControl:1.2.840.113556.1.4.803:=2)));
group.dn=ou=Groups, o=ExampleMembershipDir;
group.filter=(&amp;(cn=%g)(objectclass=mgroup));
membership.scope.depth=1;microsoft.membership.scope=sub;
membership.filter=(|(&amp;(memberobject=%M)
(objectclass=memberof))(&amp;(groupobject=%M)
(objectclass=groupmemberof)));
membership.search=true;

"Notes="Before enabling the LDAP V2 security realm, edit the
configuration parameters for your environment."/>
```

Listing 14-3 Default Novell Directory Services Template

```
<CustomRealmName="defaultLDAPRealmForNovellDirectoryServices"
RealmClassName="weblogic.security.ldaprealmv2.LDAPRealm"
ConfigurationData=
"server.host=ldapserver.example.com;
server.port=700;
useSSL=true;
server.principal=cn=Admin, DC=BEASYS
server.credential= *secret*;
user.dn=ou=people,o=example.com;
user.filter=(&amp;(cn=%u)(objectclass=person));
group.dn=ou=groups,o=example.com;
group.filter=(&amp;(cn=%g)(objectclass=groupofuniquenames));
membership.filter=(&amp;(member=%M)
   (objectclass=groupofuniquenames));"

"Notes="Before enabling the LDAP V2 security realm, edit the
configuration parameters for your environment."/>
```

Listing 14-4 Default Open LDAP Directory Services Template

```
<CustomRealmName="defaultLDAPRealmForOpenLDAPDirectoryServices"
RealmClassName="weblogic.security.ldaprealmv2.LDAPRealm"
ConfigurationData=
"server.host=ldapserver.example.com;
server.port=700;
useSSL=true;
server.principal=cn=Manager, dc=example, dc=com;
server.credential= *secret*;
user.dn=ou=people, dc=example,dc=com;
user.filter=(&amp;(uid=%u)(objectclass=person));
group.dn=ou=groups,dc=example,c=com;
group.filter=(&amp;(cn=%g)(objectclass=groupofuniquenames));
membership.filter=(&amp;(uniquemember=%M)
(objectclass=groupofuniquenames));"

"Notes="Before enabling the LDAP V2 security realm, edit the
configuration parameters for your environment."/>
```

Using Microsoft Active Directory with WebLogic Server

By default, WebLogic Server does not support Microsoft Active Directory LDAP server. To use Microsoft Active Directory with WebLogic Server, perform the following steps:

1. Go to the Security→Realms node in the left pane of the Administration Console.
2. Choose the defaultLDAPRealmforMicrosoftSiteServer option.

   The configuration window for the chosen LDAP server appears.
3. Modify the following information in the Configuration Data box with information specific to the Microsoft Active Directory LDAP server:
   - `server.host`—The host name of the LDAP server.
   - `server.port`—The port number on which the LDAP server listens.
   - `useSSL`—Specifies whether or not to use SSL to protect communications between the LDAP server and WebLogic Server. Set the value to `true` to enable the use of SSL.
   - `server.principal`—The LDAP user used by WebLogic Server to connect to the LDAP server.
   - `server.credential`—The password of the LDAP user used by WebLogic Server to connect to the LDAP server.
   - `user.dn`—The base DN of the tree in the LDAP directory that contains users.
   - `user.filter`—The LDAP search filter for finding a user given the name of the user.
   - `group.dn`—The base DN of the tree in the LDAP directory that contains groups.
   - `group.filter`—The LDAP search filter for finding a group given the name of the group.
   - `membership.filter`—The LDAP search filter for finding the members of a group given the name of the group.

   WebLogic Server authenticates by binding to the LDAP server and passing the DN and password of the user. Even if you have disabled a user account by setting the LDAP `userAccountControl` attribute to `ACCOUNTDISABLE`, the authentication will succeed unless you have modified the `user.filter` value to

ignore accounts that have been disable. Modify the `user.filter` value to only return accounts that do not have the `UF_ACCOUNTDISABLE` bit set. For example:

```
user.filter=(&(sAMAccountName=%u)(objectclassname=user)
(!userAccountControl:1.2.840.113556.1.4.803:=2))
```

When specifying the `group.filter` value, `CN` must be specified as `CN=%G` otherwise the filter fails to find the members of a group.

4. To save your changes, click the Apply button.
5. Go to the Security node.
6. Choose the Filerealm tab.
7. Configure the Caching realm. For more information, see "Configuring the Caching Realm."

   Note: When you use an LDAP Security realm, you must configure and enable the Caching realm; otherwise, the LDAP Security realm will not work.

   When configuring the Caching realm, select the defaultLDAPRealmfor*LDAPserver* (for example, defaultLDAPRealmforOpenLDAPDirectoryServices) from the pull-down menu for the Basic attribute on the General tab. The Basic attribute defines the association between the Caching realm and the alternate security realm (in this case, the LDAP Realm).
8. Reboot WebLogic Server.

Configuring the Windows NT Security Realm

The Windows NT Security realm uses account information defined for a Windows NT domain to authenticate Users and Groups. You can view Users and Groups in the Windows NT Security realm through the Administration Console, but you must manage Users and Groups through the facilities provided by Windows NT.

The Windows NT Security realm provides authentication (Users and Groups) but not authorization (ACLs). To update the ACL information in the `filerealm.properties` file that WebLogic Server uses, click the Refresh button on the General tab in the Security node after you change an ACL. If you use Groups with

your ACLs, you can reduce the frequency with which you must refresh the information in WebLogic Server. Changing the members of a Windows NT Group allows you to manage individual Users' access to WebLogic Server resources dynamically.

It is possible to use the Windows NT Security realm to authenticate against a Windows 2000 Active Directory primary domain controller. However, the authentication must be from a machine which is a member of the domain not the domain controller itself. There is no way to authenticate to the local User and Group store if the machine running the Windows NT Security realm is a member of another domain.

The Windows NT Security realm can be run on the primary domain controller, on a machine that is a member of a Windows NT domain, or on a machine that is a member of the Windows NT domain and wants to use a mutually trusted domain.

To use the Windows NT Security realm:

1. Go to the Security→Realms node in the left pane of the Administration Console.
2. In the right pane of the Administration Console, click the Configure a New NT Realm link.
3. Configuring the Windows NT Security realm involves setting attributes that define a name for the realm and the computer on which the Windows NT domain is running. To specify a realm name and computer, you must define values for the attributes shown the NT Realm Create window of the Administration Console.

   The following table describes the attributes you set in the NT Realm Configuration window.

Table 14-11 Windows NT Security Realm Attributes

Attribute	Description
Name	The name of the Windows NT Security realm, such as, AccountingRealm
Primary Domain	The host and port number of the computer where Users and Groups are defined for the Windows NT domain. If you enter multiple host and port numbers, use a comma delineated list.

4. To save your changes, click the Apply button.
5. When you have finished defining the attributes, reboot WebLogic Server.

6. Configure the Caching realm. For more information, see "Configuring the Caching Realm."

   Note: When you use an Windows NT Security realm, you must configure and enable the Caching realm; otherwise, the Windows NT Security realm will not work.

   When configuring the Caching realm, select your Windows NT Security realm from the pull-down menu for the Basic attribute on the General tab. The Basic attribute defines the association between the Caching realm and the alternate security realm (in this case, the Windows NT Security realm).

7. Go to the Security node.
8. Choose the Filerealm tab.
9. In the Caching Realm attribute, choose the name of the Caching Realm to be used with the Windows NT Security realm. A list of configured Caching Realms appears on the pull-down menu.

   Note: When you use an Windows NT Security realm, you must configure and enable the Caching realm; otherwise, the Windows NT Security realm will not work.

10. Reboot WebLogic Server.

Use the following command to verify that you have the correct privileges to run WebLogic Server as the specified Windows NT user:

`java weblogic.security.ntrealm.NTRealm` *`username password`*

where *`username`* and *`password`* are the username and password of the Windows NT account under which WebLogic Server runs.

The output from this command will tell if the specified username and password authenticated properly.

Command Output	Meaning
`auth?poppy`	The entered username and password authenticated correctly.
`auth?null`	The entered username and password did not authenticate properly.

If the test comes up with an immediate failure stating that the client or user running WebLogic Server does not have the privileges to run the Windows NT Security realm, you need to update the permissions (referred to as rights) for the Windows user running WebLogic Server.

To update the rights in Windows NT:

1. Go to Programs→Administrative Tools.
2. Select User Manager.
3. Under the Policies menu, choose the User Rights option.
4. Check the Show Advanced Users Rights option.
5. Give the following rights to the Windows user running WebLogic Server:
   - Act as part of the operating system
   - Create a token object
   - Replace a process level token
6. Verify that the Windows user running WebLogic Server is a member of the Administrators group.
7. Reboot Windows NT to ensure all the modifications take effect.
8. Verify that the Logon as System Account option is checked. Note that the Allow System to Interact with Desktop option does not need to be checked. Running the Windows NT Security realm under a specific Windows NT user account does not work.

To update the rights in Windows 2000:

1. Go to Programs→Administrative Tools.
2. Select Local Security Policy.
3. Go to Local Policies→User Rights Assignment.
4. Give the following rights to the Windows user running WebLogic Server:
   - Act as part of the operating system
   - Create a token object
   - Replace a process level token

5. Verify that the Windows user running WebLogic Server is a member of the Administrators group.
6. Reboot Windows 2000 to ensure all the modifications take effect.
7. Verify that the Logon as System Account option is checked. Note that the Allow System to Interact with Desktop option does not need to be checked. Running the Windows NT Security realm under a specific Windows NT user account does not work.

The following are common Windows NT error codes that occur when using the Windows NT Security realm:

Error Code	Meaning
1326	The host machine running the security realm does not have a trust relationship with the primary domain controller. The host machine may not be a member of the domain or the domain may not trust the host machine.
53	A network error has occurred indicating that the path to the primary domain controller could not be located. This error could occur if the domain name is misspelled or if the domain name is specified rather than the host name of the primary domain controller.

A full explanation of the Windows NT error codes is found in the `winerror.h` file.

Configuring the UNIX Security Realm

Note: The UNIX Security realm runs only on the Solaris platform.

The UNIX Security realm executes a small native program, `wlauth`, to look up Users and Groups and to authenticate Users on the basis of their UNIX login names and passwords. The `wlauth` program uses PAM (Pluggable Authentication Modules) which allows you to configure authentication services in the operating system without altering applications that use the service.

In UNIX, a user is defined as a member of a group in the following ways:

- The user is defined in a default group in `etc/passwd`.
- The user ID for a user is included in the `etc/group` entry for a specific group. The UNIX Security realm supports only this method of determining the members of a group.

After you change an ACL, click the Refresh button on the General tab in the Security to update the information in the `filerealm.properties` file that WebLogic Server uses. If you use Groups with your ACLs, you can reduce the frequency with which you must refresh the information in WebLogic Server. Changing the members of a UNIX Group allows you to manage individual Users' access to WebLogic Server resources dynamically.

It is possible to run `wlauth` to verify authentication. At a UNIX command prompt:

1. Enter `wlauth`.
2. Enter `-user_auth` *`username`*, *`password`*.

If the command returns a 0, the authentication check was successful. If the command returns a 1, the authentication check failed.

To use the UNIX Security realm:

1. Go to the Security→Realms node in the left pane of the Administration Console.
2. In the right pane of the Administration Console, click the Configure a New UNIX Realm link.
3. Configuring the UNIX Security realm involves setting attributes that define a name for the realm and the program that provides authentication services for the UNIX Security realm. To define these names, specify values for the attributes on the UNIX Realm Create window of the Administration Console.

   The following table describes the attributes you set in the UNIX Realm Create window.

Table 14-12 UNIX Security Realm Attributes

Attribute	Description
Name	The name of the UNIX Security realm, such as AccountingRealm

Table 14-12 UNIX Security Realm Attributes

Attribute	Description
AuthProgram	The name of the program used to authenticate users in the UNIX security realm. In most cases, the name of the program is `wlauth`.
Realm Classname	The name of the Java class that implements the UNIX Security realm. The Java class needs to be in the class path of WebLogic Server.

4. To save your changes, click the Apply button.
5. When you have finished defining the attributes, reboot WebLogic Server.
6. Configure the Caching realm. For more information, see "Configuring the Caching Realm."

   Note: When you use an UNIX Security realm, you must configure and enable the Caching realm; otherwise, the UNIX Security realm will not work.

   When configuring the Caching realm, select your UNIX Security realm from the pull-down menu for the Basic attribute on the General tab. The Basic attribute defines the association between the Caching realm and the alternate security realm (in this case, the UNIX Security realm).
7. Go to the Security node.
8. Choose the Filerealm tab.
9. In the Caching Realm attribute, choose the name of the Caching Realm to be used with the UNIX Security realm. A list of configured Caching Realms appears on the pull-down menu.

   Note: When you use an UNIX Security realm, you must configure and enable the Caching realm; otherwise, the UNIX Security realm will not work.
10. Reboot WebLogic Server.

If `wlauth` is not in the WebLogic Server class path or if you have given the program a name other than `wlauth`, you must add a Java command-line property when you start WebLogic Server. Edit the script you use to start WebLogic Server and add the following option after the `java` command:

```
-Dweblogic.security.unixrealm.authProgram=wlauth_prog
```

Replace *wlauth_prog* with the name of the `wlauth` program, including the full path if the program is not in the search path. Start WebLogic Server. If the `wlauth` program is in the WebLogic Server path and is named `wlauth`, this step is not needed.

Configuring the RDBMS Security Realm

The RDBMS Security realm is a BEA-provided custom security realm that stores Users, Groups and ACLs in a relational database.The RDBMS Security realm is an example and is not ment to be used in a production environment. You can perform the following managment functions on the RDBMS Security realm through the Administration Console:

Managment Function	**Support through the Administration Console**
Create User	Yes but only in memory.
Delete User	Yes
Change Password	No
Create Group	No
Delete Group	Yes
Add Group Member	Yes
Delete Group Member	Yes
Create ACL	No
Delete ACL	No
Add Permission	No
Delete Permission	No

SQL scripts that populate a database are used to create Groups for the RDBMS Security realm

The RDBMS Security realm can be used as a starting point for creating a production security realm. You can extend the RDBMS Security realm by using the following interfaces in the `weblogic.security.acl` package to add management capabilities to the RDBMS Security realm:

- `ManageableRealm`—Create Groups, create and delete ACLs, and perform lookups of Users, Groups, and ACLs.
- `User`—Change the password.
- `ACL`—Add and remove permissions for Users and Groups.

If you extend the RDBMS Security realm with any of these interfaces, you may also need to update the database schema.

Note: The RDBMS example does not work with databases that have an autocommit feature enabled. If you use the RDBMS example as a starting point for your RDBMS implementation, use explicit commit statements in your code and make sure the autocommit feature in the database you are using is disabled.

To use the RDBMS Security realm:

1. Go to the Security→Realms node in the left pane of the Administration Console.
2. In the right pane of the Administration Console, click the Configure a New RDBMS Realm link.
3. Define information about the class that implements the RDBMS Security realm.

   The following table describes the attributes you set on the General tab.

Table 14-13 RDBMS Security Realm Attributes on the General Tab

Attribute	Description
Name	Name of the RDBMS Security realm, such as, AccountingRealm
Realm Class	Name of the WebLogic class that implements the RDBMS Security realm. The Java class needs to be in the CLASSPATH of WebLogic Server.

4. To save your changes, click the Apply button.

5. Define attributes for the JDBC driver being used to connect to the database.

   The following table describes the attributes you set on the Database tab.

Table 14-14 RDBMS Security Realm Attributes on the Database Tab

Attribute	Description
Driver	Full class name of the JDBC driver. This class name must be in the CLASSPATH of WebLogic Server.
URL	URL for the database you are using with the RDBMS realm, as specified by your JDBC driver documentation.
User Name	Default user name for the database.
Password	Password for the default user of the database.

6. To save your changes, click the Apply button.

7. Define the schema used to store Users, Groups, and ACLs in the database in the Schema Properties box on the Schema tab.

   Listing 14-5 contains the database statements entered in the Schema properties for the RDBMS code example shipped with WebLogic Server in the `/samples/examples/security/rdbmsrealm` directory.

Listing 14-5 Sample Schema for RDBMS Security Realm

```
“getGroupNewStatement=true;getUser=SELECT U_NAME, U_PASSWORD FROM
users WHERE U_NAME = ?;
getGroupMembers=SELECT GM_GROUP, GM_MEMBER from groupmembers WHERE
GM_GROUP = ?;
getAclEntries=SELECT A_NAME, A_PRINCIPAL, A_PERMISSION FROM
aclentries WHERE A_NAME = ? ORDER BY A_PRINCIPAL;
getUsers=SELECT U_NAME, U_PASSWORD FROM users;
getGroups=SELECT GM_GROUP, GM_MEMBER FROM groupmembers;
getAcls=SELECT A_NAME, A_PRINCIPAL, A_PERMISSION FROM aclentries
ORDER BY A_NAME, A_PRINCIPAL;
getPermissions=SELECT DISTINCT A_PERMISSION FROM aclentries;
getPermission=SELECT DISTINCT A_PERMISSION FROM aclentries WHERE
A_PERMISSION = ?;
```

```
newUser=INSERT INTO users VALUES ( ? , ? );
addGroupMember=INSERT INTO groupmembers VALUES ( ? , ? );
removeGroupMember=DELETE FROM groupmembers WHERE GM_GROUP = ? AND
GM_MEMBER = ?;
deleteUser1=DELETE FROM users WHERE U_NAME = ?;
deleteUser2=DELETE FROM groupmembers WHERE GM_MEMBER = ?;
deleteUser3=DELETE FROM aclentries WHERE A_PRINCIPAL = ?;
deleteGroup1=DELETE FROM groupmembers WHERE GM_GROUP = ?;
deleteGroup2=DELETE FROM aclentries WHERE A_PRINCIPAL = ?"
```

8. To save your changes, click the Apply button.
9. When you have finished defining the attributes, reboot WebLogic Server.
10. Configure the Caching realm. For more information, see "Configuring the Caching Realm."

    Note: When you use an RDBMS Security realm, you must configure and enable the Caching realm; otherwise, the RDBMS Security realm will not work.

    When configuring the Caching realm, select the RDBMS Security realm from the pull-down menu for the Basic attribute on the General tab. The Basic attribute defines the association between the Caching realm and the alternate security realm (in this case, the RDBMS Security realm).
11. Go to the Security node.
12. Choose the Filerealm tab.
13. In the Caching Realm attribute, choose the name of the Caching Realm to be used with the RDBMS Security realm. A list of configured Caching Realms appears on the pull-down menu.

    Note: When you use an RDBMS Security realm, you must configure and enable the Caching realm; otherwise, the RDBMS Security realm will not work.
14. Reboot WebLogic Server.

Installing a Custom Security Realm

You can create a custom security realm that draws from an existing store of Users such as directory server on the network. To use a custom security realm, you create an implementation of the `weblogic.security.acl.AbstractListableRealm` interface or the `weblogic.security.acl.AbstractManageableRealm` interface and then use the Administration Console to install your implementation.

To install a custom security realm:

1. Go to the Security→Realms node in the left pane of the Administration Console.
2. In the right pane of the Administration Console, click the Configure a New Custom Realm link.
3. Define a name for the custom security realm, specify the interface that implements the realm, and define how the Users, Groups, and optionally ACLs are stored in the custom security realm on the Configuration window.

   The following table describes the attributes you set on the Custom Security Realm Configuration window.

Table 14-15 Custom Security Realm Attributes

Attribute	Description
Name	Name of the Custom Security realm, such as AccountingRealm
Realm Class Name	Name of the WebLogic class that implements the Custom Security realm. The Java class needs to be in the CLASSPATH of WebLogic Server.
Configuration Data	Information needed to connect to the security store.
Password	Password for the Custom Security realm. If a password is supplied, WebLogic Server encrypts it.

4. To save your changes, click the Create button.
5. When you have finished defining the attributes, reboot WebLogic Server.

6. Configure the Caching realm. For more information, see "Configuring the Caching Realm."

   Note: When you use an custom security realm, you must configure and enable the Caching realm; otherwise, the custom security realm will not work.

   When configuring the Caching realm, select the Custom Security realm from the pull-down menu for the Basic attribute on the General tab. The Basic attribute defines the association between the Caching realm and the custom security realm.

7. Go to the Security node.
8. Choose the Filerealm tab.
9. In the Caching Realm attribute, choose the name of the Caching Realm to be used with the custom security realm. A list of configured Caching Realms appears on the pull-down menu.

   Note: When you use an custom security realm, you must configure and enable the Caching realm; otherwise, the custom security realm will not work.

10. Reboot WebLogic Server.

For information about writing a custom security realm, see Writing a Custom Security Realm.

Migrating Security Realms

WebLogic Server provides a management architecture for security realms. The management architecture implemented through MBeans allows you to manage security realms through the Administration Console. If you have a security realm from a previous release of WebLogic Server, use the following information to migrate to the new architecture:

- If you are using the Windows NT, UNIX, or LDAP security realms, use the Convert weblogic.properties option in the Administration Console to convert the security realm to the new architecture. Note that you can view Users, Groups, and ACLs in a Windows NT, UNIX, or LDAP security realm in the Administration Console. However, you still need to use the tools in the Windows NT, UNIX, or LDAP environments to manage Users and Groups.

- If you are using a custom security realm, follow the steps in "Installing a Custom Security Realm" to specify information about how the Users, Groups, and optionally ACLs are stored in your custom security realm.
- The Delegating security realm is no longer supported. If you are using the Delegating security realm, you will have to use another type of security realm to store Users, Groups, and ACLs.
- If you are using the RDBMS security realm, use one of the following options to convert the security realm:
  - If you did not change the source for the RDBMS security realm, follow the steps in "Configuring the RDBMS Security Realm" to instantiate a new class for your existing RDBMS security realm and define information about the JDBC driver being used to connect to the database and the schema used by the security realm. In this case, you are creating a MBean in WebLogic Server for the RDBMS security realm.
  - If you customized the RDBMS security realm, convert your source to use the MBeans. Use the code example in the `\samples\examples\security\rdbmsrealm` directory as a guide to converting your RDBMS security realm. Once you have converted your RDBMS security realm to MBeans, follow the instructions in"Configuring the RDBMS Security Realm" to define information about the JDBC driver being used to connect to the database and the schema used by the security realm.

Defining Users

Note: This section explains how to add Users to the File realm. If you are using an alternate security realm, you must use the administration tools provided in that realm to define a User.

User and group names must be unique. You can use multibyte characters and all special characters except a comma (`,`) in user and group names.

Users are entities that can be authenticated in a WebLogic Server security realm. A User can be a person or a software entity, such as a Java client. Each User is given a unique identity within a WebLogic Server security realm. As a system administrator you must guarantee that no two Users in the same security realm are identical.

Defining Users in a security realm involves specifying a unique name and password for each User that will access resources in the WebLogic Server security realm in the Users window of the Administration Console.

WebLogic Server has two special users, `system` and `guest`:

- The `system` User is the administrative user who controls system-level WebLogic Server operations, such as starting and stopping servers, and locking and unlocking resources. The `system` User and its password are defined during the WebLogic Server installation procedure. As a security precaution, BEA recommends changing the password for the `system` User. For more information, see "Changing the System Password."
- The `guest` User is automatically provided by WebLogic Server. When authorization is not required, WebLogic Server assigns the `guest` identity to a client, thus giving the client access to any resources that are available to the `guest` user. A client can log in as the `guest` User by entering `guest` as the username and `guest` as the password when prompted by a Web browser or by supplying the `guest` username and password in a Java client. By default, the `guest` account is enabled.

  For a more secure deployment, BEA recommends running WebLogic Server with the `guest` account disabled. To disable the `guest` account, select the Guest Disabled attribute on the General tab of the Security Configuration window. Disabling the `guest` account just disables the ability to log in into the account `guest`; it does not disable the ability for unauthenticated users to access a WebLogic Server deployment.

Warning: Be advised it is still possible to access a deployment through an anonymous user if the ACLs on the anonymous user are not set properly. Set ACLs so that unauthorized access is not possible.

The `system` and `guest` Users are like other Users in a WebLogic Server security realm:

- To access WebLogic Server resources, they must have appropriate ACLs.

- To execute an operation on a WebLogic Server resource, they must provide a username and password (or digital certificate).

To define a User:

1. Go to the Security→Users node in the left pane of the Administration Console.

   The User Configuration window appears.

2. In the User Configuration window, enter the name of the User in the Name attribute.
3. Enter the a password for the User in the Password attribute.
4. Enter the password again in the Confirm Password attribute.
5. Click Create.

To delete a User:

1. Enter the name of the User in the Delete Users box on the User Configuration window.
2. Click Delete.

To change the password of a User:

1. Enter the name of the User in the Name attribute on the User Configuration window.
2. Enter the old password in the Old Password attribute.
3. Enter the new password in the New Password attribute.
4. Enter the new password again to confirm the password change.

While using WebLogic Server, you may have Users that are locked. Perform the following steps to unlock a User:

1. Open the Users window in the Administration Console.
2. Click on the Unlock Users link.
3. Enter the names of the Users you want to unlock in the Users to Unlock field.
4. Choose the servers on which you want the Users unlocked.
5. Click Unlock.

For more information about Users and the access control model in WebLogic Server, see Introduction to WebLogic Security and Security Fundamentals.

Defining Groups

Note: This section describes how to add Groups to the File realm. If you are using an alternate security realm, you need to use the management tools provided in that realm to define a Group.

User and group names must be unique. You can use multibyte characters and all special characters except a comma (,) in user and group names.

A Group represents a set of Users who usually have something in common, such as working in the same department in a company. Groups are a means of managing a number of Users in an efficient manner. When a Group is granted a permission in an ACL, all members of the Group effectively receive that permission. BEA recommends assigning permissions to Groups rather than to individual Users.

By default, WebLogic Server has the following Groups:

- All Users defined in the security realm are automatically members of the `everyone` Group.
- All Users defined in the security realm except the `guest` user are automatically members of the `users` Group.
- The `system` User is a member of the `Administrators` Group. This Group should be given the permissions appropriate for a user responsible for starting and stopping servers and maintaining a running WebLogic Server deployment. Access to this group should be limited.

You can register a Group with the WebLogic Server security realm by performing the following steps:

1. Go to the Security→Groups node in the left pane of the Administration Console.
2. Click the Create a New Group link.

   The Group Configuration window appears.

3. Enter the name of the Group in the Name attribute on the Group Configuration window. BEA recommends naming Groups in the plural. For example, Administrators instead of Administrator.
4. Click on the Users attribute and select the WebLogic Server Users you want to add to the Group.
5. Click on the Groups attribute and select the WebLogic Server Groups you want to add to the Group.
6. Click on the Apply button to create a new Group.

To delete Groups, enter the name of the Group in the Remove These Groups list box on the Group Configuration window and click Remove.

For more information about Groups and the access control model in WebLogic Server, see Introduction to WebLogic Security and Security Fundamentals.

Defining ACLs

Users access resources in a WebLogic Server security realm. Whether or not a User can access a resource is determined by the access control lists ACLs for that resource. An ACL defines the permissions by which a User can interact with the resource. To define ACLs, you create an ACL for a resource, specify the permission for the resource and then grant the permission to a specified set of Users and Groups. BEA recommends assigning ACLs to Groups.

Each WebLogic Server resource has one or more permissions that can be granted. The following table summarizes the functions for various WebLogic Server resources for which permissions can be restricted with an ACL.

Table 14-16 ACLs for WebLogic Server Resources

For this WebLogic Server resource...	This ACL...	Grants Permission for these functions...
WebLogic Servers	`weblogic.server` `weblogic.server.`*`servername`*	boot
Command-line Administration Tools	`weblogic.admin` **Note:** To add ACLs through the Administration Console, you need to define `weblogic.admin.acl.modify`.	shutdown, lockServer unlockServer, modify
MBeans	`weblogic.admin.mbean.`*`mbeaninstancename`* `weblogic.admin.mbean.`*`mbeantypename`*	read, write, access
WebLogic Events	`weblogic.event.`*`topicName`*	send receive
WebLogic JDBC connection pools	`weblogic.jdbc.connectionPool.`*`poolname`* **Note:** If you specify a value in the ACL field when configuring a JDBC connection pool, you must define the same value on the Security→ACLs node in order for the ACL to work. If you chose to use the `weblogic.jdbc.connectionPool.`*`poolname`* syntax to create an ACL for a JDBC connection pool, leave the ACL field in the JDBC connection pool configuration place blank.	reserve reset admin
WebLogic Passwords	`weblogic.passwordpolicy`	unlockuser
WebLogic JMS destinations	`weblogic.jms.topic.`*`topicName`* `weblogic.jms.queue.`*`queueName`*	send, receive
WebLogic JNDI contexts	`weblogic.jndi.`*`path`*	lookup modify list

Note: When you specify an ACL for a JDBC connection pool, you must specifically define access to the JDBC connection pool for the `system` and `guest` user in the `filerealm.properties` file. For example:

```
acl.reserve.poolforsecurity=system, guest
acl.reset.poolforsecurity=system, guest
```

To create ACLs for a WebLogic Server resource, open the Administration Console and perform the following steps:

1. Go to the Security→ACLs node in the left pane of the Administration Console.
2. In the right pane of the Administration Console, click the Create a New ACL link.

   The ACL Configuration window appears.
3. Specify the name of WebLogic Server resource that you want to protect with an ACL in the New ACL Name field.

   For example, create an ACL for a JDBC connection pool named `demopool`.
4. Click Create.
5. Click on the Add a New Permission link.
6. Specify a permission for the resource.

   You can either create separate ACLs for each permission available for a resource or one ACL that grants all the permissions for a resource. For example, you can create three ACLs for the JDBC connection pool, `demopool`: one with `reserve` permission, one with `reset` permission, and one with `shrink` permission. Or you can create one ACL with `reserve`, `reset`, and `shrink` permissions.
7. Specify Users or Groups that have the specified permission to the resource.
8. Click Apply.

When creating ACLs for resources in WebLogic Server you need to use the syntax in Table 14-16 to refer to the resource.

If you modify an existing ACL, click the Refresh button on the General tab in the Security node to update the information in the `filerealm.properties` file that WebLogic Server uses.

Before you can boot WebLogic Server, you need to give boot permission for the server to a specific Group. This security measure prevents unauthorized users from booting WebLogic Server.

By default, only the system user can modify MBeans. BEA recommends limiting the number of users that can access and modify MBeans. Use the following ACL to access to all the WebLogic Server MBeans:

```
access.weblogic.admin.mbean=Group or User name
```

If a user or group fails in an attempt to access a MBean, a `weblogic.management.NoAccessRuntimeException` is returned. The server log contains the details indicating which user attempted to access what MBean.

Before you can grant permissions to Users or Groups through the Administration Console, you need to give the Administrators Group the following permission:

```
acl.modify.weblogic.admin=Administrators
```

Configuring the SSL Protocol

The following sections describe how to obtain digital certificates and configure the SSL protocol:

- Obtaining a Private Key and Digital Certificate
- Storing Private Keys and Digital Certificates
- Defining Trusted Certificate Authorities
- Defining Attributes for the SSL Protocol
- Modifying Parameters for SSL Session Caching

For a complete description of the SSL Protocol, see Introduction to WebLogic Security and Security Fundamentals.

This release of WebLogic Server supports SSL v3.0.

Obtaining a Private Key and Digital Certificate

You need a private key and digital certificate for each deployment of WebLogic Server that will use the SSL protocol. To acquire a digital certificate from a certificate authority, you must submit your request in a particular format called a Certificate Signature Request (CSR). WebLogic Server includes a Certificate Request Generator servlet that creates a CSR. The Certificate Request Generator servlet collects information from you and generates a private key file and a certificate request file. You can then submit the CSR to a certificate authority such as VeriSign or Entrust.net. Before you can use the Certificate Request Generator servlet, WebLogic Server must be installed and running.

Note: If you obtain a private key file from a source other than the Certificate Request Generator servlet, verify that the private key file is in PKCS#5/PKCS#8 PEM format.

To generate a CSR, perform the following steps:

1. Start the Certificate Request Generator servlet. The `.war` file for the servlet is located in the `\wlserver6.1\config\\applications` directory. The `.war` file is automatically installed when you start WebLogic Server.

2. In a Web browser, enter the URL for the Certificate Request Generator servlet as follows:

   `https://`*`hostname:port`*`/certificate/`

   The components of this URL are defined as follows:

   - *`hostname`* is the DNS name of the machine running WebLogic Server.
   - *`port`* is the number of the port at which WebLogic Server listens for SSL connections. The default is 7002.

     For example, if WebLogic Server is running on a machine named `ogre` and it is configured to listen for SSL communications at the default port `7002` to run the Certificate Request Generator servlet, you must enter the following URL in your Web browser:

     ```
     https://ogre:7002/certificate/
     ```

3. The Certificate Request Generator servlet loads a form in your Web browser. Complete the form displayed in your browser, using the information in the following table:

Table 14-17 Fields on the Certificate Request Generator Form

Field	Description
Country code	Two-letter ISO code for your country. The code for the United States is US.
Organizational unit name	Name of your division, department, or other operational unit of your organization.
Organization name	Name of your organization. The certificate authority may require any host names entered in this attribute belong to a domain registered to this organization.
E-mail address	E-mail address of the administrator. The digital certificate is mailed to this e-mail address.
Full host name	Fully qualified name of the WebLogic Server on which the digital certificate will be installed. This name is the one used for DNS lookups of the WebLogic Server, for example, `node.com`. Web browsers compare the host name in the URL to the name in the digital certificate. If you change the host name later, you must request a new digital certificate.
Locality name (city)	Name of your city or town. If you operate with a license granted by a city, this attribute is required; you must enter the name of the city that granted your license.
State name	Name of the state or province in which your organization operates if your organization is in the United States or Canada, respectively. Do not abbreviate.
Private Key Password	The password used to encrypt the private key. Enter a password in this field if you want to use a protected key with WebLogic Server. If you choose to use a protected key, you are prompted for the password whenever the key is used. If you specify a password, you get a PKCS-8 encrypted private key. BEA recommends using a password to protect private keys. If you do not want to use a protected key, leave this field blank. To use protected private keys, enable the Key Encrypted attribute on the SSL tab of the Server window in the Administration Console.

Table 14-17 Fields on the Certificate Request Generator Form

Field	Description
Strength	The length (in bits) of the keys to be generated. The longer the key, the more difficult it is for someone to break the encryption. If you have the domestic version of WebLogic Server, you can choose 512-, 768-, or 1024-bit keys. The 1024-bit key is recommended. **Note:** This field only appears on the domestic version of the Certificate Request Generator servlet.

4. Click the Generate Request button.

   The Certificate Request Generator servlet displays messages informing you if any required attributes are empty or if any attributes contain invalid values. Click the Back button in your browser and correct any errors.

   When all attributes have been accepted, the Certificate Request Generator servlet generates the following files in the startup directory of your WebLogic Server:

   - `www__com-key.der`—The private key file. The name of this file should go into the Server Key File Name attribute field on the SSL tab in the Administration Console.
   - `www__com-request.dem`—The certificate request file, in binary format.
   - `www__com-request.pem`—The CSR file that you submit to the certificate authority. It contains the same data as the `.dem` file but is encoded in ASCII so that you can copy it into e-mail or paste it into a Web form.

5. Select a certificate authority and follow the instructions on that authority's Web site to purchase a digital certificate.

   - VeriSign, Inc. offers two options for WebLogic Server: Global Site Services, which features strong 128-bit encryption for domestic and export Web browsers, and Secure Site Services, which offers 128-bit encryption for domestic Web browsers and 40-bit encryption for export Web browsers.
   - Entrust.net digital certificates offer 128-bit encryption for domestic browser versions and 40-bit encryption for export browser versions.

6. When you are instructed to select a server type, choose `BEA WebLogic Server` to ensure that you receive a digital certificate that is compatible with WebLogic Server.
7. When you receive your digital certificate from the certificate authority, you need to store it in the `\wlserver6.1\config\` directory.
8. Configure WebLogic Server to use the SSL protocol, you need to enter the following information on the SSL tab in the Server Configuration window:
   - In the Server Certificate File Name attribute, enter the full directory location and name of the digital certificate that establishes the identity of WebLogic Server.
   - In the Trusted CA File Name attribute, enter the full directory location and name of the digital certificate for the certificate authority who signed the digital certificate of WebLogic Server.
   - In the Server Key File Name attribute, enter the full directory location and name of the private key file for WebLogic Server.

   For more information about configuring the SSL protocol, see Defining Attributes for the SSL Protocol.
9. If you are using a protected private key, use the following command-line option to start WebLogic Server.

   `-Dweblogic.management.pkpassword=`*`password`*

   where *`password`* is the password for the private key.

Storing Private Keys and Digital Certificates

Once you have a private key and digital certificate, copy the private key file generated by the Certificate Request Generator servlet and the digital certificate you received from the certificate authority into the `\wlserver6.1\config\` directory.

Private key files and digital certificates are generated in either PEM or Definite Encoding Rules (DER) format. The filename extension identifies the format of the digital certificate file.

A PEM (`.pem`) format private key file begins and ends with the following lines, respectively:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----

-----END ENCRYPTED PRIVATE KEY-----
```

A PEM(`.pem`) format digital certificate begins and ends with the following lines, respectively:

```
-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----
```

Note: Your digital certificate may be one of several digital certificates in the file, each of which is bounded by the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines. Typically, the digital certificate file for a WebLogic Server is in one file, with either a `.pem` or `.der` extension, and the WebLogic Server certificate chain is in another file. Two files are used because different WebLogic Servers may share the same certificate chain.

The first digital certificate in the certificate authority file is the first digital certificate in the WebLogic Server's certificate chain. The next certificates in the file are the next digital certificates in the certificate chain. The last certificate in the file is a self-signed digital certificate that ends the certificate chain.

A DER(`.der`) format file contains binary data. WebLogic Server requires that the file extension match the contents of the certificate file so be sure to save the file you receive from your certificate authority with the correct file extension.

Assign protections to the private key file and digital certificates so that only the `system` User of WebLogic Server has read privileges and all other users have no privileges to access the private key file or digital certificate. If you are creating a file with the digital certificates of multiple certificate authorities or a file that contains a certificate chain, you must use PEM format. WebLogic Server provides a tool to for converting DER-format files to PEM format, and visa versa. For more information, see WebLogic Utilities.

Defining Trusted Certificate Authorities

When establishing an SSL connection, WebLogic Server checks the identity of the certificate authority against a list of trusted certificate authorities to ensure the certificate authority currently being used is trusted.

Copy the root certificate of the certificate authority into the `\wlserver6.1\config\` directory of your WebLogic Server and set the attributes described in Defining Attributes for the SSL Protocol.

If you want to use a certificate chain, append the additional PEM-encoded digital certificates to the digital certificate of the certificate authority that issued the digital certificate for WebLogic Server. The last digital certificate in the file should be a digital certificate that is self-signed (that is, the rootCA certificate).

If you want to use mutual authentication, take the root certificates for the certificate authorities you want to accept and include them to the trusted CA file.

Defining Attributes for the SSL Protocol

The Secure Sockets Layer (SSL) protocol provides secure connections by allowing two applications connecting over a network connection to authenticate the other's identity and by encrypting the data exchanged between the applications. The SSL protocol provides server authentication and optionally client authentication, confidentiality, and data integrity.

To define attributes for the SSL protocol, perform the following steps:

1. Open the Administration Console.
2. Open the Server Configuration window.
3. Select the SSL tab. Define the attributes on this tab by entering values and selecting the required checkboxes. (For details, see the following table.)
4. Click the Apply button to save your changes.
5. Reboot WebLogic Server.

Note: If you are using a PKCS-8 protected private key, you need to specify the password for the private key on the command line when you start WebLogic Server.

The following table describes each attribute on the SSL tab of the Server Configuration window.

Table 14-18 SSL Protocol Attributes

Attribute	Description
Enabled	Enables the use of the SSL protocol. By default, this attribute is enabled.
Listen Port	Number of the dedicated port on which WebLogic Server listens for SSL connections. The default is 7002.
Server Key File Name	Directory location and name of the private key file for WebLogic Server. Start the directory location needs at the root of the WebLogic Server installation. For example: `\wlserver6.1\config\myapp\privatekey.pem.` The file extension (`.DER` or `.PEM`) indicates the method that WebLogic Server should use to read the contents of the file.
Server Certificate File Name	Full directory location and name of the digital certificate file that establishes the identity of WebLogic Server. Start the directory location at the root of the WebLogic Server installation. For example: `\wlserver6.1\config\myapp\cert.pem.` The file extension (`.DER` or `.PEM`) indicates the method that WebLogic Server should use to read the contents of the file.

Table 14-18 SSL Protocol Attributes

Attribute	Description
Server Certificate Chain File Name	Full directory location of the digital certificate used to sign the digital certificate for WebLogic Server. Start the directory location at the root of the WebLogic Server installation. For example: `\wlserver6.1\config\myapp\cacert.pem.` The file extension (`.DER` or `.PEM`) indicates the method that WebLogic Server should use to read the contents of the file. When using a certificate chain with WebLogic Server, the file should contain as its first member the digital certificate used to sign the digital certificate for WebLogic Server, the second member should contain a digital certificate used to sign the first digital certificate in the file and so on. The last digital certificate in the file should be self-signed. The Server Certificate Chain File Name attribute is required to have at least one digital certificate. If there is only one digital certificate in the file, the digital certificate must be self-signed (i.e., it must be a root CA digital certificate). When obtaining a digital certificate from a certificate authority, you should receive the digital certificate of the certificate authority and other immediate digital certificates from the certificate authority.
Client Certificate Enforced	Defines whether or not clients must present digital certificates from a trusted certificate authority to WebLogic Server.
Trusted CA File Name	Name of the file that contains the digital certificate for the certificate authorities trusted by WebLogic Server. The file specified in this attribute can contain a single digital certificate or multiple digital certificates for certificate authorities. The file extension (`.DER` or `.PEM`) tells WebLogic Server how to read the contents of the file.
CertAuthenticator	Name of the Java class that implements the CertAuthenticator interface. For more information about using the `weblogic.security.acl.CertAuthenticator` interface, see Mapping a Digital Certificate to a WebLogic User.

Table 14-18 SSL Protocol Attributes

Attribute	Description
Key Encrypted	Specifies that the private key for WebLogic Server is encrypted with a password. The default is false. If you specify this attribute, you need to use protected keys with WebLogic Server. Also, when you boot WebLogic Server, use the following command-line option to start WebLogic Server. `-Dweblogic.management.pkpassword=`*`password`* where *`password`* is the password for the private key.
Use Java	When selected, enables the use of native Java libraries. WebLogic Server provides a pure-Java implementation of the SSL protocol: native Java libraries enhance the performance for SSL operations on the Solaris, Windows NT, and IBM AIX platforms. By default, this attribute is not enabled.
Handler Enabled	Specifies whether or not WebLogic Server rejects SSL connections that fail client authentication for one of the following reasons: ■ The requested client digital certificate was not furnished. ■ The client did not submit a digital certificate. ■ The digital certificate from the client was not issued by a certificate authority specified by the `Trusted CA Filename` attribute. By default, the SSL Handler allows one WebLogic Server instance to make outgoing SSL connections to another WebLogic Server instance. For example, an EJB in a WebLogic Server may open an HTTPS stream on another Web server. With the `HandlerEnabled` attribute enabled, WebLogic Server acts as a client in an SSL connection. By default this attribute is enabled. Disable this attribute only if you want to provide your own implementation for outgoing SSL connections. **Note:** The SSL Handler has no effect on the ability of WebLogic Server to manage incoming SSL connections.

Table 14-18 SSL Protocol Attributes

Attribute	Description
Export Key Lifespan	Number of times WebLogic Server uses an exportable key between a domestic server and an exportable client before generating a new one. The more secure you want WebLogic Server to be, the fewer times the key should be used before a new one is generated. The default is to use it 500 times.
Login Timeout Millis	Number of milliseconds that WebLogic Server should wait for an SSL connection before timing out. SSL connections take longer to negotiate than regular connections. If clients are connecting over the Internet, raise the default number to accommodate additional network latency. The default value is 25,000 milliseconds.
Certificate Cache Size	Number of digital certificates that are tokenized and stored by WebLogic Server. The default is 3.
Ignore HostName Verification	Disables the default Host Name Verifier. The Host Name Verifier compares the Subject DN of a digital certificate to the host name of the server that initiated the SSL connection. Check this attribute if you do not want host name verification performed (for example, if you are using the demonstration digital certificates shipped with WebLogic Server). Disabling this attribute leaves WebLogic Server vulnerable to man-in-the-middle attacks. BEA does not recommend using the demonstration digital certificates or disabling host name verification in any type of production environment.
HostName Verifier	Name of the Java class that implements the Host Name Verifier interface. For more information about using the `weblogic.security.SSL.HostNameVerifier` interface, see Using a Custom HostName Verifier.

Note: In previous releases of WebLogic Server, it was possible to define digital certificates that were self-signed and not validated in the Server Certificate File Name attribute (or in the `weblogic.security.certificate.server` property). This was not a good security policy. Now WebLogic Server requires that both the Server Certificate File Name and the Server Certificate Chain File Name attributes be defined.

Using PKCS#7 Files

PKCS#7 files can be used with WebLogic Server. However, the certificate chain in the file must be separated into individual `pb7` format files, convert the `pb7` files to PEM format, and append the files into a single PEM file. Each PKCS#7 file has the following parts:

- Certificate information in plain text
- The digital certificate for the server
- The trusted CA certificate for the certificate authority that issued the digital certificate for the server

The digital certificate for the server and the trusted CA certificate need to be separated into pb7 files.

Before performing the steps in this section, verify that the file is PKCS#7 format by opening the file in a text editor and locating the following information:

```
"Base 64 encoded certificate with CA certificate chain in pkcs7
format"
```

To use the PKCS#7 file with WebLogic Server:

1. Open the PKCS#7 file in a text editor.
2. Copy the digital certificate for the server and the trusted CA certificate in the PKCS#7 file into separate p7b files (for example, *`servername`*`.p7b` and `CA.p7b`).
3. In Windows Explorer on Windows 2000, click one of the `p7b` files.

   A Certificates window appears.
4. In the left pane of the Certificates window, select the `p7b` file you want to convert.
5. Select the Certificates option.
6. The Certificate Export wizard appears.
7. Click Next.
8. Select the Base64 Encoded Cert option (exports PEM formats).

9. Click Next.
10. Enter a name for the converted digital certificate.
11. Click Finish.

    The resulting file is in PEM format.
12. Perform steps 3-11 for the other `pb7` file.
13. Open a text editor and include both the PEM files into a single PEM file. The order is important (include the files in the order of trust). The server digital certificate should be the first digital certificate in the file. The trusted CA certificate should be the next file.

    You cannot have blank lines between digital certificates.

Modifying Parameters for SSL Session Caching

As of WebLogic Server 6.1 Service Pack 2, the SSL code includes parameters for SSL session caching. Using a cached SSL session eliminates the need for the connection to go through the SSL handshake again. The connection can simply pick up where it left off. By using a cached SSL session, an application significantly reduces the time spent establishing SSL connections, greatly improving performance. To use a cached SSL session, both the client and the server must have the ability to do SSL session caching. All browsers have the ability to do SSL session-caching.

The server-session caching is saved in the TTL Cache. For more information about TTL Cache, see Configuring the Caching Realm. The client-side SSL session cache will only hold one SSL session on the thread of execution.

SSL session caching is turned on by default. You can modify the server-session caching default size and time-to-live by using the following command line flags:

```
-Dweblogic.security.SSL.sessionCache.size=211
-Dweblogic.security.SSL.sessionCache.ttl=600
```

Table 14-19 Parameters

parameters	min	max	default
`sessionCache.size`	1	65537	211
`sessionCache.ttl`	1	max Integer.MAX_VALUE	600

Configuring Mutual Authentication

When WebLogic Server is configured for mutual authentication, clients are required to present their digital certificates to WebLogic Server, which validates digital certificates against a list of trusted certificate authorities.

To configure your WebLogic Server for the SSL protocol and certificate authentication, complete the procedures in Configuring the SSL Protocol section.

Copy the root certificates for the certificate authorities to be used by WebLogic Server to the `\wlserver6.1\config\` directory. During mutual authentication, clients are required to present a digital certificate issued by one of these trusted certificate authorities.

To configure mutual authentication, select the Client Certificate Enforced option on the SSL tab in the Server Configuration window of the Administration Console. By default, this option is not enabled.

Configuring RMI over IIOP with SSL

You can use the SSL protocol to protect IIOP connections to RMI remote objects. The SSL protocol secures connections through authentication and encrypts the data exchanged between objects. To use the SSL protocol to protect RMI over IIOP connections, do the following:

1. Configure WebLogic Server to use the SSL protocol. For more information, see Defining Attributes for the SSL Protocol
2. Configure the client Object Request Broker (ORB) to use the SSL protocol. Refer to the product documentation for your client ORB for information about configuring the SSL protocol.
3. Use the `host2ior` utility to print the WebLogic Server IOR to the console. The `host2ior` utility prints two versions of the interoperable object reference (IOR), one for SSL connections and one for non-SSL connections. The header of the IOR specifies whether or not the IOR can be used for SSL connections.
4. Use the SSL IOR when obtaining the initial reference to the CosNaming service that accesses the WebLogic Server JNDI tree.

For more information about using RMI over IIOP, see Programming WebLogic RMI Over IIOP.

Protecting Passwords

It is important to protect the passwords that are used to access resources in WebLogic Server. In the past, usernames and passwords were stored in clear text in a WebLogic Server security realm. Now WebLogic Server hashes all passwords. When WebLogic Server receives a client request, the password presented by the client is hashed and WebLogic Server compares it to the already hashed password for matching.

Each `filerealm.properties` file has an associated `SerializedSystemIni.dat` file that is used to hash the passwords. During installation, the `SerializedSystemIni.dat` file is put in the `\wlserver6.1\config\` directory.

If for any reason the `SerializedSystemIni.dat` file is corrupted or destroyed, you must reconfigure WebLogic Server.

Take the following precautions:

- Make a backup copy of the `SerializedSystemIni.dat` file and put it in the same location as a copy of its associated `filerealm.properties` file.

- Set the permissions on the `SerializedSystemIni.dat` file protections such that the administrator of the WebLogic Server deployment has write and read privileges and no other users have any privileges.
- If you have a `weblogic.properties` file with passwords that you want to hash, use the `Convert weblogic.properties` option on the main window in the Administration Console to convert the `weblogic.properties` file to a `config.xml` file. Once the file is converted, all existing passwords are protected.

The `config.xml` file no longer has clear text passwords. In place of clear text passwords, the `config.xml` file has encrypted, hashed passwords. You cannot copy encrypted passwords from one domain to another. Instead, you can edit the `config.xml` file and replace the existing encrypted and hashed passwords with clear text passwords and then copy the file to the new domain. The Administration Console will encrypt and hash the passwords the next time it writes to the file.

Password guessing is a common type of security attack. In this type of attack, a hacker attempts to log in to a computer using various combinations of usernames and passwords. WebLogic Server has strengthened its protection against password guessing by providing a set of attributes designed to protect passwords.

To protect the passwords in your WebLogic Server deployment, you must perform the following steps:

1. Open the Administration Console.
2. Click on the Security node.
3. In the right pane of the Administration Console, click on the Passwords tab.
4. Define the desired attributes on this tab by entering values at the appropriate prompts and selecting the required checkboxes. (For details, see the following table).
5. Click the Apply button to save your choices.
6. Reboot WebLogic Server.

The following table describes each attribute on the Passwords tab.

Table 14-20 Password Protection Attributes

Attribute	Description
Minimum Password Length	Number of characters required in a password. Passwords must contain a minimum of 8 characters. The default is 8.
Lockout Enabled	Requests the locking of a user account after invalid attempts to log in to that account exceed the specified Lockout Threshold. By default, this attribute is enabled.
Lockout Threshold	Number of failed password entries for a user that can be tried to log in to a user account before that account is locked. Any subsequent attempts to access the account (even if the username/password combination is correct) raise a Security exception; the account remains locked until it is explicitly unlocked by the system administrator or another login attempt is made after the lockout duration period ends. Invalid login attempts must be made within a span defined by the `Lockout Reset Duration` attribute. The default is 5.
Lockout Duration	Number of minutes that a user's account remains inaccessible after being locked in response to several invalid login attempts within the amount of time specified by the `Lockout Reset Duration` attribute. In order to unlock a user account, you need to have the `unlockuser` permission for the `weblogic.passwordpolicy`. The default is 30 minutes.

Table 14-20 Password Protection Attributes

Attribute	Description
Lockout Reset Duration	Number of minutes within which invalid login attempts must occur in order for the user's account to be locked. An account is locked if the number of invalid login attempts defined in the `Lockout Threshold` attribute happens within the amount of time defined by this attribute. For example, if the value in this attribute is five minutes and three invalid login attempts are made within a six-minute interval, then the account is not locked. If five invalid login attempts are made within a five-minute period, however, then the account is locked. The default is 5 minutes.
Lockout Cache Size	Specifies the intended cache size of unused and invalid login attempts. The default is 5.

Installing an Audit Provider

WebLogic Server enables you to create an audit provider to receive and process notifications of security events such as authentication requests, failed or successful authorization attempts, and receipt of invalid digital certificates.

To use an audit provider, you create an implementation of the `weblogic.security.audit.AuditProvider` interface. Then use the Administration Console to install and activate your implementation.

To install an audit provider, enter the name of your implementation of the `AuditProvider` class in the Audit Provider Class attribute on the General tab under the Security node in the Administration Console. Reboot WebLogic Server.

For more information about writing an audit provider, see Auditing Security Events. For an example of creating a connection filter, see the `LogAuditProvider` example in the `\samples\examples\security` directory of the WebLogic Server installation.

Installing a Connection Filter

You can create connection filters that allow you to reject or accept client connections based on a client's origin and protocol. After the client connects, and before any work is performed on its behalf, WebLogic Server passes the client's IP number and port, protocol (HTTP, HTTPS, T3, T3S, or IIOP), and WebLogic Server port number to the connection filter. By examining this information, you can choose to allow the connection or throw a `FilterException` to terminate it.

To use a connection filter, you must first create an implementation of the `weblogic.security.net.ConnectionFilter` interface. Then use the Administration Console to install your implementation.

To install a connection filter, enter the name of your implementation of the `weblogic.security.net.ConnectionFilter` interface, in the Connection Filter attribute on the Advanced tab under Security in the Administration Console. Reboot WebLogic Server.

For information about writing a connection filter, see Filtering Network Connections. For an example of creating a connection filter, see the `SimpleConnectionFilter` example in the `\samples\examples\security` directory of the WebLogic Server installation.

Setting Up the Java Security Manager

When you run WebLogic Server under Java 2 (JDK 1.2 or 1.3), WebLogic Server can use the Java Security Manager in Java 2 to provide additional access control for WebLogic Server resources. The Java Virtual Machine (JVM) has security mechanisms built into it that you which manage through a security policy file. The Java Security Manager can enforce a set of permissions granted to CodeSource or SignedBy classes. The permissions allow certain classes running in that instance of the JVM to do or not do certain runtime operations. In many cases, where the threat model does not include malicious code being run on the JVM, the Java Security Manager is unnecessary. In cases such as when an Application Service Provider uses WebLogic Server and unknown classes are being run, the Java Security Manager is necessary.

Note: In pre-6.0 releases of WebLogic Server, you enabled the Java Security Manager by using the `-Dweblogic.security.manager` property when starting WebLogic Server. Please note the change in the property for WebLogic Server version 6.0 and greater.

To use the Java Security Manager with WebLogic Server, specify the `-Djava.security.manager` property when starting WebLogic Server.

The Java Security Manager uses a security policy file that defines permissions. The full pathname of security policy is specified in the `-Djava.security.policy` property when you start WebLogic Server. If you enable the Java Security Manager but do not specify a security policy file, the Java Security Manager uses the default security policies defined in the `java.security` and `java.policy` files in the `$JAVA_HOME/lib/security` directory.

WebLogic Server includes an example security policy file named `weblogic.policy`. This file contains a set of default permissions.

To use the Java Security Manager security policy file with your WebLogic Server deployment:

1. Edit the following lines in the `weblogic.policy` file, replacing the specified location with the location of your WebLogic Server installation:

```
grant codebase "file://BEA/-"{
  permission java.io.FilePermission "D:${/}BEA${/}=", ...
```

Note: This change assumes your installation directory structure is the same as the one described in the *BEA WebLogic Server Installation Guide.*

2. If you want to run the Administration Console, add the following grant block and permissions to the `weblogic.policy` file:

```
grant {
    permission java.io.FilePermission
"D:{/}BEA${/}wlserver6.1${/}weblogic${/}management${/}console${
/}-", "read";
    permission java.io.FilePermission
"D:{/}BEA${/}wlserver6.1${/}config${/}${/}applications${/}.wl_t
emp_do_not_delete${/}weblogic${/}management${/}console${/}-",
"read";
   permission java.util.PropertyPermission  "user.*",  "read";
   };
```

3. If you have extra directories in your `CLASSPATH` or if you are deploying applications in extra directories, add specific permissions for those directories to your `weblogic.policy` file.
4. BEA recommends taking the following precautions:
   - Make a backup copy of the `weblogic.policy` file and put the backup copy in a secure location.
   - Set the permissions on the `weblogic.policy` file such that the administrator of the WebLogic Server deployment has write and read privileges and no other users.
5. To use the Java Security Manager and the `weblogic.policy` file with your WebLogic Server deployment, use the following properties when starting WebLogic Server:

   ```
   $java... -Djava.security.manager\
   -Djava.security.policy==D:/BEA/wlserver6.1/lib/weblogic.policy
   ```

Caution: The Java security manager is partially disabled during the booting of Administration and Managed Servers. During the boot sequence, the current Java security manager is disabled and replaced with a variation of the Java security manager that has the `checkRead()` method disabled. While disabling this method greatly improves the performance of the boot sequence, it also minimally diminishes security. The startup classes for WebLogic Server are run with this partially disabled Java security manager and therefore the classes need to be carefully scrutinized for security considerations involving the reading of files.

For more information about the Java Security Manager, see the Javadoc shipped with Java 2.

Modifying the weblogic.policy File for Third Party or User-Written Classes

The best location for your server-side user code is the `weblogic/myserver/serverclasses` directory. If you have third party or user-written classes that are not in that directory, perform the following steps to protect them:

1. Copy the entire block of code in the `weblogic.policy` file from "`grant codeBase...`" to the closing bracket and semicolon.
2. Paste the selection back into the `weblogic.policy` file below the section you just copied.
3. Edit the `grant codeBase` and the `permission.java.io.FilePermission` statements so that the directories point to the location of your third party or user-written code.

This procedure creates a security policy for your code that contains exactly the same permissions as those for WebLogic Server. You should examine these permissions closely to make sure that this is the security policy you want for those directories.

Caution: JavaSoft JDK version 1.2.1 on UNIX systems applies security policies improperly if your WebLogic Server software is not installed in the root directory of the file system or disk drive. Policy is only applied correct if the path in a `grant codeBase` URL has just one component. For example, if you install WebLogic Server in `c:\test\weblogic` (or even `/home/weblogic` on Solaris), you will see `AccessControlException` even though you use the correct URL in your security policy file.

To workaround this limitation, you can either install WebLogic in the root directory (recommended) or modify the URL so that it contains only the first component of the path to your WebLogic installation. For example:

```
grant codeBase "file:/c:/test/" {
```

Problems occur when using the "`/-`" in the specified URL. This problem has been acknowledged by Sun Microsystems as bug #4261298, but they have determined that this is not a bug in the JDK. They state, "when a path

is trailed with "`/-`" it means that the element preceding it is a directory and that grant functions for all elements below it. It does not mean that you can read the directory itself." The workaround for this nuance is to add an additional `FilePermission` entry that consists of just the directory itself (with no trailing "`/-`').

Using the Recording Security Manager Utility

The Recording Security Manager utility can be used to detect permission problems that occur when starting and running WebLogic Server. The utility outputs permissions that can be added to your security policy file to resolve the permission problems that the utility finds. The Recording Security Manager is available at the BEA Developer's Center.

Configuring Security Context Propagation

Security context propagation enables Java applications running in a WebLogic Server environment to access objects and operations in BEA Tuxedo domains. The BEA WebLogic Enterprise Connectivity component of WebLogic Server provides the security context propagation capability.

With security context propagation, the security identity of a User defined in a WebLogic Server security realm is propagated as part of the service context of an Internet Inter-ORB Protocol (IIOP) request sent to the BEA Tuxedo domain over a network connection that is part of a WLEC connection pool. Each network connection in the WLEC connection pool has been authenticated using a defined User identity.

To use security context propagation, create a WLEC connection pool for each BEA Tuxedo domain you want to access from WebLogic Server. WebLogic Server populates each WLEC connection pool with IIOP connections. Java applications in a

WebLogic Server environment obtain IIOP connections from a WLEC connection pool and use those connections to call objects and invoke operations in BEA Tuxedo domains.

Before using security context propagation, add *TUXDIR*/lib/wleorb.jar and *TUXDIR*/lib/wlepool.jar to the CLASSPATH variable in the startAdminWebLogic.sh or startAdminWebLogic.cmd file.

For more information, see *Using WebLogic Enterprise Connectivity*..

To implement security context propagation:

1. Go to the Services→WLEC node in the left pane of the Administration Console.
2. In the right pane of the Administration Console, click the Create a new WLEC Connection Pool link.
3. Define the attributes in the following table:

Table 14-21 WLEC Connection Pool Attributes on the General Tab

Attribute	Description
Name	Name of the WLEC connection pool. The name must be unique for each WLEC connection pool.

Primary Addresses	A list of addresses for IIOP Listener/Handlers that can be used to establish a connection between the WLEC connection pool and the BEA Tuxedo domain. The format of each address is //*hostname*:*port*.
	The addresses must match the ISL addresses defined in the UBBCONFIG file. Multiple addresses are separated by semicolons. For example: //main1.com:1024; //main2.com:1044.
	To configure the WLEC connection pool to use the SSL protocol, use the corbalocs prefix with the address of the IIOP Listener/Handler. For example: corbalocs://*hostname*:*port*.

Failover Addresses	A list of addresses for IIOP Listener/Handlers that are used if connections cannot be established with the addresses defined in the Primary Addresses attribute. Multiple addresses are separated by semicolons. This attribute is optional.

Table 14-21 WLEC Connection Pool Attributes on the General Tab

Attribute	Description
Domain	Name of the BEA Tuxedo domain to which this WLEC connection pool connects. You can have only one WLEC connection pool per BEA Tuxedo domain. The domain name must match the `domainid` parameter in the `RESOURCES` section of the `UBBCONFIG` file for the BEA Tuxedo domain.
Minimum Pool Size	Number of IIOP connections to be added to the WLEC connection pool when WebLogic Server starts. The default is 1.
Maximum Pool Size	Maximum number of IIOP connections that can be made from the WLEC connection pool. The default is 1.

4. Click the Create button.
5. Propagate the security context for a User in a WebLogic Server security realm to a BEA Tuxedo domain by defining the attributes on the Security tab in the WLEC Connection Pool Configuration window in the Administration Console. The following table describes these attributes.

Table 14-22 WLEC Connection Pool Attributes on the Security Tab

Attribute	Description
User Name	A BEA Tuxedo user name. Required only when the security level in the BEA Tuxedo domain is `USER_AUTH`, `ACL` or `MANDATORY_ACL`.
User Password	Password for the User defined in the `User Name` attribute. Required only when you define the `User Name` attribute.
User Role	BEA Tuxedo user role. This attribute is required when the security level in the BEA Tuxedo domain is `APP_PW`, `USER_AUTH`, `ACL`, or `MANDATORY_ACL`.

Table 14-22 WLEC Connection Pool Attributes on the Security Tab

Attribute	Description
Application Password	BEA Tuxedo application password. Required when the security level in the BEA Tuxedo domain is `APP_PW`, `USER_AUTH`, `ACL`, or `MANDATORY_ACL`.
Minimum Encryption Level	Minimum SSL encryption level used between the BEA Tuxedo domain and WebLogic Server. The possible values are 0, 40, 56, and 128. Zero (0) indicates that the data is signed but not sealed. 40, 56, and 128 specify the length, in bits, of the encryption key. If the minimum level of encryption is not met, the SSL connection between BEA Tuxedo and WebLogic Server fails. The default is 40.
Maximum Encryption Level	Maximum SSL encryption level used between the BEA Tuxedo domain and WebLogic Server. The possible values are 0, 40, 56, and 128. Zero (0) indicates that the data is signed but not sealed. 40, 56, and 128 specify the length, in bits, of the encryption key. If the minimum level of encryption is not met, the SSL connection between BEA Tuxedo and WebLogic Server fails. The default is the maximum level allowed by the Encryption Package kit license.
Enable Certificate Authentication	Enables the use of certificate authentication. By default certificate authentication is disabled.
Enable Security Context	Enables the passing of the security context of the WebLogic Server User passed to the BEA Tuxedo domain. By default, security context is disabled.

6. To save your changes, click the Apply button and reboot WebLogic Server.
7. Run the `tpusradd` command to define the WebLogic Server User as an authorized User in the WebLogic Enterprise domain.

8. Set the `-E` option of the `ISL` command to configure the IIOP Listener/Handler to detect and utilize the propagated security context from the WebLogic Server realm. The `-E` option of the `ISL` command requires you to specify a principal name. The principal name defines the principal used by the WLEC connection pool to log in to the WebLogic Enterprise domain. The principal name should match the name defined in the User Name attribute when creating a WLEC connection pool.

Using certificate authentication between the WebLogic Server environment and the BEA Tuxedo environment implies performing a new SSL handshake when establishing a connection from the WebLogic Server environment to a BEA Tuxedo CORBA object. To support multiple client requests over the same SSL network connection, you must set up certificate authentication so that it operates as follows:

1. Obtain a digital certificate for the WebLogic Server User and put the private key in the `TUXDIR/udataobj/security/keys` directory of BEA Tuxedo.
2. In the `UBBCONFIG` file for the BEA Tuxedo CORBA application, use the `tpusradd` command to define the WebLogic Server User as a BEA Tuxedo user.
3. Define the IIOP Listener/Handler in the `UBBCONFIG` file with the `-E` option to indicate the WebLogic Server User is to be used for authentication.
4. Define the WebLogic Server User name in the User Name attribute when creating a WLEC Connection pool in the Administration Console of WebLogic Server.
5. Obtain a digital certificate for the IIOP Listener/Handler.
6. Specify the digital certificate in the `SEC_PRINCIPAL_NAME` option of the ISL command and use the `-S` option to indicate that a secure port should be used for communication between the BEA Tuxedo domain and the WebLogic Server security realm.

For more information about the `UBBCONFIG` file, see Creating a Configuration File in the BEA Tuxedo documentation.

For more information about the `corbalocs` prefix, see Understanding the Address Formats of the Bootstrap Object in the BEA Tuxedo documentation.

For information about BEA Tuxedo security levels, see Defining a Security Level in the BEA Tuxedo documentation.

SSL Certificate Validation

In previous releases, WebLogic Server did not ensure each certificate in a certificate chain was issued by a certificate authority. This problem meant anyone could get a personal certificate from a trusted CA, use that certificate to issue other certificates and WebLogic Server would not detect the invalid certificates. A patch (`CR090101_610sp4`) was made so that all X509 V3 CA certificates used with WebLogic Server must have the Basic Constraint extension defined as CA thus ensuring all certificates in a certificate chain were issued by a certificate authority. By default, any CA certificates not meeting this criteria are rejected. This section provides installation instructions for the patch and describes the command-line argument that controls the level of certificate validation.

Installation Instructions

To install patch `CR090101_610sp4`:

1. Backup the current WebLogic Server installation. If any of the following files were changed, the changes will be lost when the patch is installed onto the current WebLogic Server installation:

```
%WL_HOME%\common\nodemanager\config\democert.pem
%WL_HOME%\common\nodemanager\config\demokey.pem
%WL_HOME%\samples\server\config\examples\demo.crt
%WL_HOME%\samples\server\config\examples\democert.pem
%WL_HOME%\samples\server\config\examples\demokey.pem
%WL_HOME%\samples\server\examples\trusted.crt
%WL_HOME%\samples\server\config\petstore\demo.crt
%WL_HOME%\samples\server\config\petstore\democert.pem
%WL_HOME%\samples\server\config\petstore\demokey.pem
%WL_HOME%\samples\server\config\petstore\trusted.crt
```

```
%WL_HOME%\server\lib\cacerts

%WL_HOME%\server\lib\demo.crt

%WL_HOME%\server\lib\trusted.crt
```

Generally these files have not be modified. However, if you modified these files you will need to decide how to proceed with installing the updated certificates, private keys, and keystores from the patch. For example, you may decide to only select and install the service pack JAR files from the patch.

2. Unzip the contents of the zip files for the patch starting at *`WL_HOME`* and retaining the directory structure in the zip file.

   The following files are added to the WebLogic Server installation:

```
%WL_HOME%\server\lib\CR090101_610sp4_.jar
```

The following files in the WebLogic Server installation are modified:

```
%WL_HOME%\common\nodemanager\config\democert.pem

%WL_HOME%\common\nodemanager\config\demokey.pem

%WL_HOME%\samples\server\config\examples\demo.crt

%WL_HOME%\samples\server\config\examples\democert.pem

%WL_HOME%\samples\server\config\examples\demokey.pem

%WL_HOME%\samples\server\examples\trusted.crt

%WL_HOME%\samples\server\config\petstore\demo.crt

%WL_HOME%\samples\server\config\petstore\democert.pem

%WL_HOME%\samples\server\config\petstore\demokey.pem

%WL_HOME%\samples\server\config\petstore\trusted.crt

%WL_HOME%\server\lib\cacerts

%WL_HOME%\server\lib\demo.crt

%WL_HOME%\server\lib\trusted.crt
```

To avoid name conflicts, the new demo CA certificates have Subject DNs that differ from existing demo CA certificates. The modified Subject DN also makes it easier to differentiate between old and new certificate chains. New demo and demo1024 CA certificates have `Constraints` in the Common Name of their Subject DN.

If you only have the end entity certificate, you can also tell whether it is old or new by looking at the Issuer DN.

3. Update the environment scripts for WebLogic Server to include the JAR files for the patch.
   - On Windows NT, edit the following files:

     `%WL_HOME%\server\bin\setWLSEnv.cmd`

     `%WL_HOME%\server\bin\startWLS.cmd`

     `%WL_HOME%\server\bin\startNodeManager.cmd`

     Add the following to the CLASSPATH before the `weblogic_sp.jar`:

     ```
     %WL_HOME%\server\lib\CR090101_610sp4_webservice.jar;
     %WL_HOME%\server\lib\CR090101_610sp4_.jar;
     ```

   - On UNIX, edit the following files:

     `$WL_HOME/server/bin/setWLSEnv.sh`

     `$WL_HOME/server/bin/startWLS.sh`

     `$WL_HOME/server/bin/startNodeManager.sh`

     Add the following to the CLASSPATH before the `weblogic_sp.jar`:

     ```
     $WL_HOME/server/lib/CR090101_610sp4_webservice.jar;
     $WL_HOME/lib/CR090101_610sp4.jar;
     ```

4. Using the files from the patch, update the certificates, private keys, and keystores in existing domains.

Older unpatched applications will not trust the new demo CA certificates that are installed by this patch. For example, a client that is still trusting the old demo CA certificate, will reject a server that has a new demo certificate chain. Resolve this problem by updating the client's trusted CA list to include the new demo CA supplied by this patch.

Controlling the Level of Certificate Validation

By default WebLogic Server will reject any certificates in a certificate chain that do not have the Basic Constraint extension defined as CA. However, you may be using certificates that do not meet this requirement or you may want to increase the level of security to conform to the IETF RFC 2459 standard. Use the following command-line argument to control the level of certificate validation performed by WebLogic Server:

```
-Dweblogic.security.SSL.enforceConstraints
```

Table 14-23 describes the options for the command-line argument.

Table 14-23 Options for -Dweblogic.security.SSL.enforceConstraints

Option	Description

`strong` or `true`	Use this option to check that the Basic Constraints extension on the CA certificate is defined as CA.
	For example:
	`-Dweblogic.security.SSL.enforceConstraints=strong`
	or
	`-Dweblogic.security.SSL.enforceConstraints=true`
	By default, WebLogic Server performs this level of certificate validation.

`strict`	Use this option to check the Basic Constraints extension on the CA certificate is defined as CA and set to critical. This option enforces the IETF RFC 2459 standard. For example: `-Dweblogic.security.SSL.enforceConstraints=strict` This option is not the default because a number of commercially available CA certificates do not conform to the IETF RFC 2459 standard.
`off`	Use this option to disable certificate validation. Use this option carefully. For example, if you purchased CA certificates from a reputable commercial certificate authority and the certificates do not pass the new validation, use this option. However, CA certificates from most commercial certificate authorities should work with the default `strong` option. For example: `-Dweblogic.security.SSL.enforceConstraints=off` BEA does not recommend use this option in production environment. Instead, purchase new CA certificates that comply with the IETF RFC 2459 standard.

Checking Certificate Chains

WebLogic Server provides a ValidateCertChain command-line utility to check whether or not an existing certificate chain will be rejected by WebLogic Server. The utility uses certificate chains from PEM files, PKCS-12 files, PKCS-12 keystores, and JDK keystores. A complete certificate chain must be used with the utility. The following is the syntax for the ValidateCertChain command-line utility:

```
java utils.ValidateCertChain -file pemcertificatefilename
java utils.ValidateCertChain -pem pemcertificatefilename
java utils.ValidateCertChain -pkcs12store pkcs12storefilename
java utils.ValidateCertChain -pkcs12file pkcs12filename password
java utils.ValidateCertChain -jks alias storefilename [storePass]
```

Example of valid certificate chain:

```
java utils.ValidateCertChain -pem zippychain.pem

Cert[0]: CN=zippy,OU=FOR TESTING
ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US

Cert[1]: CN=CertGenCAB,OU=FOR TESTING
ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US

Certificate chain appears valid
```

Example of invalid certificate chain:

```
java utils.ValidateCertChain -jks mykey mykeystore

Cert[0]: CN=corba1,OU=FOR TESTING ONLY,
O=MyOrganization,L=MyTown,ST=MyState,C=US

CA cert not marked with critical BasicConstraint indicating it is
a CA
Cert[1]: CN=CACERT,OU=FOR TESTING ONLY,
O=MyOrganization,L=MyTown,ST=MyState,C=US

Certificate chain is invalid
```

Troubleshooting Problems with Certificates

If SSL communications were working before the patch but are failing after installing the patch, the problem is mostly likely because the certificate chain used by WebLogic Server is failing the validation.

Determine where the certificate chain is being rejected, and decide whether to update the certificate chain with one that will be accepted or change the setting of the `-Dweblogic.security.SSL.enforceConstraints` command-line argument.

To troubleshoot problems with certificates, use one of the following methods:

- If you know where the certificate chains for the processes using SSL communication are located, use the ValidateCertChain command-line utility to check whether the certificate chains will be accepted.
- Turn on SSL debug tracing on the processes using SSL communication. The syntax for SSL debug tracing is:

  ```
  -Dssl.debug=true -Dweblogic.StdoutDebugEnabled=true
  ```

  The following message indicates the SSL failure is due to problems in the certificate chain:

  ```
  <CA certificate rejected. The basic constraints for a CA
  certificate were not marked for being a CA, or were not marked
  as critical>
  ```

  When using one-way SSL, look for this error in the client log. When using two-way SSL, look for this error in the client and server logs.

CHAPTER

15 Managing Transactions

These sections discuss transaction management and provide guidelines for configuring and managing transactions through the Administration Console.

- Overview of Transaction Management
- Configuring Transactions
- Monitoring and Logging Transactions
- Moving a Server to Another Machine

For information on configuring JDBC connection pools to allow JDBC drivers to participate in distributed transactions, see “Managing JDBC Connectivity” on page 16-1.

Overview of Transaction Management

You use the Administration Console to access tools for configuring and enabling the WebLogic Server features, including the JavaTransaction API (JTA). To invoke the Administration Console, see the procedures provided in Configuring WebLogic Servers and Clusters. The transaction configuration process involves specifying values for attributes. These attributes define various aspects of the transaction environment:

- Transaction timeouts and limits
- Transaction Manager behavior
- Transaction log file prefix

Before configuring your transaction environment, you should be familiar with the J2EE components that can participate in transactions, such as EJBs, JDBC, and JMS.

- EJBs (Enterprise JavaBeans) use JTA for transactions support. Several deployment descriptors relate to transaction handling. For more information about programming with EJBs and JTA, see *Programming WebLogic Enterprise JavaBeans*.
- JDBC (Java Database Connectivity) provides standard interfaces for accessing relational database systems from Java. JTA provides transaction support on connections retrieved using a JDBC driver and transaction data source. For more information about programming with JDBC and JTA, see *Programming WebLogic JDBC*.
- JMS (Java Messaging Service) uses JTA to support transactions across multiple data resources. WebLogic JMS is an XA-compliant resource manager. For more information about programming with JMS and JTA, see *Programming WebLogic JMS*.

For more information about configuring J2EE components, see the applicable sections of this document and the Administration Console Online Help.

Configuring Transactions

The Administration Console provides default values for all JTA configuration attributes. If you specify an invalid value for any configuration attribute, the WebLogic Server does not boot when you restart it.

Configuration settings for JTA are applicable at the domain level. This means that configuration attribute settings apply to all servers within a domain. Monitoring and logging tasks for JTA are performed at the server level.

Once you configure WebLogic JTA and any transaction participants, the system can perform transactions using the JTA API and the WebLogic JTA extensions.

You can configure any transaction attributes before running applications (static configuration) or, with one exception, at application run time (dynamic configuration). The `TransactionLogFilePrefix` attribute must be set before running applications.

To configure transaction attributes, do the following:

1. Start the Administration Console.
2. Select the domain node in the left pane. The Configuration tab for the domain is displayed by default.
3. Select the JTA tab.
4. For each attribute, specify a value or, if available, accept the default value.
5. Click Apply to store new attribute values.
6. Ensure that the `Transaction Log File Prefix` attribute is set when you configure the server. For more information on setting the logging attribute, see "Monitoring and Logging Transactions."

Table 15-1 briefly describes the transaction attributes available with WebLogic Server. For detailed information about attributes, and valid and default values for them, see the Domain topic in the Administration Console Online Help.

Table 15-1 Transaction Attributes

Attribute	Description
`Timeout Seconds`	Time, in seconds, a transaction may be active before the system forces a rollback.
`Abandon Timeout Seconds`	Maximum time, in seconds, that a transaction coordinator persists in attempting to complete a transaction.
`Before Completion Iteration Limit`	Number of `beforeCompletion` callbacks that are processed before the system forces a rollback.
`Max Transactions`	Maximum number of transactions that may be active on a particular server at one time.
`Max Unique Name Statistics`	Maximum number of unique transaction names that may be tracked by a server at one time.
`Forget Heuristics`	Boolean value specifying whether the transaction manager should instruct a resource to forget any transaction with a heuristic outcome.

Additional Attributes for Managing Transactions

By default, if an XA resource that is participating in a global transaction fails to respond to an XA call from the WebLogic Server transaction manager, WebLogic Server flags the resource as unhealthy and unavailable, and blocks any further calls to the resource in an effort to preserve resource threads. The failure can be caused by either an unhealthy transaction or an unhealthy resource—there is no distinction between the two causes. In both cases, the resource is marked as unhealthy.

To mitigate this limitation, WebLogic Server provides the configuration attributes listed in Table 15-2:

Table 15-2 XA Resource Health Monitoring Configuration Attributes

Attribute	MBean	Definition
`EnableResourceHealthMonitoring`	`weblogic.management.configuration.JDBCConnectionPoolMBean`	Enables or disables resource health monitoring for the JDBC connection pool. This attribute only applies to connection pools that use an XA JDBC driver for database connections. It is ignored if a non-XA JDBC driver is used. If set to `true`, resource health monitoring is enabled. If an XA resource fails to respond to an XA call within the period specified in the `MaxXACallMillis` attribute, WebLogic Server marks the connection pool as unhealthy and blocks any further calls to the resource. If set to `false`, the feature is disabled. Default: `true`
`MaxXACallMillis`	`weblogic.management.configuration.JTAMBean`	Sets the maximum allowed duration (in milliseconds) of XA calls to XA resources. This setting applies to the entire domain. Default: `120000`

Table 15-2 XA Resource Health Monitoring Configuration Attributes

Attribute	MBean	Definition
`MaxResourceUnavailableMillis`	`weblogic.management.configuration.JTAMBean`	The maximum duration (in milliseconds) that an XA resource is marked as unhealthy. After this duration, the XA resource is declared available again, even if the resource is not explicitly re-registered with the transaction manager. This setting applies to the entire domain. Default: `1800000`
`MaxResourceRequestOnServer`	`weblogic.management.configuration.JTAMBean`	Maximum number of concurrent requests to resources allowed for each server in the domain. Default: `50` Minimum: `10` Maximum: `java.lang.Integer.MAX_VALUE`

You set these attributes directly in the config.xml file when the domain is inactive. These attributes are not available in the Administration Console. The following example shows an excerpt of a configuration file with these attributes:

```
...
   <JTA
    MaxUniqueNameStatistics="5"
    TimeoutSeconds="300"
    RecoveryThresholdMillis="150000"
    MaxResourceUnavailableMillis="900000"
    MaxResourceRequestOnServer="60"
    MaxXACallMillis="180000"
   />

 <JDBCConnectionPool
  Name="XAPool"
  Targets="myserver"
  DriverName="weblogic.qa.tests.transaction.
    testsupport.jdbc.XADataSource"
  InitialCapacity="1"
  MaxCapacity="10"
  CapacityIncrement="2"
  RefreshMinutes="5"
  TestTableName="dual"
```

```
  EnableResourceHealthMonitoring="true"
  Properties="user=scott;password=tiger;server=dbserver1"
 />

 <JDBCTxDataSource
  Name="XADataSource"
  Targets="myserver"
  JNDIName="weblogic.jdbc.XADS"
  PoolName="XAPool"
 />

...
```

Monitoring and Logging Transactions

The Administration Console allows you to monitor transactions and to specify the transaction log file prefix. Monitoring and logging tasks are performed at the server level. Transaction statistics are displayed for a specific server and each server has a transaction log file.

To display transaction statistics and to set the prefix for the transaction log files, do the following:

1. Start the Administration Console.
2. Click the server node in the left pane.
3. Select a specific server in the left pane.
4. Select the Monitoring tab.
5. Select the JTA tab. Totals for transaction statistics are displayed in the JTA dialog. (You can also click the monitoring text links to monitor transactions by resource or by name, or to monitor all active transactions.)
6. Select the Logging tab.
7. Select the JTA tab.
8. Enter a transaction log file prefix then click Apply to save the attribute setting.

For detailed information on monitoring and logging values and attributes, see the Server topic in the Administration Console Online Help.

Moving a Server to Another Machine

When an applications server is moved to another machine, it must be able to locate the transaction log files on the new disk. For this reason, we recommend moving the transaction log files to the new machine before starting the server there. By doing so, you can ensure that recovery runs properly. If the pathname is different on the new machine, update the `TransactionLogFilePrefix` attribute with the new path before starting the server.

When migrating transaction logs after a server failure, make all transaction log files available on the new machine before starting the server there. You can accomplish this by storing transaction log files on a dual-ported disk available to both machines. As in the case of a planned migration, update the `TransactionLogFilePrefix` attribute with the new path before starting the server if the pathname is different on the new machine. Ensure that all transaction log files are available on the new machine before the server is started there. Otherwise, transactions in the process of being committed at the time of a crash might not be resolved correctly, resulting in application data inconsistencies.

CHAPTER

16 Managing JDBC Connectivity

The following sections provide guidelines for configuring and managing database connectivity through the JDBC components—Data Sources, Connection Pools and MultiPools—for both local and distributed transactions:

Overview of JDBC Administration

The Administration Console provides an interface to the tools that allow you to configure and manage WebLogic Server features, including JDBC (database connectivity with Java). For most JDBC administrative functions, which include creating, managing and monitoring connectivity, systems administrators use the Administrative Console or the command-line interface. Application developers may want to use the JDBC API.

Frequently performed tasks to set and manage connectivity include:

- Defining the attributes that govern JDBC connectivity between WebLogic Server and your database management system
- Managing established connectivity
- Monitoring established connectivity

About the Administrative Console

Your primary way to set and manage JDBC connectivity is through the Administration Console. Using the Administration Console, you set connectivity statically prior to starting the server. For more information, see “Administration Console” on page 1-4.

In addition to setting connectivity, the Administration Console allows you to manage and monitor established connectivity.

About the Command-Line Interface

The command-line interface provides a way to dynamically create and manage Connection Pools. For information on how to use the command-line interface, see Appendix B, “WebLogic Server Command-Line Interface Reference.”

About the JDBC API

For information on setting and managing connectivity programatically, see Programming WebLogic JDBC at http://e-docs.bea.com/wls/docs61/jdbc/index.html.

Related Information

The JDBC drivers, used locally and in distributed transactions, interface with many WebLogic Server components and information appears in several documents. For example, information about JDBC drivers is included in the documentation sets for JDBC, JTA and WebLogic jDrivers.

Here is a list of additional resources for JDBC, JTA and Administration:

Administration and Management

- For instructions on opening the Administration Console, refer to Chapter 4, "Configuring WebLogic Servers and Clusters."
- For a complete list of the JDBC attributes, see JDBC Connection Pool, JDBC Data Sources, JDBC MultiPools, and JDBC Transaction Data Sources in the WebLogic Administration Console Online Help at http://e-docs.bea.com/wls/docs61/ConsoleHelp/index.html.
- For information about using the command-line interface, see Appendix B, "WebLogic Server Command-Line Interface Reference.".

JDBC and WebLogic jDrivers

The following documentation is written primarily for application developers. Systems Administrators may want to read the introductory material as a supplement to the material in this document.

- For information on the JDBC API, see *Programming WebLogic JDBC*. The "Introduction to WebLogic JDBC" section provides a concise overview of JDBC and JDBC drivers.
- For information on using the WebLogic jDrivers, see *Installing and Using WebLogic jDriver for Oracle* at http://e-docs.bea.com/wls/docs61/oracle/index.html, *Installing and Using WebLogic jDriver for Microsoft SQL Server* at http://e-docs.bea.com/wls/docs61/mssqlserver4/index.html, or *Installing and Using WebLogic jDriver for Informix* at http://e-docs.bea.com/wls/docs61/informix4/index.html.

Transactions (JTA)

- For information on managing JTA, see Chapter 15, “Managing Transactions.”
- For information on using third-party drivers, see "Using Third-Party JDBC XA Drivers with WebLogic Server" in ProgrammingWebLogic JTA at http://e-docs.bea.com/wls/docs61/jta/thirdpartytx.html.

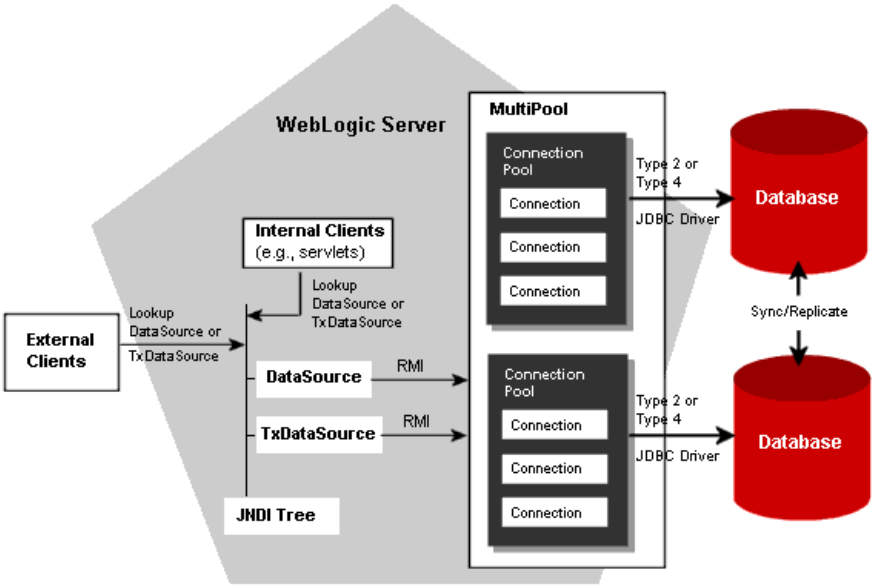
The following documentation is written primarily for application developers. Systems Administrators may want to read the following as supplements to the material in this section.

- For information on distributed transactions, see *Programming WebLogic JTA* at http://e-docs.bea.com/wls/docs61/jta/index.html.
- For information on using the WebLogic jDriver for Oracle/XA, see "Using WebLogic jDriver for Oracle/XA in Distributed Transactions" in *Installing and Using WebLogic jDriver for Oracle* at http://e-docs.bea.com/wls/docs61/oracle/trxjdbcx.html.

JDBC Components—Connection Pools, Data Sources, and MultiPools

The following sections provide a brief overview of the JDBC connectivity components—Connection Pools, MultiPools, and Data Sources.

Figure 16-1 JDBC Components in WebLogic Server



Connection Pools

A Connection Pool contains named groups of JDBC connections that are created when the Connection Pool is registered, usually when starting up WebLogic Server. Your application borrows a connection from the pool, uses it, then returns it to the pool by closing it. Read more about Connection Pools in Programming WebLogic JDBC at http://e-docs.bea.com/wls/docs61/jdbc/programming.html.

All of the settings you make with the Administration Console are static; that is, all settings are made before WebLogic Server starts. You can create dynamic Connection Pools—after the server starts—using the command line (see Appendix B, "WebLogic Server Command-Line Interface Reference,") or programatically using the API (see Creating a Dynamic Connection Pool in *Programming WebLogic JDBC*).

MultiPools

MultiPools aid in either:

- Load Balancing—pools are added without any attached ordering and are accessed using a round-robin scheme. When switching connections, the Connection Pool just after the last pool accessed is selected.
- High Availability—set up pools as an ordered list that determines the order in which Connection Pool switching occurs. For example, the first pool on the list is selected, then the second, etc.

All of the connections in a particular Connection Pool are identical; that is, they are attached to a single database. The Connection Pools within a MultiPool may, however, be associated with different DBMS. Read more about MultiPools in Programming WebLogic JDBC at http://e-docs.bea.com/wls/docs61/jdbc/programming.html.

Data Sources

A Data Source object enables JDBC applications to obtain a DBMS connection from a connection pool. Each Data Source object binds to the JNDI tree and points to a connection pool or MultiPool. Applications look up the Data Source to get a connection. Data Source objects can be defined with JTA (Tx Data Sources in the Administration Console) or without JTA (Data Sources in the Administration Console). You use Tx Data Source for distributed transactions. See "JDBC Configuration Guidelines for Connection Pools, MultiPools and DataSources" on page 16-7 for more information about using Data Sources and Tx Data Sources.

JDBC Configuration Guidelines for Connection Pools, MultiPools and DataSources

This section describes JDBC configuration guidelines for local and distributed transactions.

Overview of JDBC Configuration

To set up JDBC connectivity, you configure Connection Pools, Data Source objects (always recommended, but optional in some cases), and MultiPools (optional) by defining attributes in the Administration Console and, for dynamic connection pools, at the command line. There are three types of transactions:

- Local—non-distributed transaction
- Distributed with XA Driver—two-phase commit
- Distributed with non-XA Driver—single resource manager and single database instance

The following table describes how to use these objects in local and distributed transactions:

Table 16-1 Summary of JDBC Configuration Guidelines

Description/Object	Local Transactions	Distributed Transactions XA Driver	Distributed Transactions Non-XA Driver
JDBC driver	■ WebLogic jDriver for Oracle, Microsoft SQL Server, and Informix. ■ Compliant third-party drivers.	■ WebLogic jDriver for Oracle/XA. ■ Compliant third-party drivers.	■ WebLogic jDriver for Oracle, Microsoft SQL Server, and Informix. ■ Compliant third-party drivers.
Data Source	Data Source object recommended. (If there is no Data Source, use the JDBC API.)	Tx Data Source required.	Tx Data Source required. Set `enable two-phase commit=true` if more than one resource. See "Configuring Non-XA JDBC Drivers for Distributed Transactions" on page 16-20.
Connection Pool	Requires Data Source object when configuring in the Administration Console.	Requires TXData Source.	Requires TXData Source.
MultiPool	Connection Pool and Data Source required.	Requires TXData Source.	Requires TXData Source.

Note: For distributed transactions, use an XA-compliant driver, such as the *WebLogic jDriver for Oracle/XA*, which is the XA compliant version of the WebLogic jDriver for Oracle.

When to Use a Tx Data Source

If your applications or environment meet any of the following criteria, you should use a Tx Data Source instead of a Data Source:

- Use the Java Transaction API (JTA)
- Use the EJB container in WebLogic Server to manage transactions
- Include multiple database updates within a single transaction
- Access multiple resources, such as a database and the Java Messaging Service (JMS), during a transaction
- Use the same connection pool on multiple servers

With an EJB architecture, it is common for multiple EJBs that are doing database work to be invoked as part of a single transaction. Without XA, the only way for this to work is if all transaction participants use the exact same database connection. WebLogic Server uses the JTS driver and a TxDataSource (with Enable Two-Phase Commit selected) to do this behind the scenes without requiring you to explicitly pass the JDBC connection from EJB to EJB. With XA (requires an XA driver), you can use a Tx Data Source in WebLogic Server for distributed transactions with two-phase commit so that EJBs can use a different database connections for each part of the transaction. In either case (with or without XA), you should use a Tx Data Source.

Read more about Data Sources in *Programming WebLogic JDBC* at http://e-docs.bea.com/wls/docs61/jdbc/programming.html.

Note: Do not create two Tx Data Sources that point to the same connection pool. If a transaction uses two different Tx Data Sources which are both pointed to the same connection pool, you will get an XA_PROTO error when you try to access the second connection.

Drivers Supported for Local Transactions

- JDBC 2.0 drivers that support the JDBC Core 2.0 API (`java.sql`), including the WebLogic jDrivers for Oracle, Microsoft SQL Server, and Informix. The API allows you to create the class objects necessary to establish a connection with a data source, send queries and update statements to the data source, and process the results.

Drivers Supported for Distributed Transactions

- Any JDBC 2.0 driver that supports JDBC 2.0 distributed transactions standard extension interfaces (`javax.sql.XADataSource`, `javax.sql.XAConnection`,

`javax.transaction.xa.XAResource`), including the WebLogic jDriver for Oracle/XA.

- Any JDBC driver that supports JDBC 2.0 Core API but does not support JDBC 2.0 distributed transactions standard extension interfaces. Only one non-XA JDBC driver at a time can participate in a distributed transaction. See "Configuring Non-XA JDBC Drivers for Distributed Transactions" on page 16-20.

Configuring JDBC Drivers for Local Transactions

To configure JDBC drivers for local transactions, set up the JDBC Connection Pool as follows:

- Specify the Driver Classname attribute as the name of the class supporting the `java.sql.Driver` interface.
- Specify the database properties. These properties are passed to the specific driver as driver properties.

In the Administration Console, you specify connection properties as a name=value pair with each property on its own line. In the configuration file (`config.xml`), connection properties are listed in a string separated by semicolons. For example:

```
Properties="user=SCOTT;server=DEMO"
```

For more information on WebLogic two-tier JDBC drivers, refer to the BEA documentation for the specific driver you are using: *Installing and Using WebLogic jDriver for Oracle* at http://e-docs.bea.com/wls/docs61/oracle/index.html, *Installing and Using WebLogic jDriver for Microsoft SQL Server* at http://e-docs.bea.com/wls/docs61/mssqlserver4/index.html, or *Installing and Using WebLogic jDriver for Informix* at http://e-docs.bea.com/wls/docs61/informix4/index.html. If you are using a third-party driver, refer to Using Third-Party JDBC XA Drivers with WebLogic Server in *Programming WebLogic JTA* at http://e-docs.bea.com/wls/docs61/jta/thirdpartytx.html and the vendor-specific documentation. The following tables show sample JDBC Connection Pool and Data Source configurations using the WebLogic jDrivers.

The following table shows a sample Connection Pool configuration using the WebLogic jDriver for Oracle.

Note: New Property: "Password." This value overrides any password defined in Properties (as a name/value pair). This attribute is passed to the 2-tier JDBC driver when creating physical database connections. The value is stored in an encrypted form in the config.xml and can be used to avoid storing cleartext passwords in that file.

Table 16-2 WebLogic jDriver for Oracle: Connection Pool Configuration

Attribute Name	**Attribute Value**
Name	`myConnectionPool`
Targets	`myserver`
DriverClassname	`weblogic.jdbc.oci.Driver`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`
Properties	`user=scott` `server=localdb`
Password	`Tiger` (this value overrides any password defined in Properties as a name value pair)

The following table shows a sample Data Source configuration using the WebLogic jDriver for Oracle.

Table 16-3 WebLogic jDriver for Oracle: Data Source Configuration

Attribute Name	**Attribute Value**
Name	`myDataSource`
Targets	`myserver`
JNDIName	`myconnection`
PoolName	`myConnectionPool`

The following table shows a sample Connection Pool configuration using the WebLogic jDriver for Microsoft SQL Server.

Table 16-4 WebLogic jDriver for Microsoft SQL Server: Connection Pool Configuration

Attribute Name	**Attribute Value**
Name	`myConnectionPool`
Targets	`myserver`
DriverClassname	`weblogic.jdbc.mssqlserver4.Driver`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`

Properties	`user=sa`
	`password=secret`
	`db=pubs`
	`server=myHost:1433`
	`appname=MyApplication`
	`hostname=myhostName`

The following table shows a sample Data Source configuration using the WebLogic jDriver for Microsoft SQL Server.

Table 16-5 WebLogic jDriver for Microsoft SQL Server: Data Source Configuration

Attribute Name	**Attribute Value**
Name	`myDataSource`
Targets	`myserver`
JNDIName	`myconnection`
PoolName	`myConnectionPool`

The following table shows a sample Connection Pool configuration using the WebLogic jDriver for Informix.

Table 16-6 WebLogic jDriver for Informix: Connection Pool Configuration

Attribute Name	**Attribute Value**
Name	`myConnectionPool`
Targets	`myserver`
DriverClassname	`weblogic.jdbc.informix4.Driver`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`

Properties	`user=informix`
	`password=secret`
	`server=myDBHost`
	`port=1493`
	`db=myDB`

The following table shows a sample Data Source configuration using the WebLogic jDriver for Informix.

Table 16-7 WebLogic jDriver for Informix: Data Source Configuration

Attribute Name	**Attribute Value**
Name	`myDataSource`
Targets	`myserver`
JNDIName	`myconnection`
PoolName	`myConnectionPool`

Configuring XA JDBC Drivers for Distributed Transactions

To allow XA JDBC drivers to participate in distributed transactions, configure the JDBC Connection Pool as follows:

- Specify the `Driver Classname` attribute as the name of the class supporting the `javax.sql.XADataSource` interface.
- Make sure that the database properties are specified. These properties are passed to the specified `XADataSource` as data source properties. For more information on data source properties for the WebLogic jDriver for Oracle, see "WebLogic jDriver for Oracle/XA Data Source Properties." For information about data source properties for third-party drivers, see the vendor documentation.
- See "Additional XA Connection Pool Properties" on page 16-20 for any additional connection pool properties that may be required to support XA for your DBMS.

The following attributes are an example of a JDBC Connection Pool configuration using the WebLogic jDriver for Oracle in XA mode.

Table 16-8 WebLogic jDriver for Oracle/XA: Connection Pool Configuration

Attribute Name	**Attribute Value**
Name	`fundsXferAppPool`
Targets	`myserver`
DriverClassname	`weblogic.jdbc.oci.xa.XADataSource`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`

Properties	`user=scott`
	`password=tiger`
	`server=localdb`

The following attributes are an example of a Tx Data Source configuration using the WebLogic jDriver for Oracle in XA mode.

Table 16-9 WebLogic jDriver for Oracle/XA: Tx Data Source

Attribute Name	Attribute Value
Name	`fundsXferData Source`
Targets	`myserver`
JNDIName	`myapp.fundsXfer`
PoolName	`fundsXferAppPool`

You can also configure the JDBC Connection Pool to use a third-party vendor's driver in XA mode. In such cases, the data source properties are set via reflection on the `XADataSource` instance using the JavaBeans design pattern. In other words, for property `abc`, the `XADataSource` instance must support get and set methods with the names `getAbc` and `setAbc`, respectively.

The following attributes are an example of a JDBC Connection Pool configuration using the Oracle Thin Driver.

Table 16-10 Oracle Thin Driver: Connection Pool Configuration

Attribute Name	Attribute Value
Name	`jtaXAPool`
URL	`jdbc:oracle:thin:@baybridge:1521:bay817`
Targets	`myserver,server1`
DriverClassname	`oracle.jdbc.xa.client.OracleXADataSource`
Initial Capacity	`1`
MaxCapacity	`20`
CapacityIncrement	`2`
Properties	`user=scott` `password=tiger`

The following attributes are an example of a Tx Data Source configuration using the Oracle Thin Driver.

Table 16-11 Oracle Thin Driver: Tx Data Source Configuration

Attribute Name	**Attribute Value**
Name	`jtaXADS`
Targets	`myserver,server1`
JNDIName	`jtaXADS`
PoolName	`jtaXAPool`

Configure the JDBC Connection Pool for use with a Cloudscape driver as follows.

Table 16-12 Cloudscape: Connection Pool Configuration

Attribute Name	**Attribute Value**
Name	`jtaXAPool`
Targets	`myserver,server1`
DriverClassname	`COM.cloudscape.core.XADataSource`
Initial Capacity	`1`
MaxCapacity	`10`
CapacityIncrement	`2`
Properties	`databaseName=CloudscapeDB`
SupportsLocalTransaction	`true`

Configure the Tx Data Source for use with a Cloudscape driver as follows.

Table 16-13 Cloudscape: Tx Data Source Configuration

Attribute Name	**Attribute Value**
Name	`jtaZADS`
Targets	`myserver,myserver1`
JNDIName	`JTAXADS`
PoolName	`jtaXAPool`

WebLogic jDriver for Oracle/XA Data Source Properties

Table 16-14 lists the data source properties supported by the WebLogic jDriver for Oracle. The JDBC 2.0 column indicates whether a specific data source property is a JDBC 2.0 standard data source property (Y) or a WebLogic Server extension to JDBC (N).

The Optional column indicates whether a particular data source property is optional or not. Properties marked with Y* are mapped to the corresponding fields of the Oracle `xa_open` string (value of the `openString` property) as listed in Table 16-14. If they are not specified, their default values are taken from the `openString` property. If they are specified, their values should match those specified in the `openString` property. If the properties do not match, a `SQLException` is thrown when you attempt to make an XA connection.

Mandatory properties marked with N* are also mapped to the corresponding fields of the Oracle `xa_open` string. Specify these properties when specifying the Oracle `xa_open` string. If they are not specified or if they are specified but do not match, an `SQLException` is thrown when you attempt to make an XA connection.

Property Names marked with ** are supported, but not used, by WebLogic Server.

Table 16-14 Data Source Properties for WebLogic jDriver for Oracle/XA

Property Name	Type	Description	JDBC 2.0	Optional	Default Value
databaseName**	String	Name of a particular database on a server.	Y	Y	None
dataSourceName	String	A data source name; used to name an underlying XADataSource.	Y	Y	Connection Pool Name
description	String	Description of this data source.	Y	Y	None
networkProtocol**	String	Network protocol used to communicate with the server.	Y	Y	None
password	String	A database password.	Y	N*	None
portNumber**	Int	Port number at which a server is listening for requests.	Y	Y	None
roleName**	String	The initial SQL role name.	Y	Y	None

Table 16-14 Data Source Properties for WebLogic jDriver for Oracle/XA

Property Name	Type	Description	JDBC 2.0	Optional	Default Value
serverName	String	Database server name.	Y	Y*	None
user	String	User's account name.	Y	N*	None
openString	String	Oracle's XA open string.	N	Y	None
oracleXATrace	String	Indicates whether XA tracing output is enabled. If enabled (true), a file with a name in the form of `xa_poolnamedate.trc` is placed in the directory in which the server is started.	N	Y	false

Table 16-15 lists the mapping between Oracle's `xa_open` string fields and data source properties.

Table 16-15 Mapping of xa_open String Names to JDBC Data Source Properties

Oracle xa_open String Field Name	JDBC 2.0 Data Source Property	Optional
acc	user, password	N
sqlnet	ServerName	

Note also that users must specify `Threads=true` in Oracle's `xa_open` string. For complete description of Oracle's `xa_open` string fields, see your Oracle documentation.

Additional XA Connection Pool Properties

When using connections from a connection pool in distributed transactions, you may need to set additional properties for the connection pool so that the connection pool handles the connection properly within WebLogic Server in the context of the transaction. You set these properties in the configuration file (`config.xml`) within the `JDBCConnectionPool` tag. By default, all additional properties are set to false. You set the properties to true to enable them.

In many cases, WebLogic Server automatically sets the proper value for these properties internally so that you do not have to set them manually.

KeepXAConnTillTxComplete

Some DBMSs require that you start and end a transaction in the same physical database connection. In some cases, a transaction in WebLogic Server may start in one physical database connection and end in another physical database connection. To force a connection pool to reserve a physical connection and provide the *same* connection to an application throughout transaction processing until the transaction is complete, you set `KeepXAConnTillTxComplete="true"`. For example:

```
<JDBCConnectionPool KeepXAConnTillTxComplete="true"
DriverName="com.sybase.jdbc2.jdbc.SybXADataSource"
CapacityIncrement="5" InitialCapacity="10" MaxCapacity="25"
Name="demoXAPool" Password="{3DES}vIF8diu4H0QmdfOipd4dWA=="
Properties="User=dbuser;DatabaseName=dbname;ServerName=server_nam
e_or_IP_address;PortNumber=serverPortNumber;NetworkProtocol=Tds;r
esourceManagerName=Lrm_name_in_xa_config;resourceManagerType=2" />
```

Note: This property is *required* to support distributed transactions with DB2 and Sybase.

Configuring Non-XA JDBC Drivers for Distributed Transactions

When configuring the JDBC Connection Pool to allow non-XA JDBC drivers to participate with other resources in distributed transactions, specify the Enable Two-Phase Commit attribute for the JDBC Tx Data Source. (This parameter is ignored by resources that support the `XAResource` interface.) Note that only one non-XA connection pool at a time may participate in a distributed transaction.

Non-XA Driver/Single Resource

If you are using only one non-XA driver and it is the only resource in the transaction, leave the Enable Two-Phase Commit option unselected in the Administration Console (accept the default `enableTwoPhaseCommit = false`). In this case, Weblogic Server ignores the setting and the Transaction Manager performs a one-phase optimization.

Non-XA Driver/Multiple Resources

If you are using one non-XA JDBC driver with other XA resources, select Enable Two-Phase Commit in the Console (`enableTwoPhaseCommit = true`).

When `enableTwoPhaseCommit` is set to `true`, the non-XA JDBC resource always returns `XA_OK` during the `XAResource.prepare()` method call. The resource attempts to commit or roll back its local transaction in response to subsequent `XAResource.commit()` or `XAResource.rollback()` calls. If the resource commit or rollback fails, a heuristic error results. Application data may be left in an inconsistent state as a result of a heuristic failure.

When Enable Two-Phase Commit is not selected (`enableTwoPhaseCommit` is set to `false`), the non-XA JDBC resource causes `XAResource.prepare()` to fail. This mechanism ensures that there is only one participant in the transaction, as `commit()` throws a `SystemException` in this case. When there is only one resource participating in a transaction, the one phase optimization bypasses `XAResource.prepare()`, and the transaction commits successfully in most instances.

This non-XA JDBC driver support is often referred to as the "JTS driver" because WebLogic Server uses the WebLogic JTS Driver internally to support the feature. For more information about the WebLogic JTS Driver, see "Using the WebLogic JTS Driver" in *Programming WebLogic JDBC*.

Limitations and Risks When Using a Non-XA Driver in Global Transactions

WebLogic Server supports the participation of non-XA JDBC resources in global transactions, but there are limitations that you must consider when designing applications to use such resources. Because a non-XA driver does not adhere to the XA/2PC contracts and only supports one-phase commit and rollback operations, WebLogic Server (through the JTS driver) has to make compromises to allow the resource to participate in a transaction controlled by the Transaction Manager.

Heuristic Completions and Data Inconsistency

When Enable Two-Phase Commit is selected for a non-XA resource, (`enableTwoPhaseCommit = true`), the prepare phase of the transaction for the non-XA resource always succeeds. Therefore, the non-XA resource does not truly participate in the two-phase commit (2PC) protocol and is susceptible to failures. If a failure occurs in the non-XA resource after the prepare phase, the non-XA resource is likely to roll back the transaction while XA transaction participants will commit the transaction, resulting in a heuristic completion and data inconsistencies.

Because of the data integrity risks, the Enable Two-Phase Commit option should only be used in applications that can tolerate heuristic conditions.

Cannot Recover Pending Transactions

Because a non-XA driver manipulates local database transactions only, there is no concept of a transaction pending state in the database with regard to an external transaction manager. When `XAResource.recover()` is called on the non-XA resource, it always returns an empty set of Xids (transaction IDs), even though there may be transactions that need to be committed or rolled back. Therefore, applications that use a non-XA resource in a global transaction cannot recover from a system failure and maintain data integrity.

Possible Performance Loss with Non-XA Resources in Multi-Server Configurations

Because WebLogic Server relies on the database local transaction associated with a particular JDBC connection to support non-XA resource participation in a global transaction, when the same JDBC data source is accessed by an application with a global transaction context on multiple WebLogic Server instances, the JTS driver will always route JDBC operations to the first connection established by the application in the transaction. For example, if an application starts a transaction on one server, accesses a non-XA JDBC resource, then makes a remote method invocation (RMI) call to another server and accesses a data source that uses the same underlying JDBC driver, the JTS driver recognizes that the resource has a connection associated with the transaction on another server and sets up an RMI redirection to the actual connection on the first server. All operations on the connection are made on the one connection that was established on the first server. This behavior can result in a performance loss due to the overhead associated with setting up these remote connections and making the RMI calls to the one physical connection.

Only One Non-XA Participant

When a non-XA resource (with `enableTwoPhaseCommit = true`) is registered with the WebLogic Server Transaction Manager, it is registered with the name of the class that implements the XAResource interface. Since all non-XA resources with `enableTwoPhaseCommit = true` use the JTS driver for the XAResource interface, all non-XA resources (with `enableTwoPhaseCommit = true`) that participate in a global transaction are registered with the same name. If you use more than one non-XA resource in a global transaction, you will see naming conflicts or possible heuristic failures.

Non-XA Connection Pool and Tx Data Source Configuration Example

The following shows configuration attributes for a sample JDBC Connection Pool using a non-XA JDBC driver.

Table 16-16 WebLogic jDriver for Oracle: Connection Pool Configuration

Attribute Name	**Attribute Value**
Name	`fundsXferAppPool`
Targets	`myserver`
URL	`jdbc:weblogic:oracle`
DriverClassname	`weblogic.jdbc.oci.Driver`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`

Properties	`user=scott`
	`password=tiger`
	`server=localdb`

The following table shows configuration attributes for a sample Tx Data Source using a non-XA JDBC driver.

Table 16-17 WebLogic j Driver for Oracle: Tx Data Source Configuration

Attribute Name	**Attribute Value**
Name	`fundsXferDataSource`
Targets	`myserver,server1`
JNDIName	`myapp.fundsXfer`
PoolName	`fundsXferAppPool`
EnableTwoPhaseCommit	`true`

Configuring and Managing JDBC Connection Pools, MultiPools, and DataSources Using the Administration Console

The following sections discuss how to set database connectivity by configuring JDBC components—Connection Pools, Data Sources, and MultiPools. Once connectivity is established, you use either the Administration Console or command-line interface to manage and monitor connectivity. See Table 16-19 for descriptions of the configuration tasks and links to the Administration Console Online Help.

JDBC Configuration

In this section, we define *configuration* as including these processes:

Creating the JDBC Objects

Using the Administration Console, you create the JDBC components—Connection Pools, Data Sources, and MultiPools—by specifying attributes and database properties. See "Configuring JDBC Connectivity Using the Administration Console" on page 16-27.

First you create the connection pool or MultiPool, then the Data Source. When you create a Data Source object, you specify a connection pool or MultiPool as one of the Data Source attributes. This permanently *associates* that Data Source with a specific Connection Pool or MultiPool ("pool").

Assign the JDBC Objects

Once you configure the Data Source and associated Connection Pool (or MultiPool), you then assign each object to the same servers or clusters. Some common scenarios are as follows:

- In a cluster, assign the Data Source to the cluster, and assign the associated Connection Pool to each managed server in the cluster.
- In a single server configuration, assign each Data Source and its associated Connection Pool to the server.
- If you are using a MultiPool, assign the Connection Pools to the MultiPool; then assign the Data Source and all Connection Pools and the MultiPool to the server(s) or cluster(s).

See "Configuring JDBC Connectivity Using the Administration Console" on page 16-27 for a description of the tasks you perform.

Refer to the following table for more information on association and assignment in the configuration process.

Table 16-18 Association and Assignment Scenarios

Scenario #	Associated. . .	Assign	Target Description
1	Data Source A with Connection Pool A	1. Data Source A to Managed Server 1, and 2. Connection Pool A to Managed Server 1.	Data Source and Connection Pool assigned to the same target.
2	Data Source B with Connection Pool B	1. Data Source B to Cluster X, then 2. Connection Pool B to Cluster X.	Data Source and Connection assigned to related server/cluster targets.
3	Data Source C with Connection Pool C	■ Data Source A and Connection Pool A to Managed Server 1. - AND - ■ Data Source A to Cluster X; then assign Connection Pool A to Cluster X.	Data Source and Connection Pool assigned as a unit to two different targets.

(You can assign more than one Data Source to a pool, but there is no practical purpose for this.) You can assign these Data Source/pool combinations to more than one server or cluster, but they must be assigned in combination. For example, you can't assign a DataSource to Managed Server A if its associated Connection Pool is assigned only to Server B.

You can configure dynamic Connection Pools (after the server starts) using the command-line interface. See "JDBC Configuration Tasks Using the Command-Line Interface" on page 16-29. You can also configure dynamic Connection Pools programatically using the API (see Creating a Dynamic Connection Pool in *Programming WebLogic JDBC*).

Configuring JDBC Connectivity Using the Administration Console

The Administration Console allows you to configure, manage, and monitor JDBC connectivity. To display the tabs that you use to perform these tasks, complete the following procedure:

1. Start the Administration Console.
2. Locate the Services node in the left pane, then expand the JDBC node.
3. Select the tab specific to the component you want to configure or manage—Connection Pools, MultiPools, Data Source, or Tx Data Source.
4. Follow the instructions in the Online Help. For links to the Online Help, see Table 16-19.

The following table shows the connectivity tasks, listed in typical order in which you perform them. You may change the order; just remember you must configure an object before associating or assigning it.

Table 16-19 JDBC Configuration Tasks

Task #	JDBC Component/ Task	Description
1	Configure a Connection Pool	On the Configuration tabs, you set the attributes for the Connection Pool, such as Name, URL, and database Properties.
2	Clone a Connection Pool (Optional)	This task copies a Connection Pool. On the Configuration tabs, you change Name of pool to a unique name; and accept or change the remaining attributes. This a useful feature when you want to have identical pool configurations with different names. For example, you may want to have each database administrator use a certain pool to track individual changes to a database.
3	Configure a MultiPool (Optional)	On the MultiPool tabs, you set the attributes for the name and algorithm type, either High Availability or Load Balancing. On the Pool tab, you assign the Connection Pools to this MultiPool.

Table 16-19 JDBC Configuration Tasks

Task #	JDBC Component/ Task	Description
4	Configure a Data Source (and Associate with a Pool)	Using the Data Source tab, set the attributes for the Data Source, including the Name, JNDI Name, and Pool Name (this associates, or assigns, the Data Source with a specific pool—Connection Pool or MultiPool.)
5	Configure a Tx Data Source (and Associate with a Connection Pool)	Using the Tx Data Source tab, set the attributes for the Tx Data Source, including the Name, JNDI Name, and *Connection Pool* Name (this associates, or assigns, the Data Source with a specific pool).
6	Assign a Connection Pool to the Servers/Clusters	Using the Target tab, you assign the Connection Pool to one or more Servers or Clusters. See Table 16-18 Association and Assigninment Scenarios.
7	Assign the MultiPool to Servers or Clusters	Using the Target tab, you assign the configured MultiPool to Servers or Clusters.

Database Passwords in Connection Pool Configuration

When you create a connection pool, you typically include at least one password to connect to the database. If you use an open string to enable XA, you may use two passwords. WebLogic Server includes the following fields on the JDBC Connection Pool→Configuration→General tab:

- **`Password`**. Use this field to set the database password. If set, this value overrides any `password` value defined in the `Properties` passed to the tier-2 JDBC Driver when creating physical database connections. The value is encrypted in the `config.xml` file (stored as the `Password` attribute in the `JDBCConnectionPool` tag) and is hidden on the administration console.
- **`Open String Password`**. Use this field to set the password in the open string that the transaction manager in WebLogic Server uses to open a database connection. If set, this value overrides any password defined as part of the open string in the `Properties` field. The value is encrypted in the `config.xml` file (stored as the `XAPassword` attribute in the `JDBCConnectionPool` tag) and is hidden on the Administration Console. At runtime, WebLogic Server reconstructs the open string with the password you specify in this field. The open string in the Properties field should follow this format:

```
openString=Oracle_XA+Acc=P/userName/+SesTm=177+DB=demoPool+Thre
ads=true=Sqlnet=dvi0+logDir=.
```

Note that after the `userName` there is no password.

You can include these passwords in the `Properties` field on the JDBC Connection Pool→Configuration→General tab. However, WebLogic Server displays these passwords in clear text in the Administration Console and in the configuration file (usually `config.xml`). To avoid displaying and storing these passwords in clear text, you can enter the passwords in their respective fields.

The value for Password and Open String Password do not need to be the same. Also, if you use these fields, you should omit the respective values in the Properties field. For example, if you specify a value in the `Password` field, do not include `password=`*`password`* in the Properties field.

Note: Values that you enter in the `Password` and `Open String Password` fields override corresponding values in the `Properties` field. For example, if you enter `tiger` in the `Password` field and you enter `password=smith` in the `Properties` field, WebLogic Server will use `tiger` as the password to make connections to the database.

JDBC Configuration Tasks Using the Command-Line Interface

The following table shows what methods you use to create a dynamic Connection Pool.

Table 16-20 Setting Connectivity—Dynamic

If you want to . . .	Then use the . . .
Create a dynamic Connection Pool	■ Command line—“CREATE_POOL” on page -29, or ■ API—see "Configuring WebLogic JDBC Features" in *Programming WebLogic JDBC*

For more information, see Appendix B, “WebLogic Server Command-Line Interface Reference,” and “Creating a Dynamic Connection Pool” in *Programming WebLogic JDBC*.

Managing and Monitoring Connectivity

Managing connectivity includes enabling, disabling, and deleting the JDBC components once they have been established.

JDBC Management Using the Administration Console

To manage and monitor JDBC connectivity, refer to the following table:

Table 16-21 JDBC Management Tasks

If you want to . . .	Do this . . . in the Administration Console
Reassign a Connection Pool to a Different Server or Cluster	Using the instructions in Assign a Connection Pool to the Servers/Clusters, on the Target tab deselect the target (move target from Chosen to Available) and assign a new target.
Reassign a MultiPool to a Different Cluster	Using the instructions in Assign the MultiPool to Servers or Clusters, on the Target tab deselect the target (move target from Chosen to Available) and assign a new target.
Delete a Connection Pool	See Delete a Connection Pool in the Online Help.
Delete a MultiPool	1. Select the MultiPools node in the left pane. The MultiPools table displays in the right pane showing all the MultiPools defined in your domain. 2. Click the Delete icon in the row of the MultiPool you want to delete. A dialog displays in the right pane asking you to confirm your deletion request. 3. Click Yes to delete the MultiPools. The MultiPool icon under the MultiPools node is deleted.
Delete a Data Source	1. Select the Data Sources node in the left pane. The Data Sources table displays in the right pane showing all the Data Sources defined in your domain. 2. Click the Delete icon in the row of the Data Source you want to delete. A dialog displays in the right pane asking you to confirm your deletion request. 3. Click Yes to delete the Data Source. The Data Source icon under the DataSources node is deleted.

Table 16-21 JDBC Management Tasks

If you want to . . .	Do this . . . in the Administration Console
Monitor a Connection Pool	1. Select the pool in the left pane. 2. Select the Monitoring tab in the right pane, then select the Monitor All Active Pools link.
Modify an Attribute for a Connection Pool, MultiPool, or DataSource	1. Select the JDBC object—Connection Pool, MultiPool, or DataSource—in the left pane. 2. Select the Target tab in the right pane, and unassign the object from each server (move the object from the Chosen column to the Available column.) Then click Apply. This stops the JDBC object—Connection Pool, MultiPool, or DataSource—on the corresponding server(s). 3. Select the tab you want to modify and change the attribute. 4. Select the Target tab and reassign the object to the server(s).This starts the JDBC object—Connection Pool, MultiPool, or DataSource—on the corresponding server(s).

JDBC Management Using the Command-Line Interface

The following table describes the Connection Pool management using the command-line interface. Select the command for more information.

For information on using the Connection Pool commands, see Appendix B, "WebLogic Server Command-Line Interface Reference.".

Table 16-22 Managing Connection Pools with the Command-Line Interface

If you want to . . .	Then use this command . . .
Disable a Connection Pool	**`DISABLE_POOL`**
Enable a Connection Pool that has been disabled	**`ENABLE_POOL`**
Delete a Connection Pool	**`DESTROY_POOL`**

Table 16-22 Managing Connection Pools with the Command-Line Interface

If you want to . . .	Then use this command . . .
Confirm if a Connection Pool was created	`EXISTS_POOL`
Reset a Connection Pool	`RESET_POOL`

Increasing Performance with the Prepared Statement Cache

For each connection pool that you create in WebLogic Server, you can specify a prepared statement cache size. When you set the prepared statement cache size, WebLogic Server stores each prepared statement used in applications and EJBs until it reaches the number of prepared statements that you specify. Statements are cached *per connection*, not per connection pool. For example, if you set the prepared statement cache size to 10, WebLogic Server will store the first 10 prepared statements called by applications or EJBs using that particular connection.

When an application or EJB calls any of the prepared statements stored in the cache, WebLogic Server reuses the statement stored in the cache. Reusing prepared statements eliminates the need for parsing statements in the database, which reduces CPU usage on the database machine, improving performance for the current statement and leaving CPU cycles for other tasks.

The default value for prepared statement cache size is `0`. You can use the following methods to set the prepared statement cache size for a connection pool:

- Using the Administration Console. See JDBC Connection Pool in the *Administration Console Online Help* at `http://e-docs.bea.com/wls/docs61/ConsoleHelp/jdbcconnectionpool.html`.
- Using the WebLogic management API. See the `getPreparedStatementCacheSize()` and `setPreparedStatementCacheSize(int cacheSize)` methods in the Javadocs for WebLogic Classes at

http://e-docs.bea.com/wls/docs61/javadocs/weblogic/management/configuration/JDBCConnectionPoolMBean.html.

- Directly in the configuration file (typically `config.xml`).

To set the prepared statement cache size for a connection pool using the configuration file, before starting the server, open the `config.xml` file in an editor, then add an entry for the `PreparedStatementCacheSize` attribute in the `JDBCConnectionPool` tag. For example:

```
<JDBCConnectionPool CapacityIncrement="5"
    DriverName="com.pointbase.jdbc.jdbcUniversalDriver"
    InitialCapacity="5" MaxCapacity="20" Name="demoPool"
    Password="{3DES}ANfMduXgaaGMeS8+CR1xoA=="
    PreparedStatementCacheSize="20" Properties="user=examples"
    RefreshMinutes="0" ShrinkPeriodMinutes="15"
    ShrinkingEnabled="true" Targets="examplesServer"
    TestConnectionsOnRelease="false"
    TestConnectionsOnReserve="false"
    URL="jdbc:pointbase:server://localhost/demo"/>
```

Usage Restrictions for the Prepared Statement Cache

Using the prepared statement cache can dramatically increase performance, but you must consider its limitations before you decide to use it. Please note the following restrictions when using the prepared statement cache.

There may be other issues related to caching prepared statements that are not listed here. If you see errors in your system related to prepared statements, you should set the prepared statement cache size to `0`, which turns off prepared statement caching, to test if the problem is caused by caching prepared statements.

Calling a Stored Prepared Statement After a Database Change May Cause Errors

Prepared statements stored in the cache refer to specific database objects at the time the prepared statement is cached. If you perform any DDL (data definition language) operations on database objects referenced in prepared statements stored in the cache, the statements will fail the next time you run them. For example, if you cache a

statement such as `select * from emp` and then drop and recreate the `emp` table, the next time you run the cached statement, the statement will fail because the exact `emp` table that existed when the statement was prepared, no longer exists.

Likewise, prepared statements are bound to the data type for each column in a table in the database at the time the prepared statement is cached. If you add, delete, or rearrange columns in a table, prepared statements stored in the cache are likely to fail when run again.

Using setNull In a Prepared Statement

When using the WebLogic jDriver for Oracle to connect to the database, if you cache a prepared statement that uses a `setNull` bind variable, you must set the variable to the proper data type. If you use a generic data type, as in the following example, the statement will fail when it runs with a value other than null.

```
java.sql.Types.Long sal=null

.
.
.

if (sal == null)
    setNull(2,int)//This is incorrect
else
    setLong(2,sal)
```

Instead, use the following:

```
if (sal == null)
    setNull(2,long)//This is correct
else
    setLong(2,sal)
```

This issue occurs consistently when using the WebLogic jDriver for Oracle. It may occur when using other JDBC drivers.

Prepared Statements in the Cache May Reserve Database Cursors

When WebLogic Server caches a prepared statement, the prepared statement may open a cursor in the database. If you cache too many statements, you may exceed the limit of open cursors for a connection. To avoid exceeding the limit of open cursors for a connection, you can change the limit in your database management system or you can reduce the prepared statement cache size for the connection pool.

Determining the Proper Prepared Statement Cache Size

To determine the optimum setting for the prepared statement cache size, you can emulate your server workload in your development environment and then run the Oracle statspack script. In the output from the script, look at the number of parses per second. As you increase the prepared statement cache size, the number of parses per second should decrease. Incrementally increase the prepared statement cache size until the number or parses per second no longer decreases.

Note: Consider the usage restrictions for the prepared statement cache before you decide to use it in your production environment. See "Usage Restrictions for the Prepared Statement Cache" on page 16-33 for more information.

Using a Startup Class to Load the Prepared Statement Cache

To make the best use of the prepared statement cache and to get the best performance, you may want to create a startup class that calls each of the prepared statements that you want to store in the prepared statement cache. WebLogic Server caches prepared statements in the order that they are used and stops caching statements when it reaches the prepared statement cache size limit. By creating a startup class that calls the prepared statements that you want to cache, you can fill the cache with statements that your applications will reuse, rather than with statements that are called only a few times, thus getting the best performance increase with the least number of cached statements. You can also avoid caching prepared statements that my be problematic, such as those described in "Usage Restrictions for the Prepared Statement Cache" on page 16-33.

Even if the startup class fails, WebLogic Server loads and caches the statements for future use.

Note that each connection in effect has it's own cache of statements. If you use a startup class to cache statements, you must create the class in such a way that it gets each connection from the pool and calls the prepared statements that you want to cache on each statement.

If you enable the connection pool to grow as demand for connections increases, new connections will cache statements as the statements are used. The startup class cannot load the prepared statement cache for new connections. If you enable the connection pool to shrink, the connection pool will close connections after the shrink period has been met and connections are available. There is now way to specify which connections to close first. Therefore, the connections for which you loaded the prepared statement cache may close before non-loaded connections close.

CHAPTER

17 Managing JMS

The following sections explain how to manage the Java Message Service (JMS) for WebLogic Server:

- JMS and WebLogic Server
- Configuring JMS
- Monitoring JMS
- Tuning JMS
- Recovering from a WebLogic Server Failure

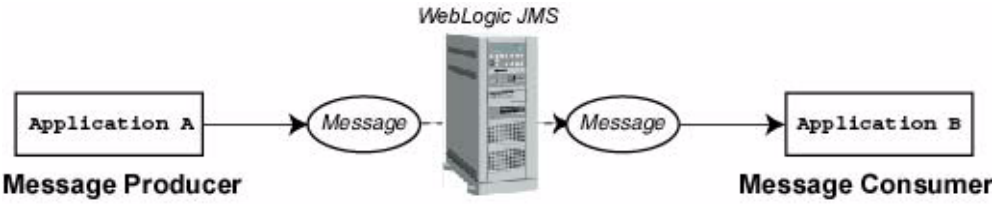
JMS and WebLogic Server

JMS is a standard API for accessing enterprise messaging systems. Specifically, WebLogic JMS:

- Enables Java applications sharing a messaging system to exchange messages.
- Simplifies application development by providing a standard interface for creating, sending, and receiving messages.

The following figure illustrates WebLogic JMS messaging.

Figure 17-1 WebLogic Server JMS Messaging



As illustrated in the figure, WebLogic JMS accepts messages from *producer* applications and delivers them to *consumer* applications.

Configuring JMS

Using the Administration Console, you define configuration attributes to:

- Enable JMS.
- Create JMS servers and target a WebLogic server instance.
- Create and/or customize values for JMS servers, connection factories, destinations (queues and topics), destination templates, destination sort order (using destination keys), persistent stores, session pools, and connection consumers.
- Set up custom JMS applications.
- Define thresholds and quotas.
- Enable any desired JMS features, such as server clustering, concurrent message processing, destination sort ordering, persistent messaging, and message paging.

WebLogic JMS provides default values for some configuration attributes; you must provide values for all others. If you specify an invalid value for any configuration attribute, or if you fail to specify a value for an attribute for which a default does not exist, WebLogic Server will not boot JMS when you restart it. A sample `examplesJMSServer` configuration is provided with the product in the Examples Server. For more information about starting the Examples Server, see "Starting the Default, Examples, and Pet Store Servers" in the *Installation Guide*.

When you migrate WebLogic Server applications from a previous release, the configuration information will be converted automatically, as described in "Migrating Existing Applications" in *Programming WebLogic JMS.*

To configure WebLogic JMS attributes, follow the procedures described in the following sections, or in the *Administration Console Online Help*, to create and configure the JMS objects.

Once WebLogic JMS is configured, applications can send and receive messages using the JMS API. For more information about developing WebLogic JMS applications, refer to "Developing a WebLogic JMS Application" in *Programming WebLogic JMS*.

Note: To assist with your WebLogic JMS configuration planning, *Programming WebLogic JMS* provides configuration checklists for the attribute requirements and/or options that support various JMS features.

JMS Configuration Naming Rules

Each server within a domain must have a name that is unique for all configuration objects in the domain. Within a domain, each server, machine, cluster, virtual host, and any other resource type must be named uniquely and must not use the same name as the domain. This unique naming rule also applies to all configurable JMS objects, such as JMS servers, stores, templates, and connection factories.

The one exception to this unique naming rule, however, is for JMS queue and topic destinations on different JMS servers in a domain, as follows:

- Queue destinations *can* use the same name as other queues on different JMS servers; topic destinations can also use the same name as other topics on different JMS servers.
- Queue destinations *cannot* use the same name with topic destinations, nor can queues nor topics use the same name as any other configurable objects.

Starting WebLogic Server and Configuring JMS

The following sections review how to start WebLogic Server and the Administration console, as well as provide a procedure for configuring a basic JMS implementation.

Starting the Default WebLogic Server

The default role for a WebLogic Server is the Administration Server. If a domain consists of only one WebLogic Server, that server is the Administration Server. If a domain consists of multiple WebLogic Servers, you must start the Administration Server first, and then you start the Managed Servers.

For complete information about starting the Administration Server, see "Starting and Stopping WebLogic Servers" on page 2-1.

Starting the Administration Console

The Administration Console is the Web-based administrator front-end (administrator client interface) to WebLogic Server. You must start the server before you can access the Administration Console for a server.

For complete details about using the Administration Console to configure a WebLogic Server, see "Administration Console" on page 1-4.

Configuring a Basic JMS Implementation

This section describes how to configure a basic JMS implementation using the Administration Console.

1. Under the Services node in the left pane, click the JMS node to expand the list.
2. Optionally, create a File Store for storing persistent messages in a flat file, and/or a Paging Store for swapping messages out to memory:
   a. Click the Stores node in the left pane, and then click the Configure a new JMSFile Store link in the right pane.
   b. On the General tab, give the store a name, specify a directory, and then click the Create button.
   c. Repeat these steps to create a Paging Store.

   Note: For more information on configuring stores, see "Configuring Stores" on page 17-12.
3. Optionally, create a JDBC Store for storing persistent messages in a database:
   a. Click the JDBC node in the left pane to expand it.

b. Click the Connection Pools node in the left pane, and then click the Configure a new JDBC Connection Pool link in the right pane.

c. On the Configuration tabs, set the attributes for the connection pool, such as Name, URL, and database Properties. Click Apply on each tab when you're done making changes.

d. On the Targets tab, target a WebLogic Server instance or a server cluster on which to deploy the connection pool by selecting either the Servers tab or the Clusters tab. Select a target by moving it from the Available list into the Chosen List, and then click Apply.

e. Return to the JMS –> Stores node, and then click the Configure a new JMSJDBCStore link in the right pane.

f. Give the JDBC Store a name, select a connection pool, and a prefix name. Then click Create.

Note: For more information on configuring JDBC connection pools, see "Configuring and Managing JDBC Connection Pools, MultiPools, and DataSources Using the Administration Console" on page 16-24.

4. Optionally, create a JMS Template to define multiple destinations with similar attribute settings. You also need a JMS Template to create temporary queues.

   a. Click the Templates node in the left pane, and then click the Configure a new JMS Template link in the right pane.

   b. On the General tab, give the template a name, and then click Create.

   c. Fill in the Thresholds & Quotas, Override, and Redelivery tabs, as appropriate. Click Apply on each tab when you're done making changes.

   Note: For more information on configuring a JMS Template, see "Configuring JMS Templates" on page 17-11.

5. Configure a JMS Server, as follows:

   a. Click the Server node in the left pane, and then click the Configure a new JMSServer link in the right pane.

   b. On the General tab, give the server a name, select a Store if you created one, select a Paging Store if you created one, and select a Template if you created one. Then click Create.

c. Fill in the Thresholds & Quotas tab, as appropriate. Click Apply when you're done making changes.

d. On the Targets tab, select a WebLogic Server instance on which to deploy the JMS server by moving it from the Available list into the Chosen List, and then clicking Apply.

Note: For more information on configuring a JMS Server, see "Configuring JMS Servers" on page 17-7.

6. Create the JMS Destinations, which are queues (Point-To-Point) or topics (Pub/Sub):

   a. Under the Servers node in the left pane, click your new JMS server instance to expand the list, and then click the Destinations node.

   b. Click either the Configure a new JMSQueue or Configure a new JMSTopic link in the right pane.

   c. On the General tab, give the destination a name and a JNDI name. Fill in the other attributes, as appropriate, and then click Create.

   d. Fill in the Thresholds & Quotas, Override, Redelivery, and Multicast (topics only) tabs, as appropriate. Click Apply on each tab when you're done making changes.

   Note: For more information on configuring a Destinations, see "Configuring Destinations" on page 17-10.

7. Create a Connection Factory to enable your JMS clients to create JMS connections:

   a. Click to the expand the Connection Factory node and in the left pane, and then click the Configure a new JMS Connection Factory link in the right pane.

   b. On the General tab, give the connection factory a name and a JNDI name. Fill in the other attributes, as appropriate, and then click Create.

   c. Fill in the Transactions tab, as appropriate. Click Apply when you're done making changes.

   d. On the Targets tab, target a WebLogic Server instance or a server cluster on which to deploy the connection factory by selecting either the Servers tab or the Clusters tab. Select a target by moving it from the Available list into the Chosen List, and then click Apply.

Note: For more information on configuring a Connection Factory, see "Configuring Connection Factories" on page 17-8.

8. Optionally, use the Destination Keys node to define the sort order for a specific destination. For more information, see "Configuring Destination Keys" on page 17-12.

9. Optionally, create JMS Session Pools, which enable your applications to process messages concurrently, and Connection Consumers (queues or topics) that retrieve server sessions and process messages. For more information, see "Configuring Session Pools" on page 17-16 and "Configuring Connection Consumers" on page 17-16.

Configuring JMS Servers

A JMS server manages connections and message requests on behalf of clients.

To create a JMS server, use the Servers node in the Administration Console and define the following:

- General configuration attributes, including:
  - Name of the JMS server.
  - Persistent store (file or JDBC database) required for persistent messaging. If you do not assign a persistent store for a JMS server, persistent messaging is not supported on that server.
  - Paging store (file recommended) required for paging. If you do not assign a paging store for a JMS server, paging is not support on that server.
  - Temporary template that is used to create all temporary destinations, including temporary queues and temporary topics.

  Note: JMS servers must be uniquely named within a domain. For more information, see "JMS Configuration Naming Rules" on page 17-3.

- Thresholds and quotas for messages and bytes (maximum number, and high and low thresholds), and whether or not bytes paging and/or messages paging is enabled.

- Target a WebLogic Server instance that is associated with a JMS server. When the target WebLogic Server boots, the JMS server boots as well. If no target WebLogic Server is specified, the JMS server will not boot.

Note: The deployment of a JMS server differs from that of a connection factory or template. A JMS server is deployed on a single server. A connection factory or template can be instantiated on multiple servers simultaneously.

For instructions on creating and configuring a JMS server, see "JMS Servers" in the Administration Console Online Help.

Configuring Connection Factories

Connection factories are objects that enable JMS clients to create JMS connections. A connection factory supports concurrent use, enabling multiple threads to access the object simultaneously. You define and configure one or more connection factories to create connections with predefined attributes. WebLogic Server adds them to the JNDI space during startup, and the application then retrieves a connection factory using WebLogic JNDI.

You can establish cluster-wide, transparent access to destinations from any server in the cluster by configuring multiple connection factories and using *targets* to assign them to WebLogic Servers. Each connection factory can be deployed on multiple WebLogic Servers. For more information on JMS clustering, refer to "WebLogic JMS Fundamentals" in *Programming WebLogic JMS*.

To configure connection factories, use the Connection Factories node in the Administration Console to define the following:

- General configuration attributes, including:
  - Name of the connection factory.
  - Name for accessing the connection factory within the JNDI namespace.
  - Client identifier (client ID) that can be used for clients with durable subscribers. (For more information about durable subscribers, see "Developing a WebLogic JMS Application" in *Programming WebLogic JMS*.)
  - Default message delivery attributes (that is, priority, time-to-live, time-to-deliver, and mode).

- Maximum number of outstanding messages that may exist for an asynchronous session and the overrun policy (that is, the action to be taken, for multicast sessions, when this maximum is reached).
- Whether or not the `close()` method is allowed to be called from the `onMessage()` method.
- Whether all messages or only previously received messages are acknowledged.

JMS connection factories must be uniquely named within a domain. For more information, see "JMS Configuration Naming Rules" on page 17-3.

- Transaction attributes—transaction time-out, whether or not Java Transaction API (JTA) user transactions are allowed, and whether or not a transaction (XA) queue or XA topic connection factory is returned.
- Targets (WebLogic Server instances) that are associated with a connection factory to support clustering. Targets enable you to limit the set of servers, groups, and/or clusters on which a connection factory may be deployed.

WebLogic JMS defines one connection factory, by default: `weblogic.jms.ConnectionFactory`. All configuration attributes are set to their default values for this default connection factory. If the default connection factory definition is appropriate for your application, you do not need to configure any additional connection factories for your application.

Note: Using the default connection factory, you have no control over the JMS server on which the connection factory may be deployed. If you would like to target a particular JMS server, create a new connection factory and specify the appropriate JMS server target(s).

For instructions on creating and configuring a connection factory, see "JMS Connection Factories" in the Administration Console Online Help.

Some connection factory attributes are dynamically configurable. When dynamic attributes are modified at run time, the new values become effective for new connections only, and do not affect the behavior of existing connections.

Configuring Destinations

A destination identifies a queue (Point-To-Point) or a topic (Pub/Sub) for a JMS server. After defining a JMS server, configure one or more destination for each JMS server.

You configure destinations explicitly or by configuring a destination template that can be used to define multiple destinations with similar attribute settings, as described in "Configuring JMS Templates" on page 17-11.

To configure destinations explicitly, use the Destinations node in the Administration Console, and define the following configuration attributes:

- General configuration attributes, including:
  - Name and type (queue or topic) of the destination.
  - Name for accessing the destination within the JNDI namespace.
  - Whether or not a store is enabled for storing persistent messages.
  - The JMS template used for creating destinations.
  - Keys used to define the sort order for a specific destination.

  Note: JMS destinations have unique naming restrictions within a domain. For more information, see "JMS Configuration Naming Rules" on page 17-3.
- Thresholds and quotas for messages and bytes (maximum number, and high and low thresholds), and whether or not bytes paging and/or messages paging is enabled on the destination.
- Message attributes that can be overridden (such as priority, time-to-live, time-to-deliver, and delivery mode).
- Message redelivery attributes, including redelivery delay override, redelivery limit, and error destination.
- Multicasting attributes, including multicast address, time-to-live (TTL), and port (for topics only).

For instructions on creating and configuring a destination, see "JMS Destinations" in the Administration Console Online Help.

Some destination attributes are dynamically configurable. When attributes are modified at run time, only incoming messages are affected; stored messages are not affected.

Configuring JMS Templates

A JMS template provides an efficient means of defining multiple destinations with similar attribute settings. JMS templates offer the following benefits:

- You do not need to re-enter every attribute setting each time you define a new destination; you can use the JMS template and override any setting to which you want to assign a new value.
- You can modify shared attribute settings dynamically simply by modifying the template.

To define the JMS template configuration attributes for destinations, use the Templates node in the Administration Console. The configurable attributes for a JMS template are the same as those configured for a destination. These configuration attributes are inherited by the destinations that use them, with the following exceptions:

- If the destination that is using a JMS template specifies an override value for an attribute, the override value is used.
- If the destination that is using a JMS template specifies a message redelivery value for an attribute, that redelivery value is used.
- The Name attribute is not inherited by the destination. This name is valid for the JMS template only. You must explicitly define a unique name for all destinations.

  Note: JMS templates have unique naming restrictions within a domain. For more information, see "JMS Configuration Naming Rules" on page 17-3.

- The JNDI Name, Enable Store, and Template attributes are not defined for JMS templates.
- The Multicast attributes are not defined for JMS templates because they apply only to topics.

Any attributes that are not explicitly defined for a destination are assigned default values. If no default value exists, be sure to specify a value within the JMS template or as a destination attribute override. If you do not do so, the configuration information remains incomplete, the WebLogic JMS configuration fails, and the WebLogic JMS does not boot.

For instructions on creating and configuring a JMS template, see "JMS Templates" in the Administration Console Online Help.

Configuring Destination Keys

Use destination keys to define the sort order for a specific destination.

To create a destination key, use the Destination Keys node in the Administration Console and define the following configuration attributes:

- Name of the destination key
- Property name on which to sort
- Expected key type
- Direction in which to sort (ascending or descending)

For instructions on creating and configuring a destination key, see "JMS Destination Keys" in the Administration Console Online Help.

Configuring Stores

The persistent store consists of a file or database that is used for persistent messaging. To create a file or database store, use the Stores node in the Administration Console and define the following configuration attributes:

- Name of the JMS persistent store.
- For a JMS file store—provide the path to the location where the messages will be saved.
- For a JMS JDBC database store—provide the JDBC connection pool and database table name prefix for use with multiple instances.

  Note: JMS stores have unique naming restrictions within a domain. For more information, see "JMS Configuration Naming Rules" on page 17-3.

Warning: You cannot configure a transaction (XA) connection pool to be used with a JDBC database store For more information, see "JMS JDBC Transactions" on page 17-14.

JMS persistent stores can increase the amount of memory required during initialization of a WebLogic Server instance as the number of stored messages increases. If initialization fails due to insufficient memory when you are rebooting WebLogic Server, increase the heap size of the Java Virtual Machine (JVM) proportionally to the number of messages that are currently stored in the JMS persistent store. Then, try rebooting the server again. For more information on setting heap sizes, see "Tuning WebLogic Server Applications" in the *WebLogic Performance and Tuning Guide*.

For instructions on creating and configuring a store, see "JMS File Stores" and "JMS JDBC Stores" for information about file and JDBC database stores, respectively, in the Administration Console Online Help.

About JMS JDBC Stores

Through the use of JDBC, JMS enables you to store persistent messages in a database, which is accessed through a designated JDBC connection pool. The JMS database can be any database that is accessible through a JDBC driver. WebLogic JMS detects some drivers for the following databases:

- Pointbase
- Microsoft SQL (MSSQL) Server
- Oracle
- Sybase
- Cloudscape
- Informix
- IBM DB2
- Times Ten

The `weblogic/jms/ddl` directory within the `weblogic.jar` file contains JMS DDL files for these databases, which are actually text files containing the SQL commands that create the JMS database tables. To use a different database, simply copy and edit any one of these `.ddl` files.

Note: The JMS samples provided with your WebLogic Server distribution are set up to work with the Cloudscape Java database. An evaluation version of Cloudscape is included with WebLogic Server and a *demoPool* database is provided.

If your existing JMS JDBC stores somehow become corrupted, you can regenerate them using the `utils.Schema` utility. For more information see, "JDBC Database Utility" in *Programming WebLogic JMS*.

JMS JDBC Transactions

You cannot configure a transaction (XA) JDBC connection pool to be used with a JMS JDBC store. JMS must use a JDBC connection pool that uses a non-`XAResource` driver (you cannot use an XA driver or a JTS driver). JMS does the XA support above the JDBC driver.

This is because WebLogic JMS is its own resource manager. That is, JMS itself implements the `XAResource` and handles the transactions without depending on the database (even when the messages are stored in the database). This means that whenever you are using JMS and a database (even if it is the same database where the JMS messages are stored), then it is two-phase commit transaction. For more information about using transactions with WebLogic JMS, see "Using Transactions with WebLogic JMS" in *Programming WebLogic JMS*.

From a performance perspective, you may boost your performance if the JDBC connection pool used for the database work exists on the same WebLogic Server as the JMS queue—the transaction will still be two-phase, but it will be handled with less network overhead. Another performance boost might be achieved by using JMS file stores rather than JMS JDBC stores.

JMS JDBC Security

Optionally, you can restrict the access control list (ACL) for the JDBC connection pool. If you restrict this ACL, you still must include the WebLogic Server *system* user and any user who sends JMS messages in the list. For more information on managing WebLogic Server security, see "Managing Security" on page 14-1.

About JMS Store Table Prefixes

The JMS database contains two system tables that are generated automatically and are used internally by JMS:

- <*prefix*>JMSStore
- <*prefix*>JMSState

The prefix name uniquely identifies JMS tables in the persistent store. Specifying unique prefixes allows multiple stores to exist in the same database. You configure the prefix via the Administration Console when configuring the JDBC store. A prefix is prepended to table names when the DBMS requires fully qualified names, or when you must differentiate between JMS tables for two WebLogic Servers, enabling multiple tables to be stored on a single DBMS.

Warning: No two JMS stores should be allowed to use the same database tables, as this will result in data corruption.

Specify the prefix using the following format, which will result in a valid table name when prepended to the JMS table name:

[[[*catalog*.]*schema*.]*prefix*]JMSStore

where *catalog* identifies the set of system tables being referenced by the DBMS and *schema* translates to the ID of the table owner. For example, in a production database the JMS administrator could maintain a unique table for the Sales department, as follows:

[[[Production.]JMSAdmin.]Sales]JMSStore

Note: For some DBMS vendors, such as Oracle, there is no catalog to set or choose, so this format simplifies to [[schema.]prefix]. For more information, refer to your DBMS documentation for instructions on how to write and use a fully-qualified table name.

Recommended JDBC Connection Pool Settings for JMS Stores

WebLogic Server provides robust JDBC connection pools that can automatically reconnect to failed databases after they come back online, without requiring you to restart WebLogic Server. To take advantage of this capability, and make your use of JMS JDBC stores more robust, configure the following attributes on the JDBC connection pool associated with the JMS JDBC store:

TestConnectionsOnReserve="true"
TestTableName="[[[*catalog*.]*schema*.]*prefix*]JMSState"

Configuring Session Pools

Server session pools enable an application to process messages concurrently. After you define a JMS server, optionally, configure one or more session pools for each JMS server.

Use the Session Pools node in the Administration Console and define the following configuration attributes:

- Name of the server session pool.
- Connection factory with which the server session pool is associated and is used to create sessions.
- Message listener class used to receive and process messages concurrently.
- Transaction attributes (acknowledge mode and whether or not the session pool creates transacted sessions).
- Maximum number of concurrent sessions.

For instructions on creating and configuring a session pool, see "JMS Session Pools" in the Administration Console Online Help.

Some session pool attributes are dynamically configurable, but the new values do not take effect until the session pools are restarted.

Configuring Connection Consumers

Connection consumers are queues (Point-To-Point) or topics (Pub/Sub) that retrieve server sessions and process messages. After you define a session pool, configure one or more connection consumers for each session pool.

To configure connection consumers, use the Session Pools node in the Administration Console to define the following configuration attributes:

- Name of the connection consumer.
- Maximum number of messages that can be accumulated by the connection consumer.

- JMS selector expression used to filter messages. For information about defining selectors, see Developing a WebLogic JMS Application" in *Programming WebLogic JMS*.
- Destination on which the connection consumer will listen.

To create and configure a connection consumer, and for detailed information about each of the connection consumer configuration attributes, see "JMS Connection Consumers" in the Administration Console Online Help.

Monitoring JMS

Using the Administration Console, you can monitor statistics for the following JMS objects: JMS servers, connections, sessions, destinations, message producers, message consumers, server session pools, and durable subscribers.

JMS statistics continue to increment as long as the server is running. Statistics are reset only when the server is rebooted.

Note: For instructions on monitoring JMS connections to WebLogic Server, refer to the Server section in the Administration Console Online Help.

Monitoring JMS Objects

To view JMS monitoring information:

1. Start the Administration Console.
2. Select the JMS node under Services, in the left pane, to expand the list of JMS services.
3. Select the JMS Server node under JMS in the left pane.

   The JMS Servers information is displayed in the right pane.
4. Select the JMS server that you want to monitor from the JMS server list or, from the JMS Servers displayed in the right pane.

5. Select the Monitoring tab to display the monitoring data.

For detailed information about the information being monitored, see the Administration Console Online Help.

Monitoring Durable Subscribers

To view JMS durable subscribers that are running on destination topics:

1. Follow steps 1–3, as described in "Monitoring JMS Objects" on page 17-17.
2. Select the Destinations node under Servers in the left pane, to expand the list of JMS topic and queue destinations.

   The JMS destination information is displayed in a table format in the right pane, with the Durable Subscriber Runtimes column listing the number of durable subscribers running (if any) for the destination topics listed in the table.
3. To view durable subscriber information for a specific topic, click the icon (or actual number) in the Durable Subscriber Runtimes column for the desired topic.

For detailed information about the information being monitored, see the Administration Console Online Help.

Tuning JMS

The following sections explain how to get the most out of your applications by implementing the administrative performance tuning features available with WebLogic JMS.

- Persistent Stores
- Using Message Paging

Persistent Stores

The following sections describe the tuning options available when using persistent stores with WebLogic Server JMS.

Disabling Synchronous Writes to File Stores

By default, WebLogic Server JMS file stores guarantee up-to-the-message integrity by using synchronous writes. Disabling synchronous writes improves file store performance, often quite dramatically, but at the expense of possibly losing sent messages or generating duplicate received messages (even if messages are transactional) in the event of an operating system crash or a hardware failure. Simply shutting down an operating system will not generate these failures, as an operating system flushes all outstanding writes during a normal shutdown. Instead, these failures can be emulated by shutting the power off to a busy server.

Note: At least one JMS vendor disables synchronous writes by default, and this vendor only allows enabling synchronous writes for sends and not for receives.

To disable synchronous writes for all JMS file stores running on a WebLogic server set the following command-line property:

```
-Dweblogic.JMSFileStore.SynchronousWritesEnabled=false
```

To disable synchronous writes for a particular JMS file store:

```
-Dweblogic.JMSFileStore.store-name.SynchronousWritesEnabled=false
```

If both properties are set, the latter command overrides the former. A log message is generated when synchronous writes are disabled. This can be used to verify that the command-line property is taking effect.

Using Message Paging

With the *message paging* feature, you can free up virtual memory during peak message load periods. This feature can greatly benefit applications with large message spaces.

JMS message paging saves memory for both persistent and non-persistent messages, as even persistent messages cache their data in memory. Paged persistent messages continue to be written to the regular backing store (file or database); and paged non-persistent messages are written to the JMS server's messsage paging store, which is configured separately.

A paged-out message does not free all of the memory that it consumes. The message header and message properties remain in memory for use with searching, sorting, and filtering.

IConfiguring Paging

Unless paging is configured and enabled, all messages (even persistent ones) are held in memory. You can configure paging for a new or existing JMS server and/or its destinations through the Administration Console. Using the attributes on the JMS Server node you can specify a paging store for a JMS server, enable bytes and/or messages paging, and configure bytes/messages high and low thresholds to start and stop paging.

Similarly, using the attributes on the Destinations node, you can configure bytes/messages paging for all topics and queues configured on a JMS server. The destinations use the paging store that is configured for the JMS server.

Also, if you use JMS templates to configure multiple destinations, you can use the attributes on the Templates node to configure paging quickly on all your destinations. To override a template's paging configuration for specific destinations, you can enable or disable paging on any destination.

For instructions on configuring a new JMS server, templates, and destinations (Topics or Queues), see "JMS Servers," "JMS Destinations," and "JMS Templates" in the Administration Console Online Help.

Note: For performance tuning purposes, you can modify the paging thresholds to any legal value at any time. Once paging is enabled, however, you cannot dynamically disable it by resetting a byte or message threshold back to -1. To prevent paging from occurring, set the byte/message high threshold to a very large number (maximum is 2^{63} -1), so that paging is not triggered.

Configuring a Paging Store for a JMS Server

Each JMS server must have its own paging store, which is used exclusively for paging out non-persistent messages for the JMS server and its destinations. It's best to use a JMS file store rather than a JMS JDBC store, as the JDBC store will perform poorly in comparison without any real benefit.

To configure a new paging store:

1. Start the Administration Console.
2. Click the JMS Store node. The right pane shows all the JMS stores.
3. Click the Create a new JMS File Store text link. The right pane shows the tabs associated with configuring a new file store.
4. Enter values in the attribute fields.
5. Click Create to create a file store instance with the name you specified in the Name field. The new instance is added under the JMS Stores node in the left pane.
6. If you have multiple JMS servers in your domain, repeat steps 3-5 for each server instance.

Configuring Paging on a JMS Server

To enable and configure paging on an existing JMS server:

1. Click the JMS Servers node. The right pane shows all the servers defined in your domain.
2. Click the server that you want to configure for paging. The right pane shows the tabs associated with configuring the server.
3. On the General tab, use the Paging Store list box to select the store that you configured to store the paged messages. Click Apply to save your changes.

   For instructions on configuring a paging store, refer to "Configuring a Paging Store for a JMS Server" on page 17-21.
4. On the Thresholds & Quotas tab, configure bytes paging:
   - Select the Bytes Paging Enabled check box.

- In the Bytes Threshold High field, enter an amount that will start bytes paging when the number of bytes on the JMS server exceeds this threshold.
- In the Bytes Threshold Low field, enter an amount that will stop bytes paging once the number of bytes on the JMS server falls below this threshold.

5. On the Thresholds & Quotas tab, configure messages paging:
   - Select the Messages Paging Enabled check box.
   - In the Messages Threshold High field, enter an amount that will start messages paging when the number of messages on the JMS server exceeds this threshold.
   - In the Messages Threshold Low field, enter an amount that will stop messages paging once the number of messages on the JMS server falls below this threshold.
6. Click Apply to save the new bytes and/or messages paging values.
7. Repeat steps 2–6 to configure paging for additional JMS servers in the domain.

   Note: Each JMS server must have its own paging store.
8. After you configure your JMS server (or servers) for paging, do one of the following:
   - If you are not configuring a JMS server's destinations for paging, reboot WebLogic Server to activate paging.
   - If you want to configure a server's destinations for paging, follow refer to either "Configuring Paging on a JMS Template" on page 17-22 or "Configuring Paging on Destinations" on page 17-23.

Configuring Paging on a JMS Template

JMS templates provide an efficient way to define multiple destinations (topics or queues) with similar attribute settings. To configure paging on a template for destinations, do the following:

1. Click the JMS node in the left pane.
2. Click the JMS Templates node. The right pane shows all the templates defined in the domain.

3. Click the template that you want to configure for paging. The right pane shows the tabs associated with configuring the template.
4. On the Thresholds & Quotas tab, configure bytes paging:
   - Select the Bytes Paging Enabled check box.
   - In the Bytes Threshold High field, enter an amount that will start bytes paging when the number of bytes on the JMS server exceeds this threshold.
   - In the Bytes Threshold Low field, enter an amount that will stop bytes paging once the number of bytes on the JMS server falls below this threshold.
5. On the Thresholds & Quotas tab, configure messages paging:
   - Select the Messages Paging Enabled check box.
   - In the Messages Threshold High field, enter an amount that will start messages paging when the number of messages on the JMS server exceeds this threshold.
   - In the Messages Threshold Low field, enter an amount that will stop messages paging once the number of messages on the JMS server falls below this threshold.
6. Click Apply to save the new bytes and/or messages paging values.
7. Repeat steps 3–5 to configure paging for additional JMS templates.
8. After configuring all of your JMS templates for paging, reboot WebLogic Server to activate paging.

Configuring Paging on Destinations

Follow these directions if you are configuring paging on destinations without using a JMS template.

1. Under JMS Servers, click to expand a server instance that is already configured for paging.
2. Click the Destinations node. The right pane shows all of the server's topics and queues.
3. Click the topic or queue that you want to configure for paging. The right pane shows the tabs associated with configuring the topic or queue.

4. On the Thresholds & Quotas tab, configure bytes paging:
   - Select the Bytes Paging Enabled check box.
   - In the Bytes Threshold High field, enter an amount that will start bytes paging when the number of bytes on the JMS server exceeds this threshold.
   - In the Bytes Threshold Low field, enter an amount that will stop bytes paging once the number of bytes on the JMS server falls below this threshold.
5. On the Thresholds & Quotas tab, configure messages paging:
   - Select the Messages Paging Enabled check box.
   - In the Messages Threshold High field, enter an amount that will start messages paging when the number of messages on the JMS server exceeds this threshold.
   - In the Messages Threshold Low field, enter an amount that will stop messages paging once the number of messages on the JMS server falls below this threshold.
6. Click Apply to save the new bytes and/or messages paging values.
7. Repeat steps 3–5 to configure paging for additional JMS destinations.
8. After you configure all your destinations for paging, reboot WebLogic Server to activate paging.

Note: If you use JMS templates to configure your destinations, a destination's explicit Byte/Messages Paging configuration overrides the template's configuration. For more information, refer to “Configuring a Destination to Override Paging on a JMS Template” on page 17-24 and to “Configuring JMS” on page 17-2.

Configuring a Destination to Override Paging on a JMS Template

Follow these directions if you want to override a template's settings and enable or disable paging on a specific destination.

1. Under JMS Servers, click to expand a server instance that is already configured for paging.
2. Click the Destinations node. The right pane shows all of the server's topics and queues.

3. Click the topic or queue that you want to configure for paging. The right pane shows the topics or queues associated with the server instance.
4. On the Thresholds & Quotas tab, configure the Bytes Paging Enabled and/or Messages Paging Enabled attributes on the destination according to how you want to override the JMS template for the destination.
   - To disable paging for the destination, select False in the Bytes Paging Enabled and/or the Messages Paging Enabled list boxes.
   - To enable paging for the destination, select True in the Bytes Paging Enabled and/or the Messages Paging Enabled list boxes.
5. Click Apply to save the new bytes and/or messages paging values.
6. Repeat steps 2–5 to configure paging for additional JMS destinations on the same server instance.
7. Once all of your destinations are configured for paging, then reboot WebLogic Server to activate paging.

JMS Paging Attributes

The following sections briefly describe the paging attributes available with WebLogic Server JMS.

JMS Server Paging Attributes

Table 17-1 describes the paging attributes that you define when configuring paging on a JMS Server. For detailed information about other JMS Server attributes, and the valid and default values for them, see "JMS Servers" in the Administration Console Online Help.

Table 17-1 JMS Server Attributes

Attribute	Description
Bytes Paging Enabled	■ If the Bytes Paging Enabled check box is not selected (False), then server bytes paging is explicitly disabled. ■ If the Bytes Paging Enabled check box is selected (True), a paging store has been configured, and both the Bytes Threshold Low and Bytes Threshold High attributes are greater than -1, then server bytes paging is enabled. ■ If either the Bytes Threshold Low or Bytes Threshold High attribute is undefined, or defined as -1, then server bytes paging is implicitly disabled—even though the Bytes Paging Enabled check box is selected (True).
Messages Paging Enabled	■ If the Messages Paging Enabled check box is not selected (False), then server messages paging is explicitly disabled. ■ If the Messages Paging Enabled check box is selected (True), a paging store has been configured, and both the Messages Threshold Low and Messages Threshold High attributes are greater than -1, then server messages paging is enabled. ■ If either the Messages Threshold Low or Messages Threshold High attribute is undefined, or defined as -1, then server paging is implicitly disabled—even though the Messages Paging Enabled check box is selected (True).

Table 17-1 JMS Server Attributes

Attribute	Description
Paging Store	The name of the persistent store where non-persistent messages are paged. A paging store cannot be the same store used for persistent messages or durable subscribers. Two JMS servers cannot use the same paging store; therefore, you must configure a unique paging store for each server.

JMS Template Paging Attributes

Table 17-3 describes the paging attributes that you define when configuring paging on JMS templates for destinations. For detailed information about other JMS template attributes, and the valid and default values for them, see "JMS Templates" in the Administration Console Online Help.

Table 17-2 JMS Template Attributes

Attribute	Description
Bytes Paging Enabled	■ If the Bytes Paging Enabled check box is not selected (False), then destination-level bytes paging is disabled for the JMS template's destinations—unless the destination setting overrides the template. ■ If the Bytes Paging Enabled check box is selected (True), a paging store has been configured for the JMS Server, and both the Bytes Threshold Low and Bytes Threshold High attributes are greater than -1, then destination-level bytes paging is enabled for the JMS template's destinations—unless the destination setting overrides the template. ■ If no value is defined in the JMS Template MBean, then the value defaults to False and bytes paging is disabled for the JMS template's destinations.

Table 17-2 JMS Template Attributes

Attribute	Description
Messages Paging Enabled	■ If the Messages Paging Enabled check box is not selected (False), then destination-level messages paging is disabled for the template's destination—unless the destination setting overrides the template. ■ If the Messages Paging Enabled check box is selected (True), a paging store has been configured for the JMS Server, and both the Messages Threshold Low and Messages Threshold High attributes are greater than -1, then destination-level messages paging is enabled for this destination—unless the destination setting overrides the template. ■ If no value is defined in the JMS Template MBean, then the value defaults to False and messages paging is disabled for the template's destinations.

JMS Destination Paging Attributes

Table 17-3 describes the attributes that you define when configuring paging on destinations. For detailed information about other JMS destination attributes, and valid and default values for them, see "JMS Destinations" in the Administration Console Online Help.

Table 17-3 JMS Destination Attributes

Attribute	Description
Bytes Paging Enabled	■ If Bytes Paging Enabled is set to False, then destination-level bytes paging is disabled for this destination. ■ If Bytes Paging Enabled is set to True, a paging store has been configured for the JMS Server, and both the Bytes Threshold Low and Bytes Threshold High attributes are greater than -1, then destination-level bytes paging is enabled for this destination. ■ If Bytes Paging Enabled is set to Default, then this value inherits the template's value—if a template is specified. If no template is configured for the destination, then the Default value is equivalent to False.
Messages Paging Enabled	■ If Messages Paging Enabled is set to False, then destination-level messages paging is disabled for this destination. ■ If Messages Paging Enabled is set to True, a paging store has been configured for the JMS Server, and both the Messages Threshold Low and Messages Threshold High attributes are greater than -1, then destination-level messages paging is enabled for this destination. ■ If Messages Paging Enabled is set to Default, then this value inherits the template's value—if a template is specified. If no template is configured for the destination, then the Default value is equivalent to False.

Note: If server paging is enabled, and destination-level paging is disabled for a given destination, than messages on the destination can still be paged if server paging is triggered. However, when destination-level paging is disabled for a given destination, then the destination's high thresholds will not force the destination to page out messages when they are exceeded.

Paging Threshold Attributes

Table 17-4 briefly describes the bytes and messages paging thresholds available with JMS servers, templates, and destinations. For detailed information about other JMS server, template, and destination attributes, and the valid and default values for them, see "JMS Servers," "JMS Destinations," and "JMS Templates" in the Administration Console Online Help.

Table 17-4 Paging Threshold Attributes

Attribute	Description
Bytes Threshold High	Start paging when the number of bytes exceeds this threshold.
Bytes Threshold Low	Stop paging when the number of bytes falls back below this threshold.
Messages Threshold High	Start paging when the number of messages exceeds this threshold.
Messages Threshold Low	Stop paging when the number of messages falls back below this threshold.

The thresholds are defined for servers, templates, and destinations as follows:

- If either bytes high/low threshold value is not defined (or is defined as -1), then the number of bytes is not used to determine when and what to page.
- If either messages high/low threshold value is not defined (or is defined as -1), then the number of messages is not used to determine when and what to page.
- A server or template/destination must have the Bytes/Messages Paging Enabled attribute set to True in order for paging to take place. If the thresholds are set, but paging is not enabled, messages are still logged on the server indicating threshold conditions.

Recovering from a WebLogic Server Failure

The following sections describe how to restart or replace a WebLogic Server instance in the event of a system failure, and provide programming considerations for gracefully terminating a JMS application following such an event.

Restarting or Replacing WebLogic Server

When a WebLogic Server fails, you can use one of three methods to perform a system recovery:

- Restart the failed server instance.
- Start up a new server instance using the same IP address as the failed server.
- Start up a new server instance using a different IP address than the failed server.

To restart the failed server instance or start up a new server instance using the *same* IP address as the failed server, boot the server and start the server processes, as described in "Starting and Stopping WebLogic Servers" on page 2-1.

To start up a new server instance using a *different* IP address than the failed server:

1. Update the Domain Name Service (DNS) so that the server alias references the new IP address.
2. Boot the server and start the server processes, as described in "Starting and Stopping WebLogic Servers" on page 2-1
3. Optionally, perform one or more of the tasks listed in the following table:

If your JMS application uses. . .	Perform the following task. . .
Persistent messaging—JDBC Store	■ If the JDBC database store physically exists on the failed server, migrate the database to a new server and ensure that the JDBC connection pool URL attribute reflects the appropriate location reference. ■ If the JDBC database does not physically exist on the failed server, access to the database has not been impacted, and no changes are required.
Persistent messaging—File Store	Migrate the file to the new server, ensuring that the pathname within the WebLogic Server home directory is the same as it was on the original server.
Transactions	Migrate the transaction log to the new server by copying all files named `<servername>*.tlog`. This can be accomplished by storing the transaction log files on a dual-ported disk that can be mounted on either machine, or by manually copying the files. If the files are located in a different directory on the new server, update that server's `TransactionLogFilePrefix` server configuration attribute before starting the new server. **Note:** If migrating following a system crash, it is very important that the transaction log files be available when the server is restarted at its new location. Otherwise, transactions in the process of being committed at the time of the crash might not be resolved correctly, resulting in data inconsistencies. All uncommitted transactions are rolled back.

Note: JMS persistent stores can increase the amount of memory required during initialization of WebLogic Server as the number of stored messages increases. When rebooting WebLogic Server, if initialization fails due to insufficient memory, increase the heap size of the Java Virtual Machine (JVM) proportionally to the number of messages that are currently stored in the JMS persistent store and try the reboot again.

Programming Considerations

You may want to program your JMS application to terminate gracefully in the event of a WebLogic Server failure. For example:

If a WebLogic Server Fails and...	Then...
You are connected to the failed WebLogic Server instance	A `JMSException` will be delivered to the connection exception listener. You must restart the application once the server is restarted or replaced.
You are not connected to the failed WebLogic Server instance	You must re-establish everything once the server is restarted or replaced.
A JMS Server is targeted on the failed WebLogic Server instance	A `ConsumerClosedException` will be delivered to the session exception listener. You must re-establish any message consumers that may have been lost.

CHAPTER

18 Using the WebLogic Messaging Bridge

The following sections explain how to configure and manage a WebLogic Messaging Bridge:

What Is a Messaging Bridge?

The WebLogic Messaging Bridge allows you to configure a forwarding mechanism between any two messaging products—thereby, providing interoperability between separate implementations of WebLogic JMS or between WebLogic JMS and another messaging product. You can use the WebLogic Messaging Bridge to integrate your messaging applications between:

- Any two implementations of WebLogic JMS, including those from separate releases of WebLogic Server.
- WebLogic JMS implementations that reside in separate WebLogic domains.

- WebLogic JMS with a third-party JMS product (for example, MQSeries).
- WebLogic JMS with non-JMS messaging products (only by using specialized adapters that are not provided with WebLogic Server).

A messaging bridge consists of two destinations that are being bridged: a source destination *from which* messages are received, and a target destination *to which* messages are forwarded. For WebLogic JMS and third-party JMS products, a messaging bridge communicates with source and target destinations using the resource adapters provided with WebLogic Server. For non-JMS messaging products, a custom adapter must be obtained from a third-party OEM vendor or by contacting BEA Professional Services in order to access non-JMS source or target destinations.

Source and target bridge destinations can be either queues or topics. For example, messages that are sent to a source topic or queue are automatically forwarded by the messaging bridge to a designated target topic or queue. You can also specify a quality of service (QOS), as well as message filters, transaction semantics, and connection retry policies.

Once a messaging bridge is configured, it is easily managed from the Administration Console, including temporarily suspending bridge traffic whenever necessary, tuning the execute thread pool size to suit your application, and monitoring the status of all your configured bridges.

Messaging Bridge Configuration Tasks

Before you can deploy a messaging bridge, you need to configure its required components:

- "About the Bridge's Resource Adapters" on page 18-3
- "Deploying the Bridge's Resource Adapters" on page 18-5
- "Configuring the Source and Target Bridge Destinations" on page 18-6
- "Configuring a Messaging Bridge Instance" on page 18-12

About the Bridge's Resource Adapters

A messaging bridge uses resource adapters to communicate with the configured source and target JMS destinations. You need to associate both the source and target JMS destinations with a supported adapter in order for the bridge to communicate with them. The JNDI name for the adapter is configured as part of the adapter's deployment descriptor.

Note: Although WebLogic JMS includes a provisional "General Bridge Destination" framework for accessing non-JMS messaging products, WebLogic Server does not provide supported adapters for such products. Therefore, you must obtain a custom adapter from a third-party OEM vendor and consult their documentation for configuration instructions. You could also contact BEA Professional Services more information about obtaining a custom non-JMS adapter.

The supported adapters are located in the *WL_HOME*\lib directory and are described in the following table.

Table 18-1 Messaging Bridge Adapters and JNDI Names

Adapter	JNDI Name	Description
jms-xa-adp.rar	eis.jms.WLSConnectionFactoryJNDIXA	Provides transaction semantics via XAResource. Used when the required QOS is *Exactly-once*. This envelops a received message and sends it within a user transaction (XA/JTA). The following requirements are necessary in order to use this adapter: ■ Any WebLogic Server implementation being bridged must be release 6.1 or later. ■ The source and target JMS connection factories must be configured to use the XAConnectionFactory. **Note:** Before deploying this adapter, refer to the "Using the Messaging Bridge to Interoperate with Different WebLogic Server Versions and Domains" on page 18-18 for specific transactional configuration requirements and guidelines.
jms-notran-adp.rar	eis.jms.WLSConnectionFactoryJNDINoTX	Provides no transaction semantics. Used when the required QOS is *Atmost-once or Duplicate-okay*. If the requested QOS is *Atmost-once*, the adapter uses the AUTO_ACKNOWLEDGE mode. If the requested QOS is *Duplicate-okay*, CLIENT_ACKNOWLEDGE is used. **Note:** For more information about the acknowledge modes used in non-transacted sessions, see "WebLogic JMS Fundamentals" in *Programming WebLogic JMS*.

Table 18-1 Messaging Bridge Adapters and JNDI Names

Adapter	JNDI Name	Description
`jms-notran-adp51.rar`	`eis.jms.WLS51ConnectionFactoryJNDINoTX`	Provides interoperability when either the source or target destination is WebLogic Server 5.1. This adapter provides no transaction semantics; therefore, it only supports a QOS of *Atmost-once* or *Duplicate-okay*. If the requested QOS is *Atmost-once*, the adapter uses the `AUTO_ACKNOWLEDGE` mode. If the requested QOS is *Duplicate-okay*, `CLIENT_ACKNOWLEDGE` is used.

You will specify the appropriate adapter by its JNDI name when you configure each source and target bridge destination on the Administration Console.

Deploying the Bridge's Resource Adapters

Before you configure the messaging bridge components, deploy the appropriate resource adapters in the WebLogic Server domain that is hosting the messaging bridge, using either of the following methods:

- On the Administration Console — Select the Domain in which you will be deploying the adapters in, and then select Deployments → Applications option, and then select the appropriate `RAR` adapter file, as defined in "Messaging Bridge Adapters and JNDI Names" on page 18-4.
  - `jms-xa-adp.rar`
  - `jms-notran-adp.rar`
  - `jms-notran-adp51.rar`
- Using the Auto Deployment feature — This method is used for quickly deploying an application on the administration server. By copying the adapters to the local `\applications` directory of the administration server, they will be automatically deployed if the server is already running. Otherwise, they will be deployed the next time you start WebLogic Server. The auto deployment method is used only in a single-server development environment for testing an application, and is not recommended for use in production mode.

Note: When configuring a messaging bridge to interoperate between WebLogic Server release 6.1 and release 5.1, then the release 5.1 resource adapter (`jms-notran-adp51.rar`) and the non-transaction adapter (`jms-notran-adp.rar`) must be deployed on the 6.1 domain running the messaging bridge.

For step-by-step instructions on deployment tasks using the Administration Console, or for more information on using the auto-deployment feature, see "Deploying Applications" on page 7-1.

Configuring the Source and Target Bridge Destinations

A messaging bridge connects two actual destinations that are mapped to bridge destinations: a source destination *from which* messages are received, and a target destination *to which* messages are sent. Depending on the messaging products that need to be bridged, there are two types of bridge destinations:

- JMS Bridge Destination – For JMS messaging products, whether it is a WebLogic JMS implementation or a third-party JMS provider, you need to configure a `JMSBridgeDestination` instance for each actual source and target JMS destination being mapped to a messaging bridge.
- General Bridge Destination – For non-JMS messaging products, you need to configure a generic `BridgeDestination` instance for each actual source and target destination being mapped to a messaging bridge.

Before starting the procedure in this section, refer to the "Using the Messaging Bridge to Interoperate with Different WebLogic Server Versions and Domains" on page 18-18 for specific configuration requirements and guidelines.

Configuring JMS Bridge Destinations

A `JMSBridgeDestination` instance defines a unique name for the actual JMS queue or topic destination within a WebLogic domain, the name of the resource adapter used to communicate with the specified destination, property information to pass to the adapter (Connection URL, Connection Factory JNDI Name, etc.), and, optionally, a user name and password.

You need to configure a `JMSBridgeDestination` instance for each actual source and target JMS destination to be mapped to a messaging bridge. Therefore, when you finish defining attributes for a source JMS bridge destination, repeat these steps to configure a target JMS bridge destination, or vice versa. You will designate the source and target JMS Bridge Destinations in "Configuring a Messaging Bridge Instance" on page 18-12.

To configure a JMS bridge destination, follow these steps.

1. In the Administration Console, click the Messaging Bridge node.
2. Click the JMS Bridge Destinations node to open the Bridge Destinations tab in the right pane.
3. In the right pane, click the Configure a new JMS Bridge Destination link. A Configuration dialog displays in the right pane showing the tabs associated with configuring a new JMS bridge destination.
4. Define the attributes in the Configuration tab.

   The following table describes the attributes you set on the Configuration tab.

Table 18-2 JMS Bridge Destination Attributes on the Configuration Tab

Attribute	Description
Name	A JMS bridge destination name for the actual JMS destination being mapped to the bridge. This name must be unique across a WebLogic domain. For example, if you are bridging between WebLogic Server releases 6.1 and 7.0, for the source destination you could change the default bridge destination name to "61to70SourceDestination". Then, when you create the corresponding target destination, you could name it as "61to70TargetDestination". Once the bridge destinations are configured, these names are listed as options in the Source Destination and Target Destination attributes on the Bridges Æ General tab.
Adapter JNDI Name	The JNDI name of the resource adapter used to communicate with the bridge destinations. For more information on which adapter name to enter, see "Messaging Bridge Adapters and JNDI Names" on page 18-4.

Table 18-2 JMS Bridge Destination Attributes on the Configuration Tab

Attribute	Description
Adapter Classpath	When connecting to a destination that is running on WebLogic Server 6.0 or earlier, the bridge destination must supply a `CLASSPATH` that indicates the locations of the classes for the earlier WebLogic Server implementation. When connecting to a third-party JMS provider, the bridge destination must supply the provider's `CLASSPATH` in the WebLogic Server `CLASSPATH`.
Connection URL	The URL of the JNDI provider used to look up the connection factory and destination.
Initial Context Factory	The factory used to get the JNDI context.
Connection Factory JNDI Name	The JMS connection factory used to create a connection for the actual JMS destination being mapped to the JMS bridge destination. **Note:** In order to use the *Exactly-once* QOS, the connection factory has to be a XAConnection Factory. For more information about connection factory and QOS requirements, see "Messaging Bridge Attributes on the General Tab" on page 18-12.
Destination JNDI Name	The JNDI name of the actual JMS destination being mapped to the JMS bridge destination.
Destination Type	Select either a Queue or Topic destination type.
User Name and Password	The user name and password that the messaging bridge will give to the bridge adapter. **Note:** All operations done to the specified destination are done using that user name and password. Therefore, the User Name/Password for the source and target destinations must have permission to access the underlying JMS destinations in order for the messaging bridge to work.

5. Click Create to create the JMS bridge destination.

6. When you finish defining attributes for a source JMS bridge destination, repeat these steps to configure a target JMS bridge destination, or vice versa.

Configuring General Bridge Destinations

A general `BridgeDestination` instance defines a unique name for the actual queue or topic destination within the domain, the name of the resource adapter used to communicate with the specified destination, a list of properties to pass to the adapter, and, optionally, a user name and password.

Note: Although WebLogic JMS includes a provisional "General Bridge Destination" framework for accessing non-JMS messaging products, WebLogic Server does not provide supported adapters for such products. Therefore, you must obtain a custom adapter from a third-party OEM vendor and consult their documentation for configuration instructions. You could also contact BEA Professional Services more information about obtaining a custom non-JMS adapter.

You need to configure a `BridgeDestination` instance for each actual source and target destination to be mapped to a messaging bridge. Therefore, when you finish defining attributes for a source general bridge destination, repeat these steps to configure a target general bridge destination, or vice versa. You will designate the source and target general Bridge Destinations in "Configuring a Messaging Bridge Instance" on page 18-12.

To configure a general bridge destination, follow these steps:

1. In the Administration Console, click the Messaging Bridge node.
2. Click the General Bridge Destinations node to open the Bridge Destinations tab in the right pane.
3. In the right pane, click the Configure a new General Bridge Destination link. A Configuration dialog displays in the right pane showing the tabs associated with configuring a new general bridge destination.
4. Define the attributes in the Configuration tab.

   The following table describes the attributes you set on the Configuration tab.

Table 18-3 General Bridge Destination Attributes on the Configuration Tab

Attribute	Description
Name	A bridge destination name for the actual destination being mapped to the bridge. This name must be unique across a WebLogic domain. For example, if you are bridging between WebLogic Server releases 6.1 and 7.0, for the source destination you could change the default bridge destination name to "61to70SourceDestination". Then, when you create the corresponding target destination, you could name it as "61to70TargetDestination". Once the bridge destinations are configured, these names are listed as options in the Source Destination and Target Destination attributes on the Bridges → General tab.
Adapter JNDI Name	A bridge destination must supply the JNDI name of the adapter used to communicate with the bridge destinations. WebLogic Server does not provide adapters for non-JMS messaging products. Therefore, you must use a specialized adapter from a third-party OEM vendor, or contact BEA Professional Services to obtain a custom adapter.
Adapter Classpath	Defines the `CLASSPATH` of the bridge destination. This is attribute is mainly used to connect to a destination running on WebLogic Server 6.0 or earlier. When connecting to a third-party product, you must supply the product's `CLASSPATH` in the WebLogic Server `CLASSPATH`.

Table 18-3 General Bridge Destination Attributes on the Configuration Tab

Attribute	Description
Properties	Specifies all the properties defined for a bridge destination. Each property must be separated by a semicolon (for example, `DestinationJNDIName=myTopic;DestinationType=topic;`). For non-JMS messaging products that use adapters provided by a third-party OEM vendor, you should consult the vendor's documentation for property configuration instructions. The following properties are required for all JMS implementations: `ConnectionURL`= URL used to establish a connection to the destination. `InitialContextFactory=` Factory used to get the JNDI context. `ConnectionFactoryJNDIName`= The JMS connection factory used to create a connection for the actual JMS destination being mapped to the JMS bridge destination. `DestinationJNDIName`= The JNDI name of the actual JMS destination being mapped to the JMS bridge destination. `DestinationType`= Queue or topic.
User Name and Password	The user name that the messaging bridge will give to the bridge adapter. **Note:** All operations done to the specified destination are done using this user name and password. Therefore, the User Name/Password for the source and target bridge destinations must have permission to access the underlying source and target destinations in order for the messaging bridge to work.

5. Click Create to create the general bridge destination.
6. When you finish defining attributes for a source general bridge destination, repeat these steps to configure a target general bridge destination, or vice versa.

Configuring a Messaging Bridge Instance

A messaging bridge instance communicates with the configured source and target bridge destinations. For each mapping of a source bridge destination to a target bridge destination, whether it is another WebLogic JMS implementation, a third-party JMS provider, or another non-JMS messaging product, you must configure a `MessagingBridge` instance via the Administration Console. Each `MessagingBridge` instance defines the source and target destination for the mapping, a message filtering selector, a QOS, transaction semantics, and various reconnection parameters.

Before starting the procedure in this section, refer to the "Using the Messaging Bridge to Interoperate with Different WebLogic Server Versions and Domains" on page 18-18 or "Bridging to a Third-Party Messaging Provider" on page 18-23 for specific configuration requirements and guidelines.

To configure a messaging bridge, follow these steps:

1. In the Administration Console, click the Messaging Bridge node.
2. Click the Bridges node to open the Bridges tab in the right pane.
3. Click the Configure a new Messaging Bridge link in the right pane. A Configuration dialog displays in the right pane showing the tabs associated with configuring a new messaging bridge.
4. Define the attributes in the General tab.

   The following table describes the attributes you set on the General tab.

Table 18-4 Messaging Bridge Attributes on the General Tab

Attribute	Description
Name	Enter a name for the messaging bridge that is unique across a WebLogic domain.
Source Destination	Select the source destination *from which* messages are received by the messaging bridge. For example, for a JMS messaging bridge, you should select the "JMS Source Bridge Destination" name that you created on the JMS Bridge Destination → Configuration tab.

Table 18-4 Messaging Bridge Attributes on the General Tab

Attribute	Description
Target Destination	Select the target destination *to which* messages are sent from the messaging bridge. For example, for a JMS messaging bridge, you should select the "JMS Target Bridge Destination" name that you created on the JMS Bridge Destination → Configuration tab.
Selector	Allows you to filter the messages that are sent across the messaging bridge. Only messages that match the selection criteria are sent across the messaging bridge. For queues, messages that do not match the selection criteria are left behind and accumulate in the queue. For topics, messages that do not match the connection criteria are dropped. For more information on using selectors to filter messages, see "Developing a WebLogic JMS Application" in *Programming WebLogic JMS*.
Quality Of Service (QOS)	Select a QOS guarantee for forwarding a message across a messaging bridge. The valid qualities of service are: *Exactly-once*—Each message will be sent exactly once. This is the highest quality of service. In order to use this QOS: ■ Any WebLogic Server implementation must be release 6.1 or later. ■ The source and target JMS connection factories must be configured to use the `XAConnectionFactory`. ■ The transaction `jms-xa-adp.rar` adapter must be deployed and identified in the Adapter JNDI Name attribute as "*`eis.jms.WLSConnectionFactoryJNDIXA`*" for both the source and target destinations. *Atmost-once*—Each message is sent at most one time. Some messages may not be delivered to the target destination. *Duplicate-okay*—Each message is sent at least one time. Duplicate messages can be delivered to the target destination.

Table 18-4 Messaging Bridge Attributes on the General Tab

Attribute	Description
QOS Degradation Allowed	When selected, the messaging bridge automatically degrades the requested QOS when the configured one is not available. If this occurs, a message is delivered to the WebLogic startup window (or log file). If this option is not selected (false), and the messaging bridge cannot satisfy the requested QOS, it will result in an error and the messaging bridge will not start.
Maximum Idle Time (seconds)	For bridges running in asynchronous mode, this is the maximum amount of time, in seconds, the messaging bridge sits idle before checking the health of its connections. For bridges running in synchronous mode, this dictates the amount of time the messaging bridge can block on a receive call if no transaction is involved.
Asynchronous Mode Enabled	Defines whether a messaging bridge works in asynchronous mode. Messaging bridges that work in asynchronous mode (true) are driven by the source destination. The messaging bridge listens for messages and forwards them as they arrive. When the value is false, the bridge works in synchronous mode even if the source supports asynchronous receiving. **Note:** For a messaging bridge with a QOS of *Exactly-once* to work in asynchronous mode, the source destination has to support the `MDBTransaction` interface described in the `weblogic.jms.extensions` Javadoc. Otherwise, the bridge automatically switches to synchronous mode if it detects that `MDBTransactions` are not supported by the source destination. For more information about `MDBTransactions`, see "Using Message-Driven Beans" in *Programming WebLogic Enterprise JavaBeans*.

Table 18-4 Messaging Bridge Attributes on the General Tab

Attribute	Description
Durability Enabled	This attribute is used only for JMS topics or for destinations with similar characteristics as a JMS topic. By enabling durability, a messaging bridge creates a durable subscription for the source destination. This allows the source JMS implementation to save messages that are sent to it when the bridge is not running. The bridge will then forward these messages to the target destination once it is restarted. If this attribute is not selected, messages that are sent to the source JMS topic while the bridge is down cannot be forwarded to the target destination. **Note:** If the messaging bridge is running on WebLogic Server 6.1, it cannot support durable subscribers when the source destination is a JMS topic running on WebLogic Server 7.0. This issue has been resolved for messaging bridges running on WebLogic Server 7.0. **Note:** If a bridge must be taken permanently offline, you must delete any durable subscriptions that use the bridge. For information on deleting durable subscribers, see "Deleting Durable Subscriptions" in *Programming WebLogic JMS*.
Started	Indicates the initial state of the messaging bridge when it is configured and whenever the server is restarted. You can also use this field to dynamically start and stop the messaging bridge. To stop the bridge, clear the check box. Conversely, reselect the check box to restart the bridge. **Note:** Unless there is a configuration issue that prevents the messaging bridge from starting, this field indicates the expected run-time state of the messaging bridge. For information on monitoring all the configured messaging bridges in your domain, see "Monitoring Messaging Bridges" on page 18-25.

5. Click Create to create the messaging bridge.
6. On the Connection Retry tab, define the bridge's reconnection time intervals according to the following table.

The source and target destinations for a messaging bridge will not always be available. As such, the messaging bridge must be able to reconnect to the destination at some periodic interval. These attributes govern the time between reconnection attempts.

Table 18-5 Messaging Bridge Attributes on the Connection Retry Tab

Attribute	Description
Minimum Delay (seconds)	The minimum delay, in seconds, between reconnection attempts. When a messaging bridge boots and cannot connect to a destination, or a connection is lost and the messaging bridge is first attempting to reconnect, it attempts to reconnect in this specified amount of seconds.
Incremental Delay (seconds)	The delay increment, in seconds, between reconnection attempts. Each time a bridge fails to reconnect, it adds this amount of seconds to the delay before making its next reconnection attempt.
Maximum Delay (seconds)	The maximum delay, in seconds, between reconnection attempts. Each reconnection attempt is delayed further by the Incremental Delay amount of seconds, but it is never delayed by more than this value.

7. Click Apply to store new attribute values.
8. On the Transactions tab, define the transaction attributes for the messaging bridge according to the following table.

Table 18-6 Messaging Bridge Attributes on the Transactions Tab

Attribute	Description
Transaction Timeout	Defines the number of seconds the transaction manager waits for each transaction before timing it out. Transaction timeouts are used when a bridge's quality of service requires two-phase transactions.
Batch Size	Defines the number of messages that the messaging bridge transfers within one transaction. Batch Size only applies to bridges that work in synchronous mode and whose quality of service require two-phase transactions.

Table 18-6 Messaging Bridge Attributes on the Transactions Tab

Attribute	Description
Batch Interval (milliseconds)	Defines the maximum time, in milliseconds, that the bridge waits before sending a batch of messages in one transaction, regardless of whether the Batch Size amount has been reached or not. The default value of -1 indicates that the bridge will wait until the number of messages reaches the *Batch Size* before it completes a transaction. Batch Interval only applies to bridges that work in synchronous mode and whose quality of service require two-phase transactions.

9. Click Apply to store new attribute values.
10. On the Targets tab, assign WebLogic Server instances to associate with the messaging bridge according to the following table.

Table 18-7 Messaging Bridge Attributes on the Targets Tab

Attribute	Description
Clusters	Defines a WebLogic Server cluster where the messaging bridge will be deployed. The messaging bridge will be available on all servers in the selected cluster.
Servers	Defines the WebLogic Servers where the messaging bridge will be deployed. The messaging bridge will be available on all the selected WebLogic Servers.

11. Click Apply to store new attribute values.

Using the Messaging Bridge to Interoperate with Different WebLogic Server Versions and Domains

The following interoperability guidelines apply when using the messaging bridge to access JMS destinations in different releases of WebLogic Server and in other WebLogic Server domains.

Note: When the messaging bridge is used to communicate between two domains running different releases of Weblogic Server, a best-practice recommendation is for the messaging bridge to be configured to run on the domain using the latest release of Weblogic Server.

Naming Guidelines for WebLogic Servers and Domains

Unique naming rules apply to all WebLogic Server deployments if more than one domain is involved. Therefore, make sure that:

- WebLogic Server instances and domain names are unique.
- WebLogic JMS server names are unique name across domains.
- All JMS connection factories targeted to servers in a cluster are uniquely named.

- If a JMS file store is being used for persistent messages, the JMS file store name must be unique across domains.

Enabling Security Interoperability for WebLogic Domains

Follow these security guidelines when a release 6.1 domain is interoperating with another release 7.0 or later domain:

1. The release 7.0 or later Credential password must *exactly match* the "system" user password configured for the 6.1 domain. Also, make sure that "system" user is a member of the Administrators group in the 7.0 domain.
2. Configure the release 7.0 or later security interoperability as follows:
   a. Expand the Domains node (for example, Examples).
   b. Select the Security → Advanced tab.
   c. Clear the Enable Generated Credential check box, if necessary.
   d. Click the Credential: Change attribute to open the Change Credential window.
   e. In the New Credential field, specify a password for the domain. This password must match the password used for the domain that you are interoperating with.
   f. Confirm the password by re-entering it in the Retype to confirm field.
   g. Click Apply.

   Note: For more information about release 6.1 domain interoperability security, see "Using Compatibility Security" in *Managing WebLogic Security*. For more information about WebLogic Server 7.0 domain interoperability security, see "Enabling Trust Between WebLogic Domains" in *Managing WebLogic Security*.

Using the Messaging Bridge To Access Destinations In a Release 6.1 or Later Domain

Use these guidelines when configuring a messaging bridge on a release 6.1 domain to provide "Exactly-once" transactional message communication between two release 6.1 or later domains.

Note: The *Exactly-once* quality of service for transactions is only supported for implementations of WebLogic Server 6.1 or later.

- A messaging bridge running on release 6.1 cannot support durable subscribers when the source destination is a JMS destination topic running on release 7.0. Therefore, when configuring a messaging bridge on release 6.1, disable the Durability Enabled attribute on the Messaging Bridge → Configuration → General tab. This issue has been resolved for messaging bridges running on release 7.0 or later.
- If a JMS file store is being used for persistent messages, the JMS file store name must be unique across WebLogic domains, as described in "Naming Guidelines for WebLogic Servers and Domains" on page 18-18.
- Make sure that security interoperability between the domains is correctly configured, as described in "Enabling Security Interoperability for WebLogic Domains" on page 18-19.
- Make sure that the transaction connection factory is enabled for both domains by selecting the User Transactions Enabled and XAConnection Factory Enabled check boxes on the Services → JMS → Connection Factories → Configuration → Transactions tab.
- Deploy the transaction resource adapter, `jms-xa-adp.rar`, on the 6.1 bridge domain, as described in "Deploying the Bridge's Resource Adapters" on page 18-5.
- When configuring the JMS bridge destinations, as described in "Configuring JMS Bridge Destinations" on page 18-6, do the following for both the source and target bridge destinations:
  - In the Adapter JNDI Name field, identify the transaction adapter's JNDI name, `eis.jms.WLSConnectionFactoryJNDIXA`.
  - Do not enter anything in the Adapter Classpath field.

- On the Messaging Bridge → Configuration → General tab, select a Quality Of Service of *Exactly-once*, as described in "Configuring a Messaging Bridge Instance" on page 18-12.

Using the Messaging Bridging To Access Destinations In a Release 6.0 Domain

When configuring a messaging bridge involves interoperability between WebLogic Server 6.1 and a release 6.0 domain, you must configure the following settings on the release 6.1 domain that the bridge is running on:

Note: The *Exactly-once* QOS for transactions is not supported for WebLogic Server 6.0. For more information on the bridge QOS options, see "Messaging Bridge Attributes on the General Tab" on page 18-12.

- Deploy the non-transaction resource adapter, `jms-notran-adp.rar` on the 6.1 bridge domain, as described in "Deploying the Bridge's Resource Adapters" on page 18-5.
- When configuring the JMS source and target destinations, as described in "Configuring JMS Bridge Destinations" on page 18-6, do the following:

  In the Adapter JNDI Name field:

  - For the source and target destinations, specify the non-transaction adapter's JNDI name as `eis.jms.WLSConnectionFactoryJNDINoTX`.

  In the Adapter Classpath field:

  - For the 6.1 destination, leave the field blank.
  - For the 6.0 destination, indicate the location of the classes for the WebLogic Server 6.0 release.

    For example, if you have WebLogic Server 6.0 GA installed in a directory named *`WL60_HOME`*, then set the Adapter Classpath as follows for the 6.0 JMS bridge destination:

    `WL60_HOME``\lib\weblogic60.jar`

- On the Messaging Bridge → Configuration → General tab, select a Quality Of Service of *Atmost-once* or *Duplicate-okay*, as described in "Configuring a Messaging Bridge Instance" on page 18-12.

Using the Messaging Bridging To Access Destinations In a Release 5.1 Domain

When configuring a messaging bridge involves interoperability between WebLogic Server 6.1 and release 5.1, you must configure the following settings on the release 6.1 domain that the messaging bridge is running on:

Note: The *Exactly-once* QOS for transactions is not supported for WebLogic Server 5.1. For more information on the bridge QOS options, see "Messaging Bridge Attributes on the General Tab" on page 18-12.

- The `jms51-interop.jar` file in the *`WL_HOME`*`\lib` directory must be in the `CLASSPATH` of the WebLogic Server 6.1 implementation.
- The release 5.1 resource adapter (`jms-notran-adp51.rar`) and the non-transaction adapter (`jms-notran-adp.rar`) must be deployed on the 6.1 bridge domain, as described in "Deploying the Bridge's Resource Adapters" on page 18-5.
- When configuring the JMS source and target destinations, as described in "Configuring JMS Bridge Destinations" on page 18-6, do the following:

  In the Adapter JNDI Name field:

  - For the 6.1 destination, specify the non-transaction adapter's JNDI name as `eis.jms.WLSConnectionFactoryJNDINoTX`.
  - For the 5.1 destination, specify the 5.1 adapter's JNDI name as `eis.jms.WLS51ConnectionFactoryJNDINoTX`.

  In the Adapter Classpath field:

  - For the 6.1 destination, leave the field blank.
  - For the 5.1 destination, indicate the location of the classes for the WebLogic Server 5.1 release, as well as the location of the `jms51-interop.jar` file for the 6.1 release.

    For example, if you have WebLogic Server 5.1 GA installed in a directory named *`WL51_HOME`* and your WebLogic Server 6.1 release is installed in *`WL81_HOME`*, then set the Adapter Classpath as follows for the 5.1 destination:

    `WL51_HOME``\classes;`*`WL51_HOME`*`\lib\`*`weblogicaux.jar`*`;`
    `WL61_HOME``\server\lib\jms51-interop.jar`

Note: If your implementation is using a 5.1 Service Pack, the corresponding `sp.jar` files must also be added to the Adapter Classpath field.

- On the Messaging Bridge → Configuration → General tab, select a Quality Of Service of *Atmost-once* or *Duplicate-okay*, as described in "Configuring a Messaging Bridge Instance" on page 18-12.

Bridging to a Third-Party Messaging Provider

When configuring a messaging bridge involves interoperability with a third-party messaging provider, you must configure the following:

- Before starting WebLogic Server:
  - Supply the provider's `CLASSPATH` in the WebLogic Server `CLASSPATH`.
  - Include the `PATH` of any native code required by the provider's client-side libraries in the WebLogic Server system `PATH`. (This variable may vary depending on your operating system.)
- In the `JMSBridgeDestination` instance for the third-party messaging product being bridged, provide *vendor-specific* information in the following attributes:
  - Connection URL
  - Initial Context Factory
  - Connection Factory JNDI Name
  - Destination JNDI Name

  For more information on configuring the remaining attributes for a JMS Bridge Destination, see "Configuring JMS Bridge Destinations" on page 18-6.

Note: The messaging bridge cannot provide the "Exactly-once" quality of service when the source and target bridge destinations are located on the same resource manager (that is, when the bridge is forwarding a global transaction

that is using the XA resource of the resource manager). For example, when using MQ Series, it is not possible to use the same Queue Manager for the source and target bridge destinations.

Managing a Messaging Bridge

Once a messaging bridge is up and running, it can managed from the Administration Console.

- Stopping and Restarting a Messaging Bridge
- Monitoring Messaging Bridges
- Configuring the Execute Thread Pool Size

Stopping and Restarting a Messaging Bridge

To temporarily suspend and restart an active messaging bridge:

1. Click to expand the Messaging Bridge node.
2. Select the messaging bridge instance that you want to stop.
3. On the Configuration General tab, clear the Started check box to stop the bridge.
4. To restart the bridge, select the Started check box.

Monitoring Messaging Bridges

You can monitor the status of all the messaging bridges in your domain from the Administration Console:

1. Expand the Servers node.
2. Select the server where the messaging bridges are configured. A dialog displays in the right pane showing the tabs associated with the selected server instance.
3. Select the Services tab.
4. Select the Bridge tab.

5. Click the Monitoring all Messaging Bridge Runtimes text link to display the monitoring data.
6. A table displays showing all the messaging bridge instances for the server and their status (either as running or not running).

Configuring the Execute Thread Pool Size

You can configure the default execute thread pool size for your messaging bridges from the Administration Console. For example, you may want to increase the default size to reduce competition from the WebLogic Server default thread pool. Entering a value of -1 disables this thread pool and forces a messaging bridge to use the WebLogic Server default thread pool.

1. Click the Servers node in the left pane to expand it.
2. Select the server instance where the messaging bridge is configured. A dialog displays in the right pane showing the tabs associated with the selected server instance.
3. Select the Services tab.
4. Select the Bridge tab.
5. Enter a new value in the Messaging Bridge Thread Pool Size field.
6. Click Apply to save your changes.

For more information about tuning execute threads, see "Tuning WebLogic Server Applications" in the *Performance and Tuning Guide*.

CHAPTER

19 Managing JNDI

The following sections describe how to manage JNDI:

Overview of JNDI Management

You use the Administration Console to manage JNDI. The JNDI API enables applications to look up objects—such as Data Sources, EJBs, JMS, and MailSessions—by name. The JNDI tree is represented by the left pane in the Administration Console.

For additional information, see Programming WebLogic JNDI at http://e-docs.bea.com/wls/docs61/jndi/index.html.

What Do JNDI and Naming Services Do?

JNDI provides a common-denominator interface to many existing naming services, such as LDAP (Lightweight Directory Access Protocol) and DNS (Domain Name System). These naming services maintain a set of bindings, which relate names to objects and provide the ability to look up objects by name. JNDI allows the components in distributed applications to locate each other.

Viewing the JNDI Tree

To view the objects in the WebLogic Server JNDI tree for a specific server, do the following:

1. Right-click the server node in the left pane. This displays a pop-up menu.
2. Select JNDI Tree. The JNDI tree for this server displays in the right pane.

Loading Objects in the JNDI Tree

Using the Administration Console, you load WebLogic Server J2EE services and components, such as RMI, JMS, EJBs, and JDBC Data Sources, in the JNDI tree.

To load an object in the JNDI tree, choose a name under which you want the object to appear in the JNDI tree. Then enter that name in the JNDI Name attribute field when you create the object. When the object is loaded, JNDI provides a path to the object.

To verify if an object has been loaded, see "Viewing the JNDI Tree".

For more information on configuring objects, see Table 19-1 Objects in JNDI Tree.

Table 19-1 Objects In JNDI Tree

Service	Bound Object w/Link to Online Help
EJB	
JDBC DataSource	JDBC Data Source and JDBC Transaction (Tx) Data Source
JMS Connection Factory	JMS Connection Factories
Web Services	Web Application Deployment Descriptor Editor
Mail	MailSession
Deployment Descriptors	BEA WebLogic J2EE Connector Architecture Attribute Descriptions

CHAPTER

20 Managing the WebLogic J2EE Connector Architecture

Based on the Sun Microsystems J2EE Connector Specification, Version 1.0, Proposed Final Draft 2, the WebLogic J2EE Connector Architecture integrates the J2EE Platform with one or more heterogeneous Enterprise Information Systems (EIS). The following sections explain how to manage and administer the WebLogic J2EE Connector Architecture:

- Overview of WebLogic J2EE Connector Architecture
- Installing a New Resource Adapter
- Configuring and Deploying a New Connector
- Monitoring
- Deleting a Connector
- Editing Resource Adapter Deployment Descriptors

For more information on the BEA WebLogic J2EE Connector Architecture, refer to Programming the WebLogic J2EE Connector Architecture.

Overview of WebLogic J2EE Connector Architecture

BEA WebLogic Server continues to build upon the implementation of the Sun Microsystems J2EE Platform Specification, Version 1.3. The J2EE Connector Architecture adds simplified Enterprise Information System (EIS) integration to the J2EE platform. The goal is to leverage the strengths of the J2EE platform—including component models, and transaction and security infrastructures—to address the challenges of EIS integration.

The J2EE Connector Architecture provides a Java solution to the problem of connectivity between the multitude of application servers and EISes. By using the J2EE Connector Architecture, it is no longer necessary for EIS vendors to customize their product for each application server. An application server vendor (such as BEA WebLogic Server) that conforms to the J2EE Connector Architecture also does not need to add custom code whenever it wants to extend its application server to support connectivity to a new EIS.

The J2EE Connector Architecture enables an EIS vendor to provide a standard resource adapter (also referred to as a connector) for its EIS; the resource adapter plugs into an application server such as WebLogic Server and provides the underlying infrastructure for the integration between an EIS and the application server.

An application server vendor (BEA WebLogic Server) extends its system only once to support the J2EE Connector Architecture and is then assured of connectivity to multiple EISes. Likewise, an EIS vendor provides one standard resource adapter and it has the capability to plug in to any application server that supports the J2EE Connector Architecture.

Installing a New Resource Adapter

This section discusses how to connect a new connector (resource adapter) to WebLogic Server by using the Administration Console.

1. Start WebLogic Server.
2. Open the Administration Console.
3. Open the Domain you will be working in.
4. Under Deployments, right-click Connectors in the left panel to display the pop-up menu.
5. Select Install a New Connector Component.
6. Enter the path of the resource adapter `.rar` in the text-entry field, or click the Browse button to browse your file system and choose the resource adapter you want to install.
7. Click the Upload button to install the resource adapter. The new resource adapter is added under the Connectors node in the left-hand pane.

Configuring and Deploying a New Connector

This section discusses how to configure and deploy a new connector using the Administration Console.

For more deployment-related information, see Chapter 3, "Resource Adapters," in Programming the WebLogic J2EE Connector Architecture.

Configuring and Deploying Resource Adapters

To configure and deploy a connector using the WebLogic Server Administration Console:

1. Start WebLogic Server.
2. Open the Administration Console.
3. Open the Domain you will be working in.
4. Under Deployments, select Connectors in the left panel. The Connector Deployments table displays in the right pane showing all the deployed Connectors (Resource Adapters).
5. Select Configure a new Connector Component.
6. Enter the following information:
   - Name—modify the default name of the connector component as needed.
   - URI Path—enter the full path of the resource adapter `.rar` file or a directory containing the resource adapter exploded directory format. For example: `c:\myaps\components\myResourceAdapter.rar`
   - Deployed—indicate whether the Resource Adapter `.rar` file should be deployed when created.
7. Click the Create button.
8. Note that the new resource adapter now appears in the Deployments table in the right pane.

Viewing Deployed Resource Adapters

To view a deployed connector in the Administration Console:

1. In the Administration Console under Deployments, select Connectors in the left panel.
2. View a list of deployed Connectors in the Connector Deployments table in the right pane.

Undeploying Deployed Resource Adapters

To undeploy a deployed connector from the WebLogic Server Administration Console:

1. In the Administration Console under Deployments, select Connectors (Resource Adapters) in the left panel.
2. In the Connector Deployments table, select the connector to undeploy.
3. Under the Configuration tab, deselect the Deployed check box.
4. Click Apply.

Undeploying a resource adapter does not remove the resource adapter name from WebLogic Server. The resource adapter remains undeployed for the duration of the Server session, as long as you do not change it once it has been undeployed. You cannot reuse the deployment name with the deploy argument until you reboot the server. You can re-use the deployment name to update the deployment, as described in "Updating Deployed Resource Adapters."

Updating Deployed Resource Adapters

When you update the contents of the resource adapter `.rar` file or deployment directory that has been deployed to WebLogic Server, those updates are not reflected in WebLogic Server until:

- You reboot the server (if the `.rar` or directory is to be automatically deployed) or
- You update the resource adapter deployment using the WebLogic Server Administration Console

From the WebLogic Server Administration Console:

1. In the Administration Console under Deployments, select Connectors (Resource Adapters) in the left panel.
2. In the Connector Deployments table, select the connector to update.
3. Update the Connector Name and Deployed status as needed.
4. Click Apply.

Monitoring

To monitor all connection pool run times for a connector, proceed as follows:

1. Select a connector to monitor in the left pane of the Console.
2. Right-click with your mouse, and select Monitor all Connector Connection Pool Runtimes from the pop-up menu.

   Connection pool run-time information is provided in the right pane for the selected connector.

Note: You can also select access the this information using the right pane of the Console. Select the specific connector that you want to monitor in the table of connectors located in the right pane. Then, select the Monitoring tab and select Monitor all Connector..."

Deleting a Connector

To delete a connector, proceed as follows:

1. Select a connector to delete in the left pane of the Console under Deployments > Connectors > Connector Name.
2. In the table of connectors located in the right pane, select the Trash Can icon.

   The following message is displayed in the right pane:

   ```
   Are you sure you want to permanently delete <Connector Name>
   from the domain configuration?
   ```

3. Click Yes to delete the connector.

Editing Resource Adapter Deployment Descriptors

This section describes the procedure for editing the following resource adapter (connector) deployment descriptors using the Administration Console Deployment Descriptor Editor:

- `ra.xml`
- `weblogic-ra.xml`

For detailed information about the elements in the resource adapter deployment descriptors, refer to Programming the WebLogic J2EE Connector Architecture.

To edit the resource adapter deployment descriptors, follow these steps:

1. Invoke the Administration Console in your browser using the following URL:

   `http://host:port/console`

   where `host` refers to the name of the computer upon which WebLogic Server is running and `port` refers to the port number to which it is listening.

2. Click to expand the Deployments node in the left pane.

3. Click to expand the Connectors node under the Deployments node.

4. Right-click the name of the resource adapter whose deployment descriptors you want to edit and choose Edit Connector Descriptor from the drop-down menu.

   The Administration Console window appears in a new browser. The left pane contains a tree structure that lists all the elements in the two resource adapter deployment descriptors and the right pane contains a form for the descriptive elements of the `ra.xml` file.

5. To edit, delete, or add elements in the resource adapter deployment descriptors, click to expand the node in the left pane that corresponds to the deployment descriptor file you want to edit, as described in the following list:

   - The RA node contains the elements of the `ra.xml` deployment descriptor.
   - The WebLogic RA node contains the elements of the `weblogic-ra.xml` deployment descriptor.

6. To edit an existing element in one of the resource adapter deployment descriptors, follow these steps:

   a. Navigate the tree in the left pane, clicking on parent elements until you find the element you want to edit.

   b. Click the element. A form appears in the right pane that lists either its attributes or subelements.

   c. Edit the text in the form in the right pane.

   d. Click Apply.

7. To add a new element to one of the resource adapter deployment descriptors, follow these steps:

   a. Navigate the tree in the left pane, clicking on parent elements until you find the name of the element you want to create.

   b. Right-click the element and chose Configure a New Element from the drop-down menu.

   c. Enter the element information in the form that appears in the right pane.

   d. Click Create.

8. To delete an existing element from one of the resource adapter deployment descriptors, follow these steps:

   a. Navigate the tree in the left pane, clicking on parent elements until you find the name of the element you want to delete.

   b. Right-click the element and chose Delete Element from the drop-down menu.

   c. Click Yes to confirm that you want to delete the element.

9. Once you have made all your changes to the resource adapter deployment descriptors, click the root element of the tree in the left pane. The root element is the either the name of the resource adapter `*.rar` archive file or the display name of the resource adapter.

10. Click Validate if you want to ensure that the entries in the resource adapter deployment descriptors are valid.

11. Click Persist to write your edits of the deployment descriptor files to disk in addition to WebLogic Server's memory.

CHAPTER

21 Managing WebLogic Server Licenses

Your WebLogic Server requires a valid license to run. The following sections explain how to install and update WebLogic licenses:

- Installing a WebLogic Server License
- Updating a License

Installing a WebLogic Server License

An evaluation copy of WebLogic Server is enabled for 30 days so you can start using WebLogic Server immediately. To use WebLogic Server beyond the 30-day evaluation period, you will need to contact your salesperson about further evaluation or purchasing a license for each IP address on which you intend to use WebLogic Server. All WebLogic Server evaluation products are licensed for use on a single server with access allowed from up to three unique client IP addresses.

If you downloaded WebLogic Server from the BEA Web site, your evaluation license is included with the distribution. The WebLogic Server installation program allows you to specify the location of the BEA home directory, and installs a BEA license file, `license.bea`, in that directory.

Updating a License

You will need to update the BEA license file if one of the following is true:

- You have purchased additional BEA software.
- You obtain a new distribution that includes new products.
- You have applied for and received an extension of your 30-day evaluation license.

In either of these cases, you will receive a license update file by email as an attachment. To update your BEA license file, do the following:

1. Save the license update file under a name other than `license.bea` in the BEA home directory.
2. Make sure that java (Java 2) is in your path. To add the JDK to your path, enter one of the following commands:
   - `set PATH=.\jdk130\bin;%PATH%` (Windows systems)
   - `set PATH=./jdk130/bin:$PATH` (UNIX systems)
3. In a command shell, `cd` to the BEA home directory and enter the following command:

   `UpdateLicense` *`license_update_file`*

   where *`license_update_file`* is the name under which you saved the license update file that you received via email. Running this command updates the `license.bea` file.
4. Save a copy of your `license.bea` file in a safe place outside the WebLogic distribution. Although no one else can use your license file, you should save this information in a place protected from either malicious or innocent tampering.

A Using the WebLogic Java Utilities

WebLogic provides several Java programs that simplify installation and configuration tasks, provide services, and offer convenient shortcuts. The following sections describe each Java utility provided with WebLogic Server. The command-line syntax is specified for all utilities and, for some, examples are provided.

- `AppletArchiver`
- `Conversion`
- `der2pem`
- `dbping`
- `deploy`
- `getProperty`
- `logToZip`
- `MulticastTest`
- `myip`
- `pem2der`
- `Schema`
- `showLicenses`
- `system`
- `t3dbping`
- `verboseToZip`
- `version`
- `writeLicense`

To use these utilities you must correctly set your `CLASSPATH`. For more information, see "Setting the Classpath Option."

AppletArchiver

The `AppletArchiver` utility runs an applet in a separate frame, keeps a record of all of the downloaded classes and resources used by the applet, and packages these into either a `.jar` file or a `.cab` file. (The `cabarc` utility is available from Microsoft.)

Syntax

```
$ java utils.applet.archiver.AppletArchiver URL filename
```

Argument	Definition
URL	URL for the applet.
filename	Local filename that is the destination for the `.jar/.cab` archive.

Conversion

If you have used an earlier version of WebLogic, you must convert your `weblogic.properties` files. Instructions for converting your files using a conversion script are available in the Administration Console Online Help section called "Conversion."

ClientDeployer

You use weblogic.ClientDeployer to extract the client-side JAR file from a J2EE EAR file, creating a deployable JAR file. The `weblogic.ClientDeployer` class is executed on the Java command line with the following syntax:

```
java weblogic.ClientDeployer ear-file client
```

The ear-file argument is an expanded directory (or Java archive file with a .ear extension) that contains one or more client application JAR files.

For example:

```
java weblogic.ClientDeployer app.ear myclient
```

where `app.ear` is the EAR file that contains a J2EE client packaged in `myclient.jar`.

Once the client-side JAR file is extracted from the EAR file, use the `weblogic.j2eeclient.Main` utility to bootstrap the client-side application and point it to a WebLogic Server instance as follows:

```
java weblogic.j2eeclient.Main clientjar URL [application args]
```

For example

```
java weblogic.j2eeclient.Main helloWorld.jar t3://localhost:7001
Greetings
```

der2pem

The `der2pem` utility converts an X509 certificate from DER format to PEM format. The `.pem` file is written in the same directory as the source `.der` file.

Syntax

```
$ java utils.der2pem derFile [headerFile] [footerFile]
```

Argument	Description
derFile	The name of the file to convert. The filename must end with a `.der` extension, and must contain a valid certificate in `.der` format.
headerFile	The header to place in the PEM file. The default header is "-----BEGIN CERTIFICATE-----". Use a header file if the DER file being converted is a private key file, and create the header file containing one of the following: ■ "-----BEGIN RSA PRIVATE KEY-----" for an unencrypted private key. ■ "-----BEGIN ENCRYPTED PRIVATE KEY-----" for an encrypted private key. **Note:** There must be a new line at the end of the header line in the file.
footerFile	The header to place in the PEM file. The default header is "-----END CERTIFICATE-----". Use a footer file if the DER file being converted is a private key file, and create the footer file containing one of the following in the header: ■ "-----END RSA PRIVATE KEY-----" for an unencrypted private key. ■ "-----END ENCRYPTED PRIVATE KEY-----" for an encrypted private key. **Note:** There must be a new line at the end of the header line in the file.

Example

```
$ java utils.der2pem graceland_org.der
Decoding
.....................................................................
```

dbping

The `dbping` command-line utility tests the connection between a DBMS and your client machine via a JDBC driver. You must complete the installation of the driver before attempting to use this utility.

Syntax

```
$ java -Dbea.home=WebLogicHome utils.dbping DBMS user password DB
```

Argument	Definition
WebLogicHome	The directory containing your WebLogic Server license (`license.bea`). For example, `d:\beaHome\`. Required only if using a BEA-supplied JDBC driver.

DBMS	Choose one of the following for your JDBC driver:
	WebLogic jDriver for Microsoft SQL Server:
	`MSSQLSERVER4`
	WebLogic jDriver for Oracle:
	`ORACLE`
	WebLogic jDriver for Informix:
	`INFORMIX4`
	Oracle Thin Driver:
	`ORACLE_THIN`
	Sybase JConnect driver:
	`JCONNECT`

user	Valid username for login. Use the same values you use with `isql` or `sqlplus`.
password	Valid password for the user. Use the same values you use with `isql` or `sqlplus`.

Argument	Definition
DB	Name of the database. Use the following format, depending on which JDBC driver you use: WebLogic jDriver for Microsoft SQL Server: *DBNAME*@*HOST*:*PORT* WebLogic jDriver for Oracle: *DBNAME* WebLogic jDriver for Informix: *DBNAME*@*HOST*:*PORT* Oracle Thin Driver: *HOST*:*PORT*:*DBNAME* Sybase JConnect driver: JCONNECT: *HOST*:*PORT*:*DBNAME* Where: ■ *HOST* is the name of the machine hosting the DBMS, ■ *PORT* is port on the database host where the DBMS is listening for connections, and ■ *DBNAME* is the name of a database on the DBMS. (For Oracle, this is the name of a DBMS defined in the tnsnames.ora file.)

deploy

The `deploy` utility gets a J2EE application from an archive file (`.jar`, `.war`, or `.ear`) and deploys the J2EE application to a running WebLogic Server. For additional information, see Assembling and Configuring Web Applications and the programming guide Developing WebLogic Server Applications.

Syntax

```
$ java weblogic.deploy [options] [action] password name
{application
  name} {source}
```

Actions (select one of the following)

Action	Description
`delete`	Deletes the application specified by the application name.
`deploy`	Deploys a J2EE application `.jar`, `.war`, `.rar`, or `.ear` file to the specified server.
`list`	Lists all applications in the specified WebLogic Server.
`undeploy`	Removes an existing application from the specified server.
`update`	Redeploys an application. **Note:** Updating an application on any single server instance to which it is targeted causes it to be updated on all servers to which is targeted. For instance, if an application is targeted to a cluster, and you update it on one of the clustered servers instances, the application will be updated on all members of cluster. Similarly, if the application is targeted to a cluster and to a standalone server instance, updating it on the standalone server instance will result in its update on the cluster, and vice versa.

Other Required Arguments

Argument	Description
`password`	Specifies the system password for the WebLogic Server.
`application name`	Identifies the name of the application. The application name can be specified at deployment time, either with the deployment or console utilities.
`source`	Specifies the exact location of the application archive file (`.jar`, `.war`, or `.ear`), or the path to the top level of an application directory.

Options

Option	Definition
`-component` *componentname:target1, target2*	Component to be deployed on various targets, must be specified as: *componentname:target1,target2* where *componentname* is the name of the `.jar`, `.rar` or `.war` file without the extension. This option can be specified multiple times for any number of components (`.jar, .rar` or `.war`). To deploy an `.ear` file, enter each of its components separately using this option and specify the `.ear` using the `-source` argument. For example, to deploy `jubilee.jar` and `wallance.war` in an `.ear` called `myDogApp.ear`, enter: `weblogic.deploy -component jubilie:myserver -component wallance:myserver deploy gumby1234 appname myDogApp.ear` (Enter the above command on a single line.) If the components are in exploded directory format, use their directory name in place of the archive file name.
`-debug`	Prints detailed debugging information to `stdout` during the deployment process.
`-help`	Prints a list of all options available for the deploy utility.
`-host` *host*	Specifies the host name of the WebLogic Server to use for deploying the J2EE application (`.jar`, `.war`, `.ear`). If you do not specify this option, the deploy utility attempts to connect using the host name `localhost`.

Option	Definition
`-jspRefreshComponentName`	Specifies the webapp component to which the refreshed files are being copied. Use this option together with the `-jspRefreshFiles` option to refresh static files. For more information on using this option, see Refreshing Static Components in *Deploying Web Applications*.
`-jspRefreshFiles`	Refreshes static files such as JSPs, HTML files, image files such as `.gif` and `.jpg`, and text files. Class files may not be refreshed. To update class files, use the update flag to redeploy your application. For more information on using this option, see Refreshing Static Components in *Deploying Web Applications.*
`-port` *port*	Specifies the port number of the WebLogic Server to use for deploying the J2EE application `.jar`, `.war`, or `.ear` file. **Note:** If you do not specify the `-port` option, `deploy` connects uses a default of 7001.
`-url` *url*	Specifies the URL of a Weblogic Server. The default is `localhost:7001`.
`-username` *username*	Name of the user with which a connection will be made. The default is `system`.
`-version`	Prints the version of the deploy utility.

Examples

The **`deploy`** utility is useful for various purposes, including the following:

- Viewing a Deployed J2EE Application
- Deploying a New J2EE Application

- Removing a Deployed J2EE Application
- Updating a Deployed J2EE Application

Viewing a Deployed J2EE Application

To view an application that is deployed on a local WebLogic Server, enter the following command:

```
% java weblogic.deploy list password
```

The value of *password* is the password for the WebLogic Server system account.

To list a deployed application on a remote server, specify the `port` and `host` options, as follows:

```
% java weblogic.deploy -port port_number -host host_name list password
```

Deploying a New J2EE Application

To deploy a J2EE application file (`.jar`, `.war`, or `.ear`) or application directory that is not deployed to WebLogic, enter the following command:

```
% java weblogic.deploy -port port_number -host host_name
   deploy password application source
```

The values are as follows:

- *application* is the string you want to assign to this Application.
- *source* is the full pathname of the J2EE application file (`.jar`,`.war`,`.ear`) you want to deploy, or the full pathname of the application directory.

For example:

```
% java weblogic.deploy -port 7001 -host localhost deploy weblogicpwd Basic_example
   c:\mysamples\ejb\basic\BasicStatefulTraderBean.jar
```

Note: The J2EE application file (`.jar`,`.war`,`.ear`) copied to the applications directory of the Administration Server is renamed with the name of the application. Therefore, in the previous example, the name of the application archive `. . ./config/mydomain/applications` directory is changed from `BasicStatefulTraderBean.jar` to `Basic_example.jar`.

Removing a Deployed J2EE Application

To remove a deployed J2EE application, you need only reference the assigned application name, as shown in the following example:

```
% java weblogic.deploy -port 7001 -host localhost undeploy
  weblogicpwd Basic_example
```

Note: Removing a J2EE application does not remove the application from WebLogic Server. You cannot re-use the application name with the `deploy` utility. You can re-use the application name to `update` the deployment, as described in the following section.

Updating a Deployed J2EE Application

To update a J2EE application, use the `update` argument and specify the name of the active J2EE application as follows:

```
% java weblogic.deploy -port 7001 -host localhost update
   weblogicpwd Basic_example
   c:\updatesample\ejb\basic\BasicStatefulTraderBean.jar
```

To update a specific component, enter the following commands:

```
% java weblogic.deploy -port 7001 -host localhost -component
Basic_example:sampleserver,exampleserver update
weblogicpwd Basic_example
c:\updatesample\ejb\basic\BasicStatefulTraderBean.jar
```

`update` will cause the application or component to be updated on *all* server instances to which is targeted. See “update” on page -7.

getProperty

The `getProperty` utility gives you details about your Java setup and your system. It takes no arguments.

Syntax

```
$ java utils.getProperty
```

Example

```
$ java utils.getProperty
-- listing properties --
user.language=en
java.home=c:\java11\bin\..
awt.toolkit=sun.awt.windows.WToolkit
file.encoding.pkg=sun.io
java.version=1.1_Final
file.separator=\
line.separator=
user.region=US
file.encoding=8859_1
java.vendor=Sun Microsystems Inc.
user.timezone=PST
user.name=mary
os.arch=x86
os.name=Windows NT
java.vendor.url=http://www.sun.com/
user.dir=C:\weblogic
java.class.path=c:\weblogic\classes;c:\java\lib\cla...
java.class.version=45.3
os.version=4.0
path.separator=;
user.home=C:\
```

logToZip

The `logToZip` utility searches an HTTP server log file in common log format, finds the Java classes loaded into it by the server, and creates an uncompressed `.zip` file that contains those Java classes. It is executed from the document root directory of your HTTP server.

To use this utility, you must have access to the log files created by the HTTP server.

Syntax

```
$ java utils.logToZip logfile codebase zipfile
```

Argument	Definition
logfile	Required. Fully-qualified pathname of the log file.
codebase	Required. Code base for the applet, or `""` if there is no code base. By concatenating the code base with the full package name of the applet, you get the full pathname of the applet (relative to the HTTP document root).
zipfile	Required. Name of the `.zip` file to create. The resulting `.zip` file is created in the directory in which you run the program. The pathname for the specified file can be relative or absolute. In the examples, a relative pathname is given, so the `.zip` file is created in the current directory.

Examples

The following example shows how a `.zip` file is created for an applet that resides in the document root itself, that is, with no code base:

```
$ cd /HTTP/Serv/docs
$ java utils.logToZip /HTTP/Serv/logs/access "" app2.zip
```

The following example shows how a `.zip` file is created for an applet that resides in a subdirectory of the document root:

```
C:\>cd \HTTP\Serv
C:\HTTP\Serv>java utils.logToZip \logs\applets\classes app3.zip
```

MulticastTest

The `MulticastTest` utility helps you debug multicast problems when configuring a WebLogic Cluster. The utility sends out multicast packets and returns information about how effectively multicast is working on your network. Specifically, `MulticastTest` displays the following types of information via standard output:

1. A confirmation and sequence ID for each message sent out by this server.
2. The sequence and sender ID of each message received from any clustered server, including this server.
3. A missed-sequenced warning when a message is received out of sequence.
4. A missed-message warning when an expected message is not received.

To use `MulticastTest`, start one copy of the utility on each node on which you want to test multicast traffic.

Warning: Do NOT run the `MulticastTest` utility by specifying the same multicast address (the `-a` parameter) as that of a currently running WebLogic Cluster. The utility is intended to verify that multicast is functioning properly before starting your clustered WebLogic Servers.

For information about setting up multicast, see the configuration documentation for the operating system/hardware of the WebLogic Server host. For more information about configuring a cluster, see *Using WebLogic Server Clusters*.

Syntax

```
$ java utils.MulticastTest -n name -a address [-p portnumber]
   [-t timeout] [-s send]
```

Argument	Definition
`-n` *`name`*	Required. A name that identifies the sender of the sequenced messages. Use a different name for each test process you start.
`-a` *`address`*	Required. The multicast address on which: (a) the sequenced messages should be broadcast; and (b) the servers in the clusters are communicating with each other. (The default for any cluster for which a multicast address is not set is 237.0.0.1.)

Argument	Definition
`-p portnumber`	Optional. The multicast port on which all the servers in the cluster are communicating. (The multicast port is the same as the listen port set for WebLogic Server, which defaults to 7001 if unset.)
`-t timeout`	Optional. Idle timeout, in seconds, if no multicast messages are received. If unset, the default is 600 seconds (10 minutes). If a timeout is exceeded, a positive confirmation of the timeout is sent to `stdout`.
`-s send`	Optional. Interval, in seconds, between sends. If unset, the default is 2 seconds. A positive confirmation of each message sent out is sent to `stdout`.

Example

```
$ java utils.MulticastTest -N server100 -A 237.155.155.1
Set up to send and receive on Multicast on Address 237.155.155.1 on
port 7001
Will send a sequenced message under the name server100 every 2
seconds.
Received message 506 from server100
Received message 533 from server200
   I (server100) sent message num 507
Received message 507 from server100
Received message 534 from server200
   I (server100) sent message num 508
Received message 508 from server100
Received message 535 from server200
   I (server100) sent message num 509
Received message 509 from server100
Received message 536 from server200
   I (server100) sent message num 510
Received message 510 from server100
Received message 537 from server200
   I (server100) sent message num 511
Received message 511 from server100
Received message 538 from server200
   I (server100) sent message num 512
Received message 512 from server100
Received message 539 from server200
   I (server100) sent message num 513
Received message 513 from server100
```

myip

The `myip` utility returns the IP address of the host.

Syntax

```
$ java utils.myip
```

Example

```
$ java utils.myip
Host toyboat.toybox.com is assigned IP address: 192.0.0.1
```

pem2der

The `pem2der` utility converts an X509 certificate from PEM format to DER format. The `.der` file is written in the same directory as the source `.pem` file.

Syntax

```
$ java utils.pem2der pemFile
```

Argument	Description
pemFile	The name of the file to be converted. The filename must end with a `.pem` extension, and it must contain a valid certificate in `.pem` format.

Example

```
$ java utils.pem2der graceland_org.pem
Decoding
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
```

Schema

The `Schema` utility lets you upload SQL statements to a database using the WebLogic JDBC drivers. For additional information about database connections, see *Programming WebLogic JDBC*.

Syntax

```
$ java utils.Schema driverURL driverClass [-u username]
   [-p password] [-verbose SQLfile]
```

Argument	Definition
driverURL	Required. URL for the JDBC driver.
driverClass	Required. Pathname of the JDBC driver class.
`-u` *username*	Optional. Valid username.
`-p` *password*	Optional. Valid password for the user.
-verbose	Optional. Prints SQL statements and database messages.
SQLfile	Required when the `-verbose` argument is used. Text file with SQL statements.

Example

The following code shows a sample `Schema` command line:

```
$ java utils.Schema "jdbc:cloudscape:demo;create=true"
   COM.cloudscape.core.JDBCDriver
   -verbose examples/utils/ddl/demo.ddl
```

The following code shows a sample `.ddl` file:

```
    DROP TABLE ejbAccounts;
    CREATE TABLE ejbAccounts
      (id   varchar(15),
       bal  float,
       type varchar(15));
    DROP TABLE idGenerator;
    CREATE TABLE idGenerator
```

```
(tablename varchar(32),
 maxkey    int);
```

showLicenses

The `showLicenses` utility displays license information about BEA products installed in this machine.

Syntax

```
$ java -Dbea.home=license_location utils.showLicenses
```

Argument	Description
license_location	The fully qualified name of the directory where the `license.bea` file exists.

Example

```
$ java -Dbea.home=d:\bea utils.showLicense
```

system

The `system` utility displays basic information about your computer's operating environment, including the manufacturer and version of your JDK, your `CLASSPATH`, and details about your operating system.

Syntax

```
$ java utils.system
```

Example

```
$ java utils.system
* * * * * * * java.version * * * * * * *
1.1.6

* * * * * * * java.vendor * * * * * * *
Sun Microsystems Inc.

* * * * * * * java.class.path * * * * * * *
\java\lib\classes.zip;\weblogic\classes;
\weblogic\lib\weblogicaux.jar;\weblogic\license
...

* * * * * * * os.name * * * * * * *
Windows NT

* * * * * * * os.arch * * * * * * *
x86

* * * * * * * os.version * * * * * * *
4.0
```

t3dbping

The `t3dbping` utility tests a WebLogic JDBC connection to a DBMS via any two-tier JDBC driver. You must have access to a WebLogic Server and a DBMS to use this utility.

Syntax

```
$ java utils.t3dbping WebLogicURL username password DBMS
driverClass driverURL
```

Argument	Definition
WebLogicURL	Required. URL of the WebLogic Server.
username	Required. Valid username of DBMS user.
password	Required. Valid password of DBMS user.
DBMS	Required. Database name.
driverClass	Required. Full package name of the WebLogic two-tier driver.
driverURL	Required. URL of the WebLogic two-tier driver.

verboseToZip

When executed from the document root directory of your HTTP server, `verboseToZip` takes the standard output from a Java application run in verbose mode, finds the Java classes referenced, and creates an uncompressed `.zip` file that contains those Java classes.

Syntax

```
$ java utils.verboseToZip inputFile zipFileToCreate
```

Argument	Definition
inputFile	Required. Temporary file that contains the output of the application running in verbose mode.
zipFileToCreate	Required. Name of the `.zip` file to be created. The resulting `.zip` file is be created in the directory in which you run the program.

UNIX Example

```
$ java -verbose myapplication > & classList.tmp
$ java utils.verboseToZip classList.tmp app2.zip
```

NT Example

```
$ java -verbose myapplication > classList.tmp
$ java utils.verboseToZip classList.tmp app3.zip
```

version

The `version` utility displays version information about your installed WebLogic via `stdout`.

Syntax

```
$ java weblogic.Admin -url host:port -username username -password
password VERSION
```

Example

```
$ java weblogic.Admin
-url localhost:7001 -username system -password foo VERSION
```

writeLicense

The `writeLicense` utility writes information about all your WebLogic licenses in a file called `writeLicense.txt`, located in the current directory. This file can then be emailed, for example, to WebLogic technical support.

Syntax

```
$ java utils.writeLicense -nowrite -Dbea.home=path
```

Argument	Definition
`-nowrite`	Required. Sends the output to `stdout` instead of `writeLicense.txt`.
`-Dbea.home`	Required. Sets WebLogic system home (the root directory of your WebLogic installation). **Note:** This argument is required unless you are running `writeLicense` from your WebLogic system home.

Examples

```
$ java utils.writeLicense -nowrite
```

Example of UNIX Output

```
* * * * * * System properties * * * * * *

* * * * * * * java.version * * * * * * *
1.1.7

* * * * * * * java.vendor * * * * * * *
Sun Microsystems Inc.

* * * * * * * java.class.path * * * * * * *
c:\weblogic\classes;c:\weblogic\lib\weblogicaux.jar;
c:\java117\lib\classes.zip;c:\weblogic\license
...
```

Example of Windows NT Output

```
* * * * * * * os.name * * * * * * *
Windows NT

* * * * * * * os.arch * * * * * * *
x86

* * * * * * * os.version * * * * * * *
4.0

* * * * * * IP * * * * * *
Host myserver is assigned IP address: 192.1.1.0

* * * * * * Location of WebLogic license files * * * * * *
No WebLogicLicense.class found

No license.bea license found in
weblogic.system.home or current directory

Found in the classpath: c:/weblogic/license/license.bea
Last Modified: 06/02/1999 at 12:32:12

* * * * * * Valid license keys * * * * * *
Contents:
Product Name    : WebLogic
 IP Address     : 192.1.1.0-255
 Expiration Date: never
 Units          : unlimited
 key            : b2fcf3a8b8d6839d4a252b1781513b9
...

* * * * * * All license keys * * * * * *
Contents:
Product Name    : WebLogic
 IP Address     : 192.1.1.0-255
 Expiration Date: never
 Units          : unlimited
 key            : b2fcf3a8b8d6839d4a252b1781513b9
...

* * * * * * WebLogic version * * * * * *
WebLogic Build: 4.0.x xx/xx/1999 10:34:35 #xxxxx
```

B WebLogic Server Command-Line Interface Reference

The following sections discuss the WebLogic Server command-line interface syntax, and describe each WebLogic Server administration, connection pool administration, and Mbean management command:

About the Command-Line Interface

As an alternative to the Administration Console, WebLogic Server offers a command-line interface to its administration tools, as well as to many configuration and run-time Mbean properties.

Use the command-line interface if:

- You want to create scripts for administration and management efficiency.
- You cannot access the Administration Console through a browser.
- You prefer using the command-line interface over a graphical user interface.

Before You Begin

The examples in this document are based on the following assumptions:

- WebLogic Server is installed in the `c:/weblogic` directory.
- The JDK is located in the `c:/java` directory.
- You have started WebLogic Server from the directory in which it was installed.

Before you can run WebLogic Server commands, you must do the following:

1. Install and configure the WebLogic Server software, as described in the *WebLogic Server Installation Guide*. See `http://e-docs.bea.com/wls/docs61/install/index.html`.
2. Set `CLASSPATH` correctly. See "Setting the Classpath Option" on page 2-10.
3. Enable the command-line interface by performing one of the following steps:
   - Start the server from the directory in which it was installed.
   - If you are not starting the server from its installation directory, enter the following command, replacing `c:/weblogic` with the name of the directory in which the WebLogic Server software is installed:

     ```
     -Dweblogic.system.home=c:/weblogic
     ```

An administrator must have the appropriate access control permissions to run commands used to manage run-time MBeans.

See the following sections:

- "WebLogic Server Administration Command Reference" on page -6
- "WebLogic Server Connection Pools Administration Command Reference" on page -27

"Mbean Management Command Reference" on page -37

Using WebLogic Server Commands

This section presents the syntax and required arguments for using WebLogic Server commands. WebLogic Server commands are not case-sensitive.

Syntax

```
java weblogic.Admin [–url URL]
   [ { -username username [-password password] } |
     { [-userconfigfile config-file] [-userkeyfile admin-key] }
   ]
   COMMAND arguments
```

Connection and User Credentials Arguments

Note: When you invoke most `weblogic.Admin` commands, you specify the arguments in Table 21-1 to connect to a WebLogic Server instance and to specify the user credentials of a WebLogic Server user who has permission to invoke the command.

Table 21-1 Conection and User Credentials Arguments

Argument	Definition
`-url` *`URL`*	Specifies one of the following: ■ The listen address of the domain's Administration Server. In most cases, we recommend that you use this URL because it runs the command within the security context of the Administration Server. ■ The listen address of the WebLogic Server that is the target of the command. Use this URL if you cannot access the Administration Server and you want to target a Managed Server. The format is *`hostname`*`:`*`port`*. The default is `localhost:7001`.
`-username` *`username`*	Username that has permission to invoke the command you specify. If you do not specify this argument, `weblogic.Admin` uses a user-configuration file and key file.

Table 21-1 Conection and User Credentials Arguments

Argument	Definition
`-password` *`password`*	The password that is associated with the username. If you specify `-username` *`username`* but do not specify the `-password` argument, `weblogic.Admin` prompts you for a password. If *`WL_HOME`*`\server\bin` is specified in the `PATH` environment variable, `weblogic.Admin` uses a set of WebLogic Server libraries that prevent the password from being echoed to standard out. For information on setting environment variables, see "Setting the Classpath Option" on page 2-10.
`-userconfigfile` *`config-file`*	Specifies the name and location of a user-configuration file, which contains an encrypted username and password. The encrypted username must have permission to invoke the command you specify. If you do not specify `-userconfigfile` *`config-file`*, `weblogic.Admin` searches for a user-configuration file at the default path name. (See "STOREUSERCONFIG" on page -18.)
`-userkeyfile` *`admin-key`*	Specifies the name and location of the key file that is associated with the user-configuration file you specify. When you create a user-configuration file, the `STOREUSERCONFIG` command uses a key file to encrypt the username and password. Only the key file that encrypts a user-configuration file can decrypt the username and password. If you do not specify `-userkeyfile` *`admin-key`*, `weblogic.Admin` searches for a key file at the default path name. (See "STOREUSERCONFIG" on page -18.)

Note: The exit code for all commands is 1 if the Administration client cannot connect to the server.

Summary of User Credentials Arguments

Table 21-1 describes the alternatives that the `weblogic.Admin` utility provides for passing usernames and passwords to a server instance.

In a development environment in which security is not a top priority, you can use the `-username` and `-password` arguments when invoking the `weblogic.Admin` utility directly on the command line or in scripts. With these arguments, the username and password are not encrypted. If you store the values in a script, the user credentials can be used by anyone who has read access to the script.

In an environment in which security is a top priority, create user-configuration files and key files. A user-configuration file contains encrypted user credentials that can be decrypted only by a single key file. You can include the `-userconfigfile` *`config-file`* and `-userkeyfile` *`admin-key`* arguments in scripts without exposing the plain text user credentials to those with read privileges for the script. For information about creating a user-configuration and key file, see "STOREUSERCONFIG" on page -18.

The following list summarizes the order of precedence for the `weblogic.Admin` user-credentials arguments:

- If you specify `-username` *`username`* `-password` *`password`*, the utility passes the unencrypted values to the server instance you specify in the `-url` argument.

  These arguments take precedence over the `{ -userconfigfile` *`config-file`* `-userkeyfile` *`admin-key`* `}` arguments. If you specify both `{ -username` *`username`* `-password` *`password`* `}` and `{ -userconfigfile` *`config-file`* `-userkeyfile` *`admin-key`* `}`, the `weblogic.Admin` utility uses the `{ -username` *`username`* `-password` *`password`* `}` arguments and ignores the user-configuration and key file arguments.

- If you specify `-username` *`username`*, the utility prompts for a password. Then it passes the unencrypted values to the server instance you specify in the `-url` argument.

  This argument also takes precedence over the `{ -userconfigfile` *`config-file`* `-userkeyfile` *`admin-key`* `}` arguments.

- If you specify `{ -userconfigfile` *`config-file`* `-userkeyfile` *`admin-key`* `}` and do not specify `{ -username` *`username`* `[-password` *`password`*`]}`, the utility passes the values that are encrypted in *`config-file`* to the server instance you specify in the `-url` argument.

- If you specify neither `{ -username` *`username`* `[-password` *`password`*`] }` nor `{ -userconfigfile` *`config-file`* `-userkeyfile` *`admin-key`* `}`, the utility searches for a user-configuration file and key file at the default path names. The default path names vary depending on the JVM and the operating system. See "Configuring the Default Path Name" on page -20.

Examples of Providing User Credentials

The following command specifies the username **weblogic** and password **weblogic** directly on the command line:

```
java weblogic.Admin -username weblogic -password weblogic COMMAND
```

The following command uses a user-configuration file and key file that are located at the default pathname:

```
java weblogic.Admin COMMAND
```

See "Configuring the Default Path Name" on page -20.

The following command uses a user-configuration file named `c:\wlUser1-WebLogicConfig.properties` and a key file named `e:\secure\myKey`:

```
java -userconfigfile c:\wlUser1-WebLogicConfig.properties
-userkeyfile e:\secure\myKey COMMAND
```

WebLogic Server Administration Command Reference

The following sections provide information about the WebLogic server administration commands.

Table B-1 presents an overview of WebLogic Server administration commands. The following sections describe command syntax and arguments, and provide an example for each command.

See also "WebLogic Server Connection Pools Administration Command Reference" on page -27.

Table B-1 WebLogic Server Administration Commands Overview

Task	Command	Description
Cancel shut down a WebLogic Server	`CANCEL_SHUTDOWN`	Cancels the SHUTDOWN command for the WebLogic Server that is specified in the URL. See "CANCEL_SHUTDOWN" on page -9.

Table B-1 WebLogic Server Administration Commands Overview (Continued)

Task	Command	Description
Connect to WebLogic Server	`CONNECT`	Makes the specified number of connections to the WebLogic Server and returns two numbers representing the total time for each round trip and the average amount of time (in milliseconds) that each connection is maintained. See "CONNECT" on page -10.
Get Help for one or more commands	`HELP`	Provides syntax and usage information for all WebLogic Server commands (by default) or for a single command if a command value is specified on the HELP command line. See "HELP" on page -11.
View WebLogic Server licenses	`LICENSES`	Lists the licenses for all the WebLogic Server instances installed on a specific server. See "LICENSES" on page -12.
List JNDI naming tree node bindings	`LIST`	Lists the bindings of a node in the JNDI naming tree. See "LIST" on page -13.
Lock WebLogic Server	`LOCK`	Locks a WebLogic Server against non-privileged logins. Any subsequent login attempt initiates a security exception which may contain an optional string message. See "LOCK" on page -14.
Verify WebLogic Server listening ports	`PING`	Sends a message to verify that a WebLogic Server is listening on a port, and is ready to accept WebLogic client requests. See "PING" on page -15.
Viewing server log files	`SERVERLOG`	Displays the server log file generated on a specific server. See "SERVERLOG" on page -16.
Shut down a WebLogic Server	`SHUTDOWN`	Shuts down the WebLogic Server that is specified in the URL. See "SHUTDOWN" on page -17.
Encrypt user credentials in a file	`STOREUSERCONFIG`	Creates an encrypted user-configuration file and its associated key file. You can pass the encrypted values to a server instance instead of entering a username and password on the command line. See "STOREUSERCONFIG" on page -18.

Table B-1 WebLogic Server Administration Commands Overview (Continued)

Task	Command	Description
View threads	`THREAD_DUMP`	Provides a real-time snapshot of the WebLogic Server threads that are currently running. See "THREAD_DUMP" on page -24.
Unlock a WebLogic Server	`UNLOCK`	Unlocks the specified WebLogic Server after a `LOCK` operation. See "UNLOCK" on page -25.
View WebLogic Server version	`VERSION`	Displays the version of the WebLogic Server software that is running on the machine specified by the value of *`URL`*. See "VERSION" on page -26.

Note: The exit code for all commands is 1 if the Administration client cannot connect to the server.

CANCEL_SHUTDOWN

The CANCEL_SHUTDOWN command cancels the SHUTDOWN command for a specified WebLogic Server.

When you use the SHUT_DOWN command, you can specify a delay (in seconds). An administrator may cancel the shutdown command during the delay period. Be aware that the SHUTDOWN command disables logins, and they remain disabled even after cancelling the shutdown. Use the UNLOCK command to re-enable logins.

See "SHUTDOWN" on page -17 and "UNLOCK" on page -25.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
CANCEL_SHUTDOWN
```

Example

In the following example, a system user named `system` with a password of `gumby1234` requests to cancel the shutdown of the WebLogic Server listening on port 7001 on machine `localhost`:

```
java weblogic.Admin -url t3://localhost:7001 -username system
   -password gumby1234 CANCEL_SHUTDOWN
```

CONNECT

Makes the specified number of connections to the WebLogic Server and returns two numbers representing the total time for each round trip and the average amount of time (in milliseconds) that each connection is maintained.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
CONNECT count
```

Argument	Definition
count	Number of connections to be made.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `CONNECT` command to establish 25 connections to a server named `localhost` and return information about those connections:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 CONNECT 25
```

HELP

Provides syntax and usage information for all WebLogic Server commands (by default) or for a single command if a command value is specified on the HELP command line.

Syntax

```
java weblogic.Admin HELP [COMMAND]
```

Example

In the following example, information about using the `PING` command is requested:

```
java weblogic.Admin HELP PING
```

The `HELP` command returns the following to `stdout`:

```
Usage: weblogic.Admin [-url url] [-username username]
   [-password password] <COMMAND> <ARGUMENTS>

         PING <count> <bytes>
```

LICENSES

Lists the licenses for all WebLogic Server instances installed on the specified server.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
LICENSES
```

Example

In the following example, an administrator using the default username (`guest`) and default password (`guest`) requests the license information for a WebLogic Server running on port 7001 of machine `localhost`:

```
java weblogic.Admin -url localhost:7001 -username guest
   -password guest LICENSES
```

LIST

Lists the bindings of a node in the JNDI naming tree.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
   LIST context
```

Argument	Definition
context	Required. The JNDI context for lookup, for example, `weblogic`, `weblogic.ejb`, `javax`.

Example

In this example, user `adminuser`, who has a password of `gumby1234`, requests a list of the node bindings in `weblogic.ejb`:

```
java weblogic.Admin -username adminuser -password gumby1234
   LIST weblogic.ejb
```

LOCK

Locks a WebLogic Server against non-privileged logins. Any subsequent login attempt initiates a security exception which may contain an optional string message.

Note: This command is privileged. It requires the password for the WebLogic Server administrative user.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
LOCK "string_message"
```

Argument	Definition
"string_message"	Optional. Message, in double quotes, to be supplied in the security exception that is thrown if a non-privileged user attempts to log in while the WebLogic Server is locked.

Example

In the following example, a WebLogic Server is locked.

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234
   LOCK "Sorry, WebLogic Server is temporarily out of service."
```

Any application that subsequently tries to log into the locked server with a non-privileged username and password receives the specified message: `Sorry, WebLogic Server is temporarily out of service`.

PING

Sends a message to verify that a WebLogic Server is listening on a port, and is ready to accept WebLogic client requests.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
PING [round_trips] [message_length]
```

Argument	Definition
round_trips	Optional. Number of pings.
message_length	Optional. Size of the packet to be sent in each ping. Requests for pings with packets larger than 10 MB throw exceptions.

Example

In the following example, the command checks a WebLogic Server running on port 7001 of machine `localhost` ten (10) times.

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 PING 10
```

SERVERLOG

Displays the log file generated on a specific server.

- If you do not specify a URL, the server log for the Administration Server is displayed by default.
- If you specify a server URL, you can retrieve a log for a non-Administration Server.
- If you omit the `starttime` and `endtime` arguments, a running display of the entire server log is started.

Syntax

```
java.weblogic.Admin [Connection and User Credentials Arguments]
SERVERLOG [[starttime]|[endtime]]
```

Argument	Definition
starttime	Optional. Earliest time at which messages are to be displayed. If not specified, messages display starts, by default, when the SERVERLOG command is executed. The date format is *yyyy/mm/dd*. Time is indicated using a 24-hour clock. The start date and time are entered inside quotation marks, in the following format: "*yyyy/mm/dd hh:mm*"
endtime	Optional. Latest time at which messages are to be displayed. If not specified, the default is the time at which the SERVERLOG command is executed. The date format is *yyyy/mm/dd*. Time is indicated using a 24-hour clock. The end date and time are entered inside quotation marks, in the following format: "*yyyy/mm/dd hh:mm*"

Example

In the following example, a request is made for a running display of the log for the server listening on port 7001 on machine `localhost`.

```
java weblogic.Admin -url localhost:7001
SERVERLOG "2001/12/01 14:00" "2001/12/01 16:00"
```

The request specifies that the running display should begin at 2:00 p.m. on December 1, 2001, and end at 4:00 p.m. on December 1, 2001.

SHUTDOWN

Shuts down the WebLogic Server that is specified in the URL.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
SHUTDOWN [seconds] [“lockMessage”]
```

Argument	Definition
seconds	Optional. Number of seconds allowed to elapse between the invoking of this command and the shutdown of the server.
“lockMessage”	Optional. Message, in double quotes, to be supplied in the message that is sent if a user tries to log in while the WebLogic Server is locked.

Example

In the following example, a user with the `adminuser` username and an administrative password of `gumby1234` shuts down a WebLogic Server that is listening on port 7001 of machine `localhost`:

```
java weblogic.Admin -url localhost:7001 -username adminuser
-password gumby1234 SHUTDOWN 300 “Server localhost is shutting
down.”
```

After the command is issued, an interval of five minutes (300 seconds) elapses. Then the command shuts down the specified server and sends the following message to `stdout`:

```
Server localhost is shutting down.
```

STOREUSERCONFIG

Creates a user-configuration file and an associated key file. The user-configuration file contains an encrypted username and password. The key file contains a secret key that is used to encrypt and decrypt the username and password.

When you use other `weblogic.Admin` or `weblogic.Deployer` commands, you can specify the user-configuration file and key file instead of entering an unencrypted username and password on the command line or including unencrypted credentials in scripts. See "Summary of User Credentials Arguments" on page -4.

Only the key file that originally encrypted the username and password can decyrpt the values. If you lose the key file, you must create a new user-configuration and key file pair.

Caution: You must ensure that only authorized users can access the key file. Any user who accesses a valid user-configuration and key file pair gains the privileges of the encrypted username. To secure access to the key file, you can store the key file in a directory that provides read and write access only to authorized users, such as WebLogic Server administrators. Alternatively, you can write the key file to a removable medium, such as a floppy or CD, and lock the medium in a drawer when it is not being used.

Unlike other `weblogic.Admin` commands, the `STOREUSERCONFIG` command does not connect to a WebLogic Server instance. The data encryption and file creation are accomplished by the JVM in which the `STOREUSERCONFIG` command runs. Because it does not connect to a WebLogic Server instance, the command cannot verify that the username and password are valid WebLogic Server credentials.

Syntax

```
java weblogic.Admin
  -username username [-password password]
  [ -userconfigfile config-file ] [ -userkeyfile keyfile ]
  STOREUSERCONFIG
```

Argument	Definition
`-userconfig-file` *`config-file`*	Specifies a file pathname at which the `STOREUSERCONFIG` command creates a user-configuration file. The pathname can be absolute or relative to the directory from which you enter the command. If a file already exists at the specified pathname, the command overwrites the file with a new file that contains the newly encrypted username and password. If you do not specify this option, `STOREUSERCONFIG` does the following: ■ To determine the directory in which to create the user-configuration file, it uses the JVM user-home directory. The default value varies depending on the SDK and type of operating system. See "Configuring the Default Path Name" on page -20. ■ To determine the file name, it prepends your operating-system username to the string `-WebLogicConfig.properties`. For example, *`username`*`-WebLogicConfig.properties`. You can use Java options to specify a different username. See "Configuring the Default Path Name" on page -20.
`-userkeyfile` *`keyfile`*	Specifies a file pathname at which the `STOREUSERCONFIG` command creates a key file. The pathname can be absolute or relative to the directory from which you enter the command. If a file already exists at the specified pathname, `STOREUSERCONFIG` uses the existing key file to encrypt the new user-configuration file. If you do not specify this option, `STOREUSERCONFIG` does the following: ■ To determine the directory in which to create the key file, it uses the JVM user-home directory. The default value varies depending on the SDK and type of operating system. See "Configuring the Default Path Name" on page -20. ■ To determine the file name, it prepends your operating-system username to the string `-WebLogicKey.properties`. For example, *`username`*`-WebLogicKey.properties`. You can use Java options to specify a different username. See "Configuring the Default Path Name" on page -20.

Argument	Definition (Continued)
`-username` *`username`* `[-password` *`password`* `]`	Specifies the username and password to encrypt. The `STOREUSER-CONFIG` command does **not** verify that the username and password are valid WebLogic Server user credentials. If you omit the `-password` *`password`* argument, `STOREUSERCON-FIG` prompts you to enter a password.

Configuring the Default Path Name

If you do not specify the location in which to create and use a user-configuration file and key file, the `weblogic.Admin` and `weblogic.Deployer` utilities supply the following default values:

- *`user-home-directory`*`\`*`username`*`-WebLogicConfig.properties`
- *`user-home-directory`*`\`*`username`*`-WebLogicKey.properties`

Where *`user-home-directory`* is the home directory of the operating-system user account as determined by the JVM, and *`username`* is your operating-system username.

The value of the home directory varies depending on the SDK and type of operating system. For example, on UNIX, the home directory is usually "`~`*`username`*." On Windows, the home directory is usually "`C:\Documents and Settings\`*`username`*".

You can use the following Java options to specify values for *`user-home-directory`* and *`username`*:

- `-Duser.home=`*`pathname`* specifies the value of *`user-home-directory`*
- `-Duser.name=`*`usernanme`* specifies the value of *`username`*.

For example, the following command configures the user-home directory to be `c:\myHome` and the user name to be `wlAdmin`. The command will search for the following user-configuration file and user key file:
`c:\myHome\wlAdmin-WebLogicConfig.properties`
`c:\myHome\wlAdmin-WebLogicKey.properties`

```
java -Duser.home=c:\myHome -Duser.name=wlAdmin
weblogic.Admin COMMAND
```

Creating User-Configuration and Key Files

To create user-configuration and key files:

1. Use the `-username` *`username`* and `-password` *`password`* arguments to specify the username and password to be encrypted.
2. Specify the name and location of the user-configuration and key files by doing one of the following:
   - Use the `-userconfigfile` *`config-file`* and `-userkeyfile` *`key-file`* arguments:
     ```
     java weblogic.Admin -username username -password password
      -userconfigfile config-file -userkeyfile key-file
     STOREUSERCONFIG
     ```
   - Use the default behavior to create files named *`user-home-directory`*`\`*`username`*`-WebLogicConfig.properties` and *`user-home-directory`*`\`*`username`*`-WebLogicKey.properties`:
     ```
     java weblogic.Admin -username username -password password
     STOREUSERCONFIG
     ```
   - Use the `-Duser.home=`*`directory`* and `-Duser.name=`*`username`* Java options to create files named *`directory`*`\`*`username`*`-WebLogicConfig.properties` and *`directory`*`\`*`username`*`-WebLogicKey.properties`:
     ```
     java -Duser.home=directory -Duser.name=username
     weblogic.Admin -username username -password password
     STOREUSERCONFIG
     ```

You can change the name and location of a user-configuration file or a key file after you create them, as long as you use the two files as a pair.

Using a Single Key File for Multiple User-Configuration Files

To use one key file to encrypt multiple user-configuration files:

1. Create an initial user-configuration file and key file pair.

   For example, enter the following command:

   ```
   java weblogic.Admin -username username -password password
   -userconfigfile c:\AdminConfig -userkeyfile e:\myKeyFile
   STOREUSERCONFIG
   ```

2. When you create an additional user-configuration file, specify the existing key file.

   For example, enter the following command:

   ```
   java weblogic.Admin -username username -password password
   -userconfigfile c:\anotherConfigFile -userkeyfile e:\myKeyFile
   STOREUSERCONFIG
   ```

Examples

In the following example, a user who is logged in to a UNIX operating system as `joe` encrypts the username `wlAdmin` and password `wlPass`:

```
java weblogic.Admin -username wlAdmin -password wlPass
STOREUSERCONFIG
```

The command determines whether a key file named `~joe/joe-WebLogicKey.properties` exists. If such a file does not exist, it prompts the user to select `y` to confirm creating a key file. If the command succeeds, it creates two files:
`~joe\joe-WebLogicConfig.properties`
`~joe\joe-WebLogicKey.properties`

The file `joe-WebLogicConfig.properties` contains an encrypted version of the strings `wlAdmin` and `wlPass`. Any command that uses the `~joe\joe-WebLogicConfig.properties` file must specify the `~joe\joe-WebLogicKey.properties` key file.

In the following example, the user `joe` is a System Administrator who wants to create a user-configuration file for an operating-system account named `pat`. For the sake of convenience, `joe` wants to create the user-configuration file in `pat`'s home directory, which will simplify the syntax of the `weblogic.Admin` commands that `pat` invokes. For added security, only one key file exists at `joe`'s organization, and it is located on a removable hard drive.

To create a user configuration file in `pat`'s home directory that is encrypted and decrypted by a key file name `e:\myKeyFile`:

```
java -Duser.name=pat -Duser.home="C:\Documents and Settings\pat"
weblogic.Admin -username wlOperatorPat -password wlOperator1
-userkeyfile e:\myKeyFile
STOREUSERCONFIG
```

A user who logs in to `pat`'s account can use the following syntax to invoke `weblogic.Admin` commands:

```
java weblogic.Admin -userkeyfile e:\myKeyFile COMMAND
```

For information on using user-configuration and key files, see "Summary of User Credentials Arguments" on page -4.

THREAD_DUMP

Provides a real-time snapshot of the WebLogic Server threads that are currently running.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
THREAD_DUMP
```

UNLOCK

Unlocks the specified WebLogic Server after a LOCK operation.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
UNLOCK
```

Argument	Definition
username	Required. A valid administrative username must be supplied to use this command.
password	Required. A valid administrative password must be supplied to use this command.

Example

In the following example, an administrator named `adminuser` with a password of `gumby1234` requests the unlocking of the WebLogic Server listening on port 7001 on machine `localhost`:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 UNLOCK
```

VERSION

Displays the version of the WebLogic Server software that is running on the machine specified by the value of *URL*.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
VERSION
```

Example

In the following example, a user requests the version of the WebLogic Server running on port 7001 on machine `localhost`:

```
java weblogic.Admin -url localhost:7001 -username guest
   -password guest VERSION
```

Note: In this example, the default value of both the *username* and *password* arguments, `guest`, is used.

WebLogic Server Connection Pools Administration Command Reference

Table B-2 presents an overview of WebLogic Server administration commands for connection pools. The following sections describe command syntax and arguments, and provide an example for each command.

For additional information about connection pools see *Programming WebLogic JDBC* at `http://e-docs.bea.com/wls/docs61/jdbc/index.html` and Managing JDBC Connectivity in the *Administration Guide* at `http://e-docs.bea.com/wls/docs61/adminguide/jdbc.html`.

Table B-2 WebLogic Server Administration Commands Overview—Connection Pools

Task	Command	Description
Create a Dynamic Connection Pool	`CREATE_POOL`	Allows creation of connection pool while WebLogic Server is running. Note that dynamically created connection pools cannot be used with DataSources or TxDataSources. See "CREATE_POOL" on page -29
Destroy a Connection Pool	`DESTROY_POOL`	Connections are closed and removed from the pool and the pool dies when it has no remaining connections. Only the "system" user or users granted "admin" permission by an ACL associated with a connection pool can destroy the pool. See "DESTROY_POOL" on page -32.
Disable a Connection Pool	`DISABLE_POOL`	You can temporarily disable a connection pool, preventing any clients from obtaining a connection from the pool. Only the "system" user or users granted "admin" permission by an ACL associated with a connection pool can disable or enable the pool. See "DISABLE_POOL" on page -33.
Enable a Connection Pool	`ENABLE_POOL`	When a pool is enabled after it has been disabled, the JDBC connection states for each in-use connection are exactly as they were when the connection pool was disabled; clients can continue JDBC operations exactly where they left off. See "ENABLE_POOL" on page -34.

Table B-2 WebLogic Server Administration Commands Overview—Connection Pools

Task	Command	Description
Determine if a Connection Pool Exists	`EXISTS_POOL`	Tests whether a connection pool with a specified name exists in the WebLogic Server. You can use this command to determine whether a dynamic connection pool has already been created or to ensure that you select a unique name for a dynamic connection pool you want to create. See "EXISTS_POOL" on page -35.
Resets a Connection Pool	`RESET_POOL`	Closes and reopens all allocated connections in a connection pool. This may be necessary after the DBMS has been restarted, for example. Often when one connection in a connection pool has failed, all of the connections in the pool are bad. See "RESET_POOL" on page -36.

CREATE_POOL

Allows creation of connection pool while WebLogic Server is running. For more information, see "Creating a Connection Pool Dynamically" in *Programming WebLogic JDBC* at http://e-docs.bea.com/wls/docs61/jdbc/programming.html#programming004.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
CREATE_POOL poolName  aclName=aclX,
   props=myProps,initialCapacity=1,maxCapacity=1,
   capacityIncrement=1,allowShrinking=true,shrinkPeriodMins=15,
   driver=myDriver,url=myURL
```

Argument	Definition
`poolName`	Required. Unique name of pool.
`aclName`	Required. Identifies the different access lists within `fileRealm.properties` in the server config directory. Paired name must be dynaPool.
`props`	Database connection properties; typically in the format "database login name; database password; server network id".
`initialCapacity`	Initial number of connections in a pool. If this property is defined and a positive number > 0, WebLogic Server creates these connections at boot time. Default is 1; cannot exceed maxCapacity.
`maxCapacity`	Maximum number of connections allowed in the pool. Default is 1; if defined, maxCapacity should be =>1.
`capacityIncrement`	Number of connections that can be added at one time. Default = 1.
`allowShrinking`	Indicates whether or not the pool can shrink when connections are detected to not be in use. Default = true.
`shrinkPeriodMins`	Required. Interval between shrinking. Units in minutes. Minimum = 1.If `allowShrinking = True`, then default = 15 minutes.

Argument	Definition
driver	Required. Name of JDBC driver. Only local (non-XA) drivers can participate.
url	Required. URL of the JDBC driver.
testConnsOnReserve	Indicates reserved test connections. Default = False.
testConnsOnRelease	Indicates test connections when they are released. Default = False.
testTableName	Database table used when testing connections; must be present for tests to succeed. Required if either testConnOnReserve or testConOnRelease are defined.
refreshPeriod	Sets the connection refresh interval. Every unused connection will be tested using TestTableName. Connections that do not pass the test will be closed and reopened in an attempt to reestablish a valid physical database connection. If TestTableName is not set then the test will not be performed.
loginDelaySecs	The number of seconds to delay before creating each physical database connection. This delay takes place both during initial pool creation and during the lifetime of the pool whenever a physical database connection is created. Some database servers cannot handle multiple requests for connections in rapid succession. This property allows you to build in a small delay to let the database server catch up. This delay takes place both during initial pool creation and during the lifetime of the pool whenever a physical database connection is created.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `CREATE_POOL` command to create a dynamic connection pool:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 CREATE_POOL myPool

java weblogic.Admin -url t3://forest:7901 -username system
   -password gumby1234 CREATE_POOL dynapool6 "aclName=someAcl,
   allowShrinking=true,shrinkPeriodMins=10,
   url=jdbc:weblogic:oracle,driver=weblogic.jdbc.oci.Driver,
```

```
initialCapacity=2,maxCapacity=8,
props=user=SCOTT;password=tiger;server=bay816"
```

DESTROY_POOL

Connections are closed and removed from the pool and the pool dies when it has no remaining connections. Only the "system" user or users granted "admin" permission by an ACL associated with a connection pool can destroy the pool.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
DESTROY_POOL poolName [true|false]
```

Argument	Definition
poolName	Required. Unique name of pool.
false (soft shutdown)	Soft shutdown waits for connections to be returned to the pool before closing them.
true (default—hard shutdown)	Hard shutdown kills all connections immediately. Clients using connections from the pool get exceptions if they attempt to use a connection after a hard shutdown.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `DESTROY_POOL` command temporarily freeze the active pool connections:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 DESTROY_POOL myPool false
```

DISABLE_POOL

You can temporarily disable a connection pool, preventing any clients from obtaining a connection from the pool. Only the "system" user or users granted "admin" permission by an ACL associated with a connection pool can disable or enable the pool.

You have to options for disabling a pool. 1) Freezing the connections in a pool that you later plan to enable, and 2) destroy the connections.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
DISABLE_POOL poolName [true|false]
```

Argument	Definition
poolName	Name of the connection pool
false (disables and **suspends**)	Disables the connection pool, and suspends clients that currently have a connection. Attempts to communicate with the database server throw an exception. Clients can, however, close their connections while the connection pool is disabled; the connections are then returned to the pool and cannot be reserved by another client until the pool is enabled.
true (default—disables and **destroys**)	Disables the connection pool, and destroys the client's JDBC connection to the pool. Any transaction on the connection is rolled back and the connection is returned to the connection pool.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `DISABLE_POOL` command to freeze a connection that is to be enabled later:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 DISABLE_POOL myPool false
```

ENABLE_POOL

When a pool is enabled, the JDBC connection states for each in-use connection are exactly as they were when the connection pool was disabled; clients can continue JDBC operations exactly where they left off.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
ENABLE_POOL poolName
```

Argument	Definition
poolName	Name of the connection pool.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `ENABLE_POOL` command to reestablish connections that have been disabled (frozen):

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 ENABLE_POOL myPool
```

EXISTS_POOL

Tests whether a connection pool with a specified name exists in the WebLogic Server. You can use this method to determine whether a dynamic connection pool has already been created or to ensure that you select a unique name for a dynamic connection pool you want to create.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
EXISTS_POOL poolName
```

Argument	Definition
poolName	Name of connection pool.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `EXISTS_POOL` command to determine wether or not a pool with a specific name exists:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 EXISTS_POOL myPool
```

RESET_POOL

This command resets the connections in a registered connection pool.

This is a privileged command. You must supply the password for the WebLogic Server administrative user to use this command. You must know the name of the connection pool, which is an entry in the `config.xml` file.

Syntax

```
java weblogic.Admin URL RESET_POOL poolName system password
```

Argument	Definition
URL	The URL of the WebLogic Server host and port number of the TCP port at which WebLogic is listening for client requests; use `"t3://host:port."`
poolName	Name of a connection pool as it is registered in the WebLogic Server's config.xml file.
password	Administrative password for the user "system". You must supply the username "system" and the administrative password to use this Admin command.

Example

This command refreshes the connection pool registered as "eng" for the WebLogic Server listening on port 7001 of the host xyz.com.

```
java weblogic.Admin t3://xyz.com:7001 RESET_POOL eng system gumby
```

Mbean Management Command Reference

Table B-3 presents an overview of the Mbean management commands. The following sections describe command syntax and arguments, and provide an example for each command.

Table B-3 Mbean Management Command Overview

Task	Command(s)	Description
Create configuration Mbeans	`CREATE`	Creates an instance of a configuration Mbean. Returns `OK` to `stdout` when successful. This command cannot be used for run-time Mbeans. See "CREATE" on page -38.
Delete configuration Mbeans	`DELETE`	Deletes a configuration Mbean. Returns `OK` in `stdout` when successful. This command cannot be used for run-time Mbeans. See "DELETE" on page -39.
View run-time Mbean attributes	`GET`	Displays run-time Mbean attributes. See "GET" on page -40.
Invoke run-time Mbeans	`INVOKE`	Invokes methods that are not designed to get or set attributes. This command can call only run-time Mbeans. See "INVOKE" on page -42.
View run-time metrics and statistics	`INVOKE` `GET`	Run the `INVOKE` and `GET` commands to view run-time metrics and statistics. These commands can call only run-time Mbeans. See "INVOKE" on page -42, and "GET" on page -40.
Set configuration Mbean attributes	`SET`	Sets the specified attribute values for the named configuration Mbean. Returns `OK` on `stdout` when successful. This command cannot be used for run-time Mbeans. See "SET" on page -43.

CREATE

Creates an instance of a configuration Mbean. Returns `OK` to `stdout` when successful. This command cannot be used for run-time Mbeans. The Mbean instance is saved in the `config.xml` file or the security realm, depending on where the changes have been made.

Note: When you create Mbeans, configuration objects are also created.

For more information about creating Mbeans, see *Developing WebLogic Server Applications*, at `http://e-docs.bea.com/wls/docs61/programming/index.html`.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
CREATE –name name –type mbean_type
   [–domain domain_name]

java weblogic.Admin [Connection and User Credentials Arguments]
CREATE –mbean mbean_name
```

Argument	Definition
name	Required. The name you choose for the Mbean that you are creating.
mbean_type	Required. When creating attributes for multiple objects of the same type.
mbean_name	Required. Fully qualified name of an Mbean, in the following format: “*domain*:Type=*type*,Name=*name*” `Type` specifies a type of object grouping and `Name` specifies the Mbean name.
domain_name	Optional. Name of the domain; for example, `mydomain`. If *domain_name* is not specified, the default domain name is used.

Example

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 CREATE -mbean
   “mydomain:Type=Server,Name=acctServer”
```

DELETE

Deletes a configuration Mbean. Returns `OK` in `stdout` when successful. This command cannot be used for run-time Mbeans.

Note: When you delete Mbeans, configuration objects are also deleted.

For more information about deleting Mbeans, see *Developing WebLogic Server Applications*, at `http://e-docs.bea.com/wls/docs61/programming/index.html`.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
DELETE {–type mbean_type|–mbean mbean_name}
```

Arguments	Definition
mbean_type	Required. When deleting attributes for multiple objects of the same type.
mbean_name	Required. Fully qualified name of an Mbean, in the following format: “*domain*:Type=*type*,Name=*name*” `Type` specifies a type of object grouping, and `Name` specifies the Mbean name.

Example

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 DELETE -mbean
   “mydomain:Type=Server,Name=AcctServer”
```

GET

Displays run-time Mbean attributes. You can request a list of attributes for multiple objects of the same type by requesting attributes for the following:

- All Mbeans that belong to the same Mbean type:

  ```
  GET {-pretty} -type mbean_type
  ```

- A specific Mbean:

  ```
  GET {-pretty} -mbean mbean_name
  ```

The name of each of the specified Mbeans is included in the output. If `-pretty` is specified, each attribute name-value pair is displayed on a new line.

The `GET` command can only call run-time Mbeans.

The name-value pair for each attribute is specified within curly brackets. This format facilitates scripting by simplifying the parsing of the output.

The name of the Mbean is included in the output as follows:

```
{mbeanname mbean_name {property1 value} {property2 value}. . .}
{mbeanname mbean_name {property1 value} {property2 value} . . .}
. . .
```

If `-pretty` is specified, each attribute name-value pair is displayed on a new line. The name of each of the specified Mbeans is also included in the output, which is displayed as follows:

```
mbeanname: mbean_name
property1: value
property2: value
.
.
.
mbeanname: mbean_name
property1: value
property2: value
```

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments] GET
{-pretty} {–type mbean_type|–mbean mbean_name}
   [–property property1] [–property property2]...
```

Argument	Definition
mbean_type	Required. When getting attributes for multiple objects of the same type, output includes the name of the Mbean.
mbean_name	Fully qualified name of an Mbean, in the following format: *"domain*:Type=*type*,Location=*location*,Name=*name"* Type specifies a type of object grouping, Location specifies the location of the Mbean, and Name supplies the Mbean name.
pretty	Optional. Produces well-formatted output.
property	Optional. The name of the Mbean attribute or attributes to be listed. **Note:** If an attribute is not specified using this argument, all attributes are displayed.

Example

In the following example, a user requests a display of the Mbean attributes for a server named localhost, which is listening on port 7001:

```
java weblogic.Admin -url localhost:7001 GET -pretty -type Server
```

INVOKE

Invokes the specified method (including arguments) on the specified Mbean. This command can call only run-time Mbeans. Use this command to invoke methods that do not get or set Mbean attributes.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments]
INVOKE {-type mbean_type|-mbean mbean_name} -method
   methodname [argument . . .]
```

Arguments	Definition
mbean_type	Required when invoking attributes for multiple objects of the same type, and must include the fully qualified name of the Mbean, as follows: "domain:Name:name,Type=*type*,Application=*application*"
mbean_name	Required. Fully qualified name of an Mbean, as follows: "domain:Type=*type*,Location=*location*,Name=*name*" where: ■ Type specifies the type of object grouping ■ Location specifies the location of the Mbean ■ Name is the Mbean name When the argument is a String array, the arguments must be passed in the following format: "*String1*;*String2*;. . . "
methodname	Required. Name of the method to be invoked. Following the method name, the user can specify arguments to be passed to the method call, as follows: "*domain*:Name=*name*,Type=*type*"

Example

The following example invokes an administration Mbean named admin_one using the method getAttributeStringValue:

```
java weblogic.Admin -username system -password gumby1234 INVOKE
   -mbean mydomain:Name=admin_one,Type=Administrator
   -method getAttributeStringValue PhoneNumber
```

SET

Sets the specified attribute values for the named configuration Mbean. Returns `OK` on `stdout` when successful. This command cannot be used for run-time Mbeans.

New values are saved to the `config.xml` file or the security realm, depending on where the new values have been defined.

Syntax

```
java weblogic.Admin [Connection and User Credentials Arguments] SET
{–type mbean_type|–mbean mbean_name}
   –property property1 property1_value
   [-property property2 property2_value] . . .
```

Argument	Definition
mbean_type	Required when invoking properties for multiple objects of the same type, and must include the fully qualified name of the Mbean, as follows: "domain:Name:name,Type=*type*,Application=*application*"
mbean_name	Required. Must include the fully qualified name of an Mbean, in the following format: "*domain*:Name=*name*,Location:*location*,Type=*type*" where: ■ `Name` is the Mbean name ■ `Location` specifies the location of the Mbean ■ `Type` specifies the type of object grouping
property	Required. The name of the attribute property to be set.
property _value	Required. The value to be set with the attribute property. ■ When the argument is an Mbean array, the arguments must be passed in the following format: "*domain*:Name=*name*,Type=*type*;*domain*:Name=*name*,Type=*type*" ■ When the argument is a String array, the arguments must be passed in the following format: "*String1*;*String2*;. . . " ■ When setting the attribute properties for a JDBC Connection Pool, you must pass the arguments in the following format: "user:*username*;password:*password*;server:*servername*"

C WebLogic SNMP Agent Command-Line Reference

WebLogic Server can use Simple Network Management Protocol (SNMP) to communicate with enterprise-wide management systems. The WebLogic Server subsystem that gathers WebLogic management data, converts it to SNMP communication modules (trap notifications), and forwards the trap notifications to third-party SNMP management systems is called the WebLogic SNMP agent. The WebLogic SNMP agent runs on the Administration Server and collects information from all Managed Servers within a domain.

The WebLogic SNMP agent provides a command-line interface that enables you to:

- Retrieve the value of WebLogic Server attributes that are exposed as managed objects in the WebLogic Server MIB.
- Generate and receive WebLogic Server traps.

The following sections describe working with the WebLogic SNMP agent through its command-line interface:

For more information about using SNMP with WebLogic Server, refer to the WebLogic SNMP Management Guide.

Required Environment and Syntax for the SNMP Command-Line Interface

Before you use the WebLogic SNMP agent's command-line interface, set up your environment and note command syntax information as described in the following sections.

Environment

To set up your environment for the WebLogic SNMP agent's command-line interface:

1. Install and configure the WebLogic Server software, as described in the *WebLogic Server Installation Guide*. See `http://e-docs.bea.com/wls/docs61/install/index.html`.
2. Enable the WebLogic SNMP agent on the Domain → Configuration → SNMP tab of the Administration Console.

   Note: The `snmpv1trap` and `snmptrapd` commands do not require the SNMP agent to be enabled.
3. Open a command prompt (shell) and do the following:
   a. Add a supported SDK to the shell's `PATH` environment variable.
   b. Set the `CLASSPATH` environment variable as described in "Setting the Classpath Option" on page 2-10.

Common Arguments

All WebLogic SNMP agent commands take the following form:

`java` *`command-name arguments`*

Table C-1 describes arguments that are common to most WebLogic SNMP agent commands.

Table C-1 Common Command Line Arguments

Argument	Definition
`-d`	Includes debugging information and packet dumps in the command output.
`-c` *`snmpCommunity`* [*`@server_name`* | *`@domain_name`*]	Specifies the community name that the WebLogic SNMP agent uses to secure SNMP data **and** specifies the server instance that hosts the objects with which you want to interact. To request the value of an object on the Administration Server, specify: *`snmpCommunity`* where *`snmpCommunity`* is the SNMP community name that you set in the Community Prefix field on the Domain → Configuration → SNMP tab of the Administration Console. To request the value of an object on a Managed Server, specify: *`snmpCommunity@server_name`* where *`server_name`* is the name of the Managed Server. To request the value of an object for all server instances in a domain, send a community string with the following form: *`snmpCommunity@domain_name`* If you do not specify a value, the command assumes `-c public`, which attempts to retrieve the value of an object that is on the Administration Server.
`-p` *`snmpPort`*	Specifies the port number on which the WebLogic SNMP agent listens for requests. If you do not specify a value, the command assumes `-p 161`.
`-t` *`timeout`*	Specifies the number of milliseconds the command waits to successfully connect to the SNMP agent. If you do not specify a value, the command assumes `-t 5000`.
`-r` *`retries`*	Specifies the number of times the command retries unsuccessful attempts to connect to the SNMP agent. If you do not specify a value, the command exits on the first unsuccessful attempt.

Table C-1 Common Command Line Arguments

Argument	Definition
host	Specifies the DNS name or IP address of the computer that hosts the WebLogic Server Administration Server, which is where the WebLogic SNMP agent runs.

Commands for Retrieving the Value of WebLogic Server Attributes

Table C-2 is an overview of commands that retrieve the value of WebLogic Server MBean attributes that are exposed in the WebLogic Server MIB.

Table C-2 Overview of Commands for Retrieving the Value of WebLogic Server Attributes

Command	Description
`snmpwalk`	Returns a recursive list of all managed objects that are below a specified node in the MIB tree. See "snmpwalk" on page -49.
`snmpgetnext`	Returns a description of the managed object that immediately follows an OID that you specify. See "snmpgetnext" on page -51.
`snmpget`	Returns a description of managed objects that correspond to one or more object-instance OIDs. See "snmpget" on page -53.

snmpwalk

Returns a recursive list of all managed objects that are below a specified node in the MIB tree.

If you specify the OID for an object type, the command returns a list of all instances of that type along with all instances of any child object types.

For example, if you specify the OID for an object type that corresponds to an MBean, this command returns a description of all instances of the MBean **and** all instances of the attributes within the MBeans.

To see the WebLogic Server MIB tree, refer to the WebLogic Server SNMP MIB Reference. For more information about the structure of the MIB and its object identifiers (OIDs), refer to "SNMP MIB for WebLogic" in *WebLogic SNMP Management Guide*.

Syntax

```
java snmpwalk [-d] [-c snmpCommunity] [-p snmpPort]
              [-t timeout] [-r retries] host OID
```

Argument	Definition
OID	Specifies the object ID of the node from which you want to retrieve a recursive list of object values. Start the value with '.'; otherwise, references are assumed to be relative to the standard MIB (`.1.3.6.1.2.1`), not the WebLogic Server MIB.

For information about the command arguments that are not listed in the above table, refer to Table C-1.

Example

The following example returns all attributes of the `ServerRuntimeMBean` instance that is hosted on the Administration Server. Note that the OID `.1.3.6.1.4.1.140.625.360` refers to the `serverRuntimeTable` object type in the WebLogic MIB.

```
java snmpwalk localhost .1.3.6.1.4.1.140.625.360
```

If you invoke this command from a computer that is running the Examples Server, the command returns output similar to the following truncated output. Note that the output includes the full OID for each attribute instance below the `serverRuntimeTable` object.

```
Object ID:
.1.3.6.1.4.1.140.625.360.1.1.32.101.98.52.50.55.97.53.101.55.101.
56.97.51.98.97.52.99.97.57.53.100.51.51.98.102.51.98.57.48.98.51.
55
STRING: eb427a5e7e8a3ba4ca95d33bf3b90b37

Object ID:
.1.3.6.1.4.1.140.625.360.1.5.32.101.98.52.50.55.97.53.101.55.101.
56.97.51.98.97.52.99.97.57.53.100.51.51.98.102.51.98.57.48.98.51.
55
STRING: ServerRuntime:examplesServer

Object ID:
.1.3.6.1.4.1.140.625.360.1.10.32.101.98.52.50.55.97.53.101.55.101
.56.97.51.98.97.52.99.97.57.53.100.51.51.98.102.51.98.57.48.98.51
.55
STRING: ServerRuntime

Object ID:
.1.3.6.1.4.1.140.625.360.1.15.32.101.98.52.50.55.97.53.101.55.101
.56.97.51.98.97.52.99.97.57.53.100.51.51.98.102.51.98.57.48.98.51
.55
STRING: examplesServer
...
```

The following example retrieves the name of all servers in the `examples` domain. The OID specified in the example command is the numerical value that the WebLogic Server MIB assigns to the `serverRuntimeName` object type.

```
java snmpwalk -c public@examples localhost
.1.3.6.1.4.1.140.625.360.1.15
```

The following example returns all attributes of the `ServerRuntimeMBean` instance that is hosted on a Managed Server named `MS1`. Note that the OID `.1.3.6.1.4.1.140.625.360` refers to the `serverRuntimeTable` object in the WebLogic MIB.

```
java snmpwalk -c public@MS1 localhost .1.3.6.1.4.1.140.625.360
```

snmpgetnext

Returns a description of the managed object that immediately follows one or more OIDs that you specify.

Instead of the recursive listing that the `snmpwalk` command provides, this command returns the description of only the one managed object whose OID is the next in sequence. You could string together a series of `snmpgetnext` commands to achieve the same result as the `snmpwalk` command.

If you specify an object type, this command returns the first instance of the object type, regardless of how many instances of the type exist.

To see the WebLogic Server MIB tree, refer to the WebLogic Server SNMP MIB Reference. For information about the structure of the MIB and its object identifiers (OIDs), refer to "SNMP MIB for WebLogic" in *WebLogic SNMP Management Guide*.

Syntax

```
java snmpgetnext [-d] [-c snmpCommunity] [-p snmpPort]
             [-t timeout] [-r retries] host OID [OID]...
```

Argument	Definition
OID [*OID*]...	Specifies one or more object IDs. You can specify an OID for an object type or an object instance. Start the values with '.'; otherwise, references are assumed to be relative to the standard MIB (`.1.3.6.1.2.1`), not the WebLogic Server MIB.

For information about the command arguments that are not listed in the above table, refer to Table C-1.

Example

The following example retrieves the name of a JDBC connection pool that has been deployed on the Administration Server. The OID in the example command is for the `jdbcConnectionPoolRuntimeName` object type, which represents the `Name` attribute of the `JDBCConnectionPoolRuntime` MBean.

```
java snmpgetnext localhost .1.3.6.1.4.1.140.625.190.1.15
```

The command returns output similar to the following:

```
Response PDU received from 127.0.0.1/127.0.0.1, community: public
Object ID:
.1.3.6.1.4.1.140.625.190.1.15.32.49.51.54.56.100.54.98.102.97.101
.101.52.100.101.49.53.50.99.55.98.57.55.57.56.54.53.98.49.55.102.
100.102
STRING: demoXAPool
```

To determine whether there are additional JDBC connection pools deployed on the Administration Server, you can use the output of the initial `snmpgetnext` command as input for an additional `snmpgetnext` command:

```
java snmpgetnext localhost
.1.3.6.1.4.1.140.625.190.1.15.32.49.51.54.56.100.54.98.102.97.101
.101.52.100.101.49.53.50.99.55.98.57.55.57.56.54.53.98.49.55.102
.100.102
```

The command returns output similar to the following:

```
Response PDU received from 127.0.0.1/127.0.0.1, community: public
Object ID:
.1.3.6.1.4.1.140.625.190.1.15.32.102.50.55.102.57.102.52.50.54.48
.101.98.97.49.50.100.100.57.52.53.54.52.53.54.53.49.52.50.56.51.
56.102
STRING: demoPool
```

snmpget

Retrieves the value of one or more object instances. This command does not accept OIDs for object types.

Syntax

```
java snmpget [-d] [-c snmpCommunity] [-p snmpPort]
             [-t timeout] [-r retries] host object-instance-OID
             [object-instance-OID]...
```

Argument	Definition
object-instance-OID [*object-instance-OID*]...	The object ID of an object instance. This command does not accept OIDs for object types. Start the value with '.'; otherwise, references are assumed to be relative to the standard MIB, not the WebLogic Server MIB.

Example

The following example retrieves the value of the `serverRuntimeState` and `serverRuntimeListenPort` attribute instances for the Administration Server.

```
java snmpget localhost
.1.3.6.1.4.1.140.625.360.1.60.32.102.100.48.98.101.102.100.99.102
.52.98.97.48.48.49.102.57.53.51.50.100.102.53.55.97.101.52.56.99
.99.97.99
.1.3.6.1.4.1.140.625.360.1.35.32.102.100.48.98.101.102.100.99.102
.52.98.97.48.48.49.102.57.53.51.50.100.102.53.55.97.101.52.56.99.
99.97.99
```

The command returns output similar to the following:

```
Response PDU received from /127.0.0.1, community: public
Object ID:
.1.3.6.1.4.1.140.625.360.1.60.32.102.100.48.98.101.102.100.99.102
.52.98.97.48.48.49.102.57.53.51.50.100.102.53.55.97.101.52.56.99.
99.97.99
STRING: RUNNING
Object ID:
.1.3.6.1.4.1.140.625.360.1.35.32.102.100.48.98.101.102.100.99.102
.52.98.97.48.48.49.102.57.53.51.50.100.102.53.55.97.101.52.56.99.
99.97.99
INTEGER: 7001
```

Commands for Testing Traps

Table C-3 is an overview of commands that generate and receive traps for testing purposes.

Table C-3 Overview of Commands for Retrieving Information about WebLogic Server

Command	Description
`snmpv1trap`	Constructs an SNMPv1 trap and distributes it to the SNMP manager or trap daemon that is running on the specified host and listening on the specified port number. See "snmpv1trap" on page -55.
`snmptrapd`	Starts a daemon that receives traps and prints information about the trap. See "snmptrapd" on page -58.

snmpv1trap

Constructs an SNMPv1 trap and distributes it to the SNMP manager or trap daemon that is running on the specified host and listening on the specified port number. For more information about the trap daemon, refer to "snmptrapd" on page -58.

As part of invoking this command, you specify the value for fields within the trap packet that you want to send. The values that you specify must resolve to traps that are defined in the WebLogic Server MIB. For information about WebLogic Server traps and the fields that trap packets require, refer to "Format of WebLogic Trap Notifications" in the *WebLogic SNMP Management Guide*.

Syntax

```
java snmpv1trap [-d] [-c snmpCommunity] [-p TrapDestinationPort]
          TrapDestinationHost .1.3.6.1.4.140.625
          agent-addr generic-trap specific-trap timestamp
          [OID {INTEGER | STRING | GAUGE | TIMETICKS | OPAQUE |
          IPADDRESS | COUNTER} value] ...
```

Argument	Definition
`-c snmpCommunity`	Specifies a password (community name) that secures the data in the trap. If you do not specify a value, the command assumes `-c public`.
`-p TrapDestinationPort`	Specifies the port number on which the SNMP manager or trap daemon is listening. If you do not specify a value, the command assumes `-p 162`.
`TrapDestinationHost`	Specifies the DNS name or IP address of the computer that hosts the SNMP manager or trap daemon.
`.1.3.6.1.4.140.625`	Specifies the value of the trap's `enterprise` field, which contains the beginning portion of the OID for all WebLogic Server traps.
`agent-addr`	Specifies the value of the trap's `agent address` field. This field is intended to indicate the computer on which the trap was generated. When using the `snmpv1trap` command to generate a trap, you can specify any valid DNS name or IP address.

Argument	Definition
generic-trap	Specifies the value of the trap's `generic trap type` field. For a list of valid values, refer to "Format of WebLogic Trap Notifications" in the *WebLogic SNMP Management Guide*.
specific-trap	Specifies the value of the trap's `specific trap type` field. For a list of valid values, refer to "Format of WebLogic Trap Notifications" in the *WebLogic SNMP Management Guide*.
timestamp	Specifies the value of the trap's `timestamp` field. This field is intended to indicate the length of time between the last re-initialization of the SNMP agent and the time at which the trap was issued. When using the `snmpv1trap` command to generate a trap, any number of seconds is sufficient.
OID {INTEGER | STRING | GAUGE | TIMETICKS | OPAQUE | IPADDRESS | COUNTER} *value*	(Optional) Specifies the value of the trap's `variable bindings` field, which consists of name/value pairs that further describe the trap notification. For each name/value pair, specify an OID, a value type, and a value. For example, a log message trap includes a `trapTime` binding to indicate the time at which the trap is generated. To include this variable binding in the test trap that you generate, specify the OID for the `trapTime` variable binding, the `STRING` keyword, and a string that represents the time: `.1.3.6.1.4.1.140.625.100.5 STRING "2:00 pm"`

Example

The following example generates a log message trap that contains the `trapTime` and `trapServerName` variable bindings. It broadcasts the trap through port 165. In the example:

- `6` is the generic trap value that specifies "other WebLogic Server traps."
- `60` is the specific trap value that WebLogic Server uses to identify log message traps.
- `.1.3.6.1.4.1.140.625.100.5` is the OID for the `trapTime` variable binding and `.1.3.6.1.4.1.140.625.100.10` is the OID for the `trapServerName` variable binding.

```
java snmpv1trap -p 165 localhost .1.3.6.1.4.140.625 localhost 6 60
1000 .1.3.6.1.4.1.140.625.100.5 STRING "2:00 pm"
.1.3.6.1.4.1.140.625.100.10
STRING localhost
```

The SNMP manager (or trap daemon) that is listening at port number `165` receives the trap. If the trap daemon is listening on `165`, it returns the following:

```
Trap received from: /127.0.0.1, community: public
Enterprise: .1.3.6.1.4.140.625
Agent: /127.0.0.1
TRAP_TYPE: 6
SPECIFIC NUMBER: 60
Time: 1000
VARBINDS:
Object ID: .1.3.6.1.4.1.140.625.100.5
STRING: 2:00 pm
Object ID: .1.3.6.1.4.1.140.625.100.10
STRING: localhost
```

snmptrapd

Starts a daemon that receives traps and prints information about the trap.

Syntax

```
java snmpv1trap [-d] [-c snmpCommunity] [-p TrapDestinationPort]
```

Argument	Definition
`-c snmpCommunity`	Specifies that community name that the SNMP agent (or `snmpv1trap` command) used to generate the trap. If you do not specify a value, the command assumes `-c public`.
`-p TrapDestinationPort`	Specifies the port number on which the trap daemon receives traps. If you do not specify a value, the command assumes `-p 162`.

Example

The following command starts a trap daemon and instructs it to listen for requests on port 165. The daemon runs in the shell until you kill the process or exit the shell:

```
java snmptrapd -p 165
```

If the command succeeds, the trap daemon returns a blank line with a cursor. The trap daemon waits in this state until it receives a trap, at which point it prints the trap.

Example: Sending Traps to the Trap Daemon

To generate WebLogic Server traps and receive them through the trap daemon:

1. Open a command prompt (shell) and do the following:
   a. Add a supported SDK to the shell's `PATH` environment variable.
   b. Set the `CLASSPATH` environment variable as described in "Setting the Classpath Option" on page 2-10.
2. To start the trap daemon, enter the following command:

```
java snmptrapd
```

3. Open another shell and do the following:

   a. Add a supported SDK to the shell's `PATH` environment variable.

   b. Set the `CLASSPATH` environment variable as described in "Setting the Classpath Option" on page 2-10.

4. To generate a trap, enter the following command:

```
java snmpv1trap localhost .1.3.6.1.4.140.625 localhost 6 65
1000
```

The `snmpv1trap` command generates a `serverStart` Trap and broadcasts it through port 162.

In the shell in which the trap daemon is running, the daemon prints the following:

```
Trap received from: /127.0.0.1, community: public
Enterprise: .1.3.6.1.4.140.625
Agent: /127.0.0.1
TRAP_TYPE: 6
SPECIFIC NUMBER: 65
Time: 1000
VARBINDS:
```

D Parameters for Web Server Plug-ins

The following sections describe the parameters that you use to configure the Apache, Netscape, and Microsoft IIS Web Server plug-ins:

- Overview
- General Parameters for Web Server Plug-Ins
- SSL Parameters for Web Server Plug-Ins

Overview

You enter the parameters for each Web Server Plug-in special configuration files. Each Web Server has a different name for this configuration file and different rules for formatting the file. For details, see the following sections on each plug-in:

- “Installing and Configuring the Apache HTTP Server Plug-In” on page 11-1
- “ Installing and Configuring the Microsoft Internet Information Server (ISAPI) Plug-In” on page 12-1
- “Installing and Configuring the Netscape Enterprise Server Plug-In (NSAPI)” on page 13-1

Enter Web Server plug-ins parameters as described in the following table.

General Parameters for Web Server Plug-Ins

Note: Parameters are case sensitive.

Parameter	Default	Description
`WebLogicHost`	`none`	Identifies a single instance of WebLogic Server to which HTTP requests should be forwarded. **Note:** Use only when proxying to a single server instance. To proxy to a WebLogic Server cluster, use the `WebLogicCluster` instead.
`WebLogicPort`	`none`	Port at which the WebLogic Server host is listening for WebLogic connection requests. If you are using SSL between the plug-in and WebLogic Server, set this parameter to the SSL listen port (see "Configuring the Listen Port" on page 8-5) and set the `SecureProxy` parameter to `ON`). **Note:** Use only when proxying to a single server instance. If you are using a WebLogic Server Cluster, use the `WebLogicCluster` parameter instead of `WebLogicPort`.

Parameter	Default	Description
`WebLogicCluster`	`none`	Identifes the WebLogic Server instances to which HTTP requests should be forwarded. The `WebLogicCluster` parameter specifies the host name and listen port for each server instance specified. The method of specifying the parameter, and the required format vary by plug-in. See the examples in: ■ "Installing and Configuring the Netscape Enterprise Server Plug-In (NSAPI)" ■ "Installing and Configuring the Microsoft Internet Information Server (ISAPI) Plug-In" ■ "Installing and Configuring the Apache HTTP Server Plug-In" If you are using SSL between the plug-in and WebLogic Server, set the port number to the SSL listen port (see "Configuring the Listen Port" on page 8-5) and set the `SecureProxy` parameter to ON. The plug-in does a simple round-robin among all available cluster members. The server list specified in this property is a starting point for the dynamic server list that the server and plug-in maintain. WebLogic Server and the plug-in work together to update the server list automatically with new, failed, and recovered servers. If a mixed set of clusters and single servers is specified, the dynamic list returned for this parameter will return only the clustered servers. You can disable the use of the dynamic cluster list by setting the `DynamicServerList` parameter to `OFF`. The plug-in directs HTTP requests containing a cookie, URL-encoded session, or a session stored in the POST data to the server in the cluster that originally created the cookie.

Parameter	Default	Description

`PathTrim`	`null`	String trimmed by the plug-in from the beginning of the original URL, before the request is forwarded to WebLogic Server. For example, if the URL
		`http://myWeb.server.com/weblogic/foo`
		is passed to the plug-in for parsing and if `PathTrim` has been set to strip off `/weblogic` before handing the URL to WebLogic Server, the URL forwarded to WebLogic Server is:
		`http://myWeb.server.com:7001/foo`
		Note that if you are newly converting an existing third-party server to proxy requests to WebLogic Server using the plug-in, you will need to change application paths to `/foo` to include `weblogic/foo`. You can use `PathTrim` and `PathPrepend` in combination to change this path.

`PathPrepend`	`null`	String that the plug-in prepends to the beginning of the original URL, after `PathTrim` is trimmed and before the request is forwarded to WebLogic Server.
`ConnectTimeoutSecs`	`10`	Maximum time in seconds that the plug-in should attempt to connect to the WebLogic Server host. Make the value greater than `ConnectRetrySecs`. If `ConnectTimeoutSecs` expires without a successful connection, even after the appropriate retries (see `ConnectRetrySecs`), an `HTTP 503/Service Unavailable` response is sent to the client. You can customize the error response by using the `ErrorPage` parameter.
`WLDNSRefreshInterval`	0 (Lookup once, during startup)	Only applies to NSAPI and Apache. If defined in the proxy configuration, specifies number of seconds interval at which WebLogic Server refreshes the server list.

Parameter	Default	Description
`ConnectRetrySecs`	`2`	Interval in seconds that the plug-in should sleep between attempts to connect to the WebLogic Server host (or all of the servers in a cluster). Make this number less than the `ConnectTimeoutSecs`. The number of times the plug-in tries to connect before returning an `HTTP 503/Service Unavailable` response to the client is calculated by dividing `ConnectTimeoutSecs` by `ConnectRetrySecs`. To specify no retries, set `ConnectRetrySecs` equal to `ConnectTimeoutSecs`. However, the plug-in attempts to connect at least twice. You can customize the error response by using the `ErrorPage` parameter.
FilterPriorityLevel (Microsoft Internet Information Server only)	`2`	`The values for this parameter are 0 (low), 1 (medium), and 2 (high). The default value is 2.` This priority should be set in the iisforward.ini file.This property is used to set the priority level for the iisforward.dll filter in IIS. Priority level is used by IIS to decide which filter will be invoked first, in case multiple filters match the incoming request.

Parameter	Default	Description
`Debug`	`OFF`	Sets the type of logging performed for debugging operations. It is not advisable to switch on these debugging options in production systems. The debugging information is written to the `/tmp/wlproxy.log` file on UNIX systems and `c:\TEMP\wlproxy.log` on Windows NT/2000 systems. You can override this location and filename by setting the `WLLogFile` parameter to a different directory and file. For Debug to work correctly, the system administrator must ensure that write permission for the tmp or TEMP directory has been set to the user logged on to the server. You can set any of the following logging options (the `HFC`, `HTW`, `HFW`, and `HTC` options may be set in combination by entering them separated by commas, for example "`HFC,HTW`"): `ON` — Logs only informational and error messages. `OFF` — No debugging information is logged. `HFC` — The plug-in logs headers from the client, informational, and error messages. `HTW` — Logs headers sent to WebLogic Server, informational messages, and error messages. `HFW` — Logs headers sent from WebLogic Server, informational messages, and error messages. `HTC` — The plug-in logs headers sent to the client, informational messages, and error messages. `ALL` — The plug-in logs headers sent to and from the client and to and from WebLogic Server, information messages, and error messages. `ERR` — Prints only the Error messages in the plug-in.

Parameter	Default	Description
WLLogFile	See the Debug parameter	Specifies path and file name for the log file that is generated when the Debug parameter is set to ON. You must create this directory before setting this parameter.
WLTempDir	See the Debug parameter	Specifies the directory where a wlproxy.log will be created. If the location fails, the Plug-In resorts to creating the log file under C:/temp in Windows and /tmp in all Unix platforms. Also specifies the location of the _wl_proxy directory for post data files. When both WLTempDir and WLLogFile are set, WLLogFile will override as to the location of wlproxy.log. WLTempDir will still determine the location of _wl_proxy directory.
DebugConfigInfo	OFF	Enables the special query parameter "__WebLogicBridgeConfig". Use it to get details about configuration parameters from the plug-in. For example, if you enable "__WebLogicBridgeConfig" by setting DebugConfigInfo and then send a request that includes the query string ?__WebLogicBridgeConfig, then the plug-in gathers the configuration information and run-time statistics and returns the information to the browser. The plug-in does not connect to the WebLogic Server in this case. This parameter is strictly for debugging and the format of the output message can change with releases. For security purposes, keep this parameter turned OFF in production systems.
StatPath (Not available for the Microsoft Internet Information Server Plug-In)	false	If set to true, the plug-in checks the existence and permissions of the translated path ("Proxy-Path-Translated") of the request before forwarding the request to WebLogic Server. If the file does not exist, an HTTP 404 File Not Found response is returned to the client. If the file exists but is not world-readable, an HTTP 403/Forbidden response is returned to the client. In either case, the default mechanism for the Web server to handle these responses fulfills the body of the response. This option is useful if both the WebLogic Server Web Application and the Web Server have the same document root. You can customize the error response by using the ErrorPage parameter.
ErrorPage	none	You can create your own error page that is displayed when your Web server is unable to forward requests to WebLogic Server.

Parameter	Default	Description
`WLSocketTimeoutSecs`	`2 (must be greater than 0)`	Set the timeout for the socket while connecting, in seconds.
`HungServerRecoverSecs)`	`300`	Defines the amount of time the plug-in waits for a response to a request from WebLogic Server. The plug-in waits for `HungServerRecoverSecs` for the server to respond and then declares that server dead, and fails over to the next server. The value should be set to a very large value. If the value is less than the time the servlets take to process, then you may see unexpected results. Minimum value: `10` Maximum value: `Unlimited`
`Idempotent`	`ON`	When set to `ON` and if the servers do not respond within `HungServerRecoverSecs)`, the plug-ins fail over. If set to "`OFF`" the plug-ins do not fail over. If you are using the Netscape Enterprise Server Plug-In, or Apache HTTP Server you can set this parameter differently for different URLs or MIME types.
`CookieName`	`JSESSIONID`	If you change the name of the WebLogic Server session cookie in the WebLogic Server Web Application, you need to change the `CookieName` parameter in the plug-in to the same value. The name of the WebLogic session cookie is set in the WebLogic-specific deployment descriptor, in the `<session-descriptor>` (see `http://e-docs.bea.com/wls/docs61/webapp/weblogic_xml.html#session-descriptor)` element.

Parameter	Default	Description
`DefaultFileName`	`none`	If the URI is "`/`" then the plug-in performs the following steps: 1. Trims the path specified with the `PathTrim` parameter. 2. Appends the value of `DefaultFileName`. 3. Prepends the value specified with `PathPrepend`. This procedure prevents redirects from WebLogic Server. Set the `DefaultFileName` to the default welcome page of the Web Application in WebLogic Server to which requests are being proxied. For example, If the `DefaultFileName` is set to `welcome.html`, an HTTP request like "`http://somehost/weblogic`" becomes "`http://somehost/weblogic/welcome.html`". For this parameter to function, the same file must be specified as a welcome file in all the Web Applications to which requests are directed. For more information, see "Configuring Welcome Pages" at `http://e-docs.bea.com/wls/docs61/webapp/components`. Note for Apache users: If you are using Stronghold or Raven versions, define this parameter inside of a `Location` block, and not in an `IfModule` block.
`MaxPostSize`	`-1`	Maximum allowable size of POST data, in bytes. If the content-length exceeds `MaxPostSize`, the plug-in returns an error message. If set to `-1`, the size of POST data is not checked. This is useful for preventing denial-of-service attacks that attempt to overload the server with POST data.

`MatchExpression`
(Apache HTTP Server only)
	`none`	When proxying by MIME type, set the filename pattern inside of an `IfModule` block using the `MatchExpression` parameter.
		Example when proxying by MIME type:
		`<IfModule mod_weblogic.c>`
		`  MatchExpression *.jsp`
		`   WebLogicHost=myHost|paramName=value`
		`</IfModule>`
		Example when proxying by path:
		`<IfModule mod_weblogic.c>`
		`  MatchExpression /weblogic`
		`   WebLogicHost=myHost|paramName=value`
		`</IfModule>`

Parameter	Default	Description
`FileCaching`	`ON`	When set to `ON`, and the size of the POST data in a request is greater than 2048 bytes, the POST data is stored on disk in a temporary file and forwarded to WebLogic Server in chunks of 8192 bytes. Setting `FileCaching` to `ON`, however, can cause a problem with the progress bar displayed by a browser that indicates the progress of a download. The browser shows that the download has completed even though the file is still being transferred. When set to `OFF` and size of the POST data in a request is greater than 2048 bytes, the POST data is stored in memory and sent to WebLogic Server in chunks of 8192 bytes. Setting to `OFF` causes problems if the server goes down while processing the request because the plug-in is not able to fail over.
`WlForwardPath` (Microsoft Internet Information Server only)	`null`	If `WlForwardPath` is set to "/" all requests are proxied. To forward any requests starting with a particular string, set `WlForwardPath` to the string. For example, setting `WlForwardPath` to `/weblogic` forwards all requests starting with `/weblogic` to Weblogic Server. This parameter is required if you are proxying by path. You can set multiple strings by separating the strings with commas. For example: `WlForwardPath=/weblogic,/bea`.
`KeepAliveSecs` (Does not apply to Apache HTTP Server version 1.3.x)	`30`	The length of time after which an inactive connection between the plug-in and WebLogic Server is closed. You must set `KeepAliveEnabled` to `true` (`ON` when using the Apache plug-in) for this parameter to be effective. The value of this parameter must be less than or equal to the value of the Duration field set in the Administration Console on the Server/HTTP tab, or the value set on the `server` Mbean with the `KeepAliveSecs` attribute.
`KeepAliveEnabled` (Does not apply to Apache HTTP Server version 1.3.x)	`true` (Netscape and Microsoft IIS plug-in) `ON` (Apache plug-in)	Enables pooling of connections between the plug-in and WebLogic Server. Valid values for the Netscape and Microsoft IIS plug-ins are `true` and `false`. Valid values for the Apache plugin-in are `ON` and `OFF`.

Parameter	Default	Description
`QueryFromRequest` (Apache HTTP Server only)	`OFF`	When set to `ON`, specifies that the Apache plug-in use `(request_rec *)r->the request` to pass the query string to WebLogic Server. (For more information, see your Apache documentation.) This behavior is desirable in the following situations: ■ When a Netscape version 4.x browser makes requests that contain spaces in the query string ■ If you are using Raven Apache 1.5.2 on HP When set to `OFF`, the Apache plug-in uses `(request_rec *)r->args` to pass the query string to WebLogic Server.
`MaxSkipTime`	`10`	Valid only if `DynamicServerList` is set to `OFF`. If a WebLogic Server listed in either the `WebLogicCluster` parameter or a dynamic cluster list returned from WebLogic Server fails, the failed server is marked as "bad" and the plug-in attempts to connect to the next server in the list. `MaxSkips` sets the amount of time after which the plug-in will retry the server marked as "bad". The plug-in attempts to connect to a new server in the list each time a unique request is received (that is, a request without a cookie).
`DynamicServerList`	`ON`	When set to `OFF`, the plug-in ignores the dynamic cluster list used for load balancing requests proxied from the plug-in and only uses the static list specified with the `WebLogicCluster` parameter. Normally this parameter should remain set to `ON`. There are some implications for setting this parameter to `OFF`: ■ If one or more servers in the static list fails, the plug-in could waste time trying to connect to a dead server, resulting in decreased performance. ■ If you add a new server to the cluster, the plug-in cannot proxy requests to the new server unless you redefine this parameter. WebLogic Server automatically adds new servers to the dynamic server list when they become part of the cluster.

Parameter	Default	Description
`WLProxySSL`	`OFF`	Set this parameter to ON to maintain SSL communication between the plug-in and WebLogic Server when the following conditions exist: ■ An HTTP client request specifies the HTTPS protocol ■ The request is passed through one or more proxy servers (including the WebLogic Server proxy plug-ins) ■ The connection between the plug-in and WebLogic Server uses the HTTP protocol When WLProxySSL is set to ON, the location header returned to the client from WebLogic Server specifies the HTTPS protocol.
WLExcludePathOrMimeType	`None`	Set this parameter to exclude certain requests from proxying. This parameter can be defined locally at the Location tag level as well as globally. When the property is defined locally, it does not override the global property but defines a union of the two parameters.

SSL Parameters for Web Server Plug-Ins

Note: Parameters are case sensitive.

Parameter	Default	Description
SecureProxy	OFF	Set this parameter to ON to enable the use of the SSL protocol for all communication between the WebLogic Server proxy plug-in and WebLogic Server. Remember to configure a port on the corresponding WebLogic Server for the SSL protocol before defining this parameter. This parameter may be set at two levels: in the configuration for the main server and—if you have defined any virtual hosts—in the configuration for the virtual host. The configuration for the virtual host inherits the SSL configuration from the configuration of the main server if the setting is not overridden in the configuration for the virtual host.
TrustedCAFile	none	Name of the file that contains the digital certificates for the trusted certificate authorities for the WebLogic Server proxy plug-in. This parameter is required if the SecureProxy parameter is set to ON. The filename must include the full directory path of the file.
RequireSSLHostMatch	true	Determines whether the host name to which the WebLogic Server proxy plug-in is connecting must match the Subject Distinguished Name field in the digital certificate of the WebLogic Server to which the proxy plug-in is connecting.
SSLHostMatchOID	22	The ASN.1 Object ID (OID) that identifies which field in the Subject Distinguished Name of the peer digital certificate is to be used to perform the host match comparison. The default for this parameter corresponds to the CommonName field of the Subject Distinguished Name. Common OID values are: ■ Sur Name—23 ■ Common Name—22 ■ Email—13 ■ Organizational Unit—30 ■ Organization—29 ■ Locality—26

Configuring Web Applications and Clusters for the Plug-in

Set the following attributes on a cluster or a Web application to configure security for applications accessed via the plug-in.

- `WeblogicPluginEnabled`—If you set this attribute to true for a cluster or a Web application that receives requests from the `HttpClusterServlet`, the servlet will respond to `getRemoteAddr` calls with the address of the browser client from the proprietary `WL-Proxy-Client-IP` header, instead of returning the Web server address.
- `ClientCertProxy` Enabled—If you set this attribute to true for a cluster or a Web application that receives requests from `HttpClusterServlet`, the plug-in sends client certs to the cluster in the special `WL-Proxy-Client-Cert` header, allowing user authentication to be performed on the proxy server.

Index

D

E

L

M

T

U

V

W