BEA WebLogic Server™

Configuring and Managing WebLogic Server

Version 8.1
Revised: June 28, 2006

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

About This Document

1. Overview of WebLogic Server System Administration

2. Overview of WebLogic Server Domains

5. Configuring, Starting, and Stopping Node Manager

6. Setting Up a WebLogic Server Instance as a Windows Service

7. Server Life Cycle

8. Configuring Web Server Functionality for WebLogic Server

9. Monitoring a WebLogic Server Domain

10.Recovering Failed Servers

11.Configuring Network Resources

About This Document

This document describes how to configure and manage a WebLogic Server domain.

The document is organized as follows:

- Chapter 1, "Overview of WebLogic Server System Administration." provides an overview of WebLogic Server system administration tools and capabilities.
- Chapter 2, "Overview of WebLogic Server Domains." provides general information about WebLogic Server domains.
- Chapter 4, "Overview of Node Manager." describes Node Manager, a stand-alone Java application that you use to remotely control and monitor WebLogic Server instances.
- Chapter 5, "Configuring, Starting, and Stopping Node Manager." describes how to use the Node Manager.
- Chapter 6, "Setting Up a WebLogic Server Instance as a Windows Service." describes how to run WebLogic Server automatically on the Windows platform.
- Chapter 7, "Server Life Cycle." describes the operational phases of a WebLogic Server instance, from start up to shut down.
- Chapter 8, "Configuring Web Server Functionality for WebLogic Server." describes how to use WebLogic Server as a Web server.
- Chapter 9, "Monitoring a WebLogic Server Domain." describes how to monitor the runtime state of a WebLogic Server domain.

- Chapter 10, "Recovering Failed Servers." describes failover procedures for WebLogic Server instances.
- Chapter 11, "Configuring Network Resources." describes how to optimize your WebLogic Server domain for your network.
- Appendix A, "Starting and Stopping Servers: Quick Reference." provides quick procedures for starting WebLogic Server instances.

Audience

This document is written for system administrators responsible for implementing a WebLogic Server installation.

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.

Related Information

WebLogic Server Administration Console Help at http://e-docs.bea.com/wls/docs81/ConsoleHelp/index.html

Using WebLogic Server Clusters at http://e-docs.bea.com/wls/docs81/cluster/index.html

WebLogic Server Command Reference at http://e-docs.bea.com/wls/docs81/admin_ref/index.html

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 the user is told to enter from the keyboard.
	Examples:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

`monospace italic text`	Placeholders. *Example*: `String CustomerName;`

`UPPERCASE MONOSPACE TEXT`	Device names, environment variables, and logical operators.
	*Example*s:
	`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}`

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

CHAPTER 1

Overview of WebLogic Server System Administration

The following sections provide an overview of system administration for WebLogic Server:

Introduction to System Administration

You manage a WebLogic Server installation by using any of several system administration tools provided with WebLogic Server. A WebLogic Server installation can consist of a single WebLogic Server instance or multiple instances, each hosted on one or more physical machines. The system administration tools include the Administration Console, command line utilities, and an API, with which you manage security, database connections, messaging, transaction

processing, and the runtime configuration of your applications. The tools also allow you to monitor the health of the WebLogic Server environment to ensure maximum availability and performance for your applications.

WebLogic Server Domains

The basic administrative unit for a WebLogic Server installation is called a *domain*. A domain is a logically related group of WebLogic Server resources that you manage as a unit. A domain always includes only one instance of WebLogic Server called the *Administration Server*. The Administration Server serves as a central point of contact for server instances and system administration tools. A domain may also include additional WebLogic Server instances called *Managed Servers*.

You can configure some or all of these Managed Servers to be part of a WebLogic Server *cluster*. A cluster is a group of WebLogic Server instances that work together to provide scalability and high-availability for applications. A Managed Server in a cluster can act as a backup for services such as JMS and JTA that are hosted on another server instance in the cluster. Your applications are also deployed and managed as part of a domain.

A Managed Server can also function as a virtual host.

You can organize your domains based on criteria such as:

- *Logical divisions of applications*. For example, a domain devoted to end-user functions such as shopping carts and another domain devoted to back-end accounting applications.
- *Physical location*. Domains for different locations or branches of your business.
- *Size*. Domains organized in small units that can be managed more efficiently, perhaps by different personnel.

For more information about WebLogic Server domains, see:

- "Overview of WebLogic Server Domains" on page 2-1
- "Creating and Configuring Domains Using the Configuration Wizard" on page 3-1
- Configuring Domains in the *Administration Console Online Help*

Figure 1-1 WebLogic Server Domain

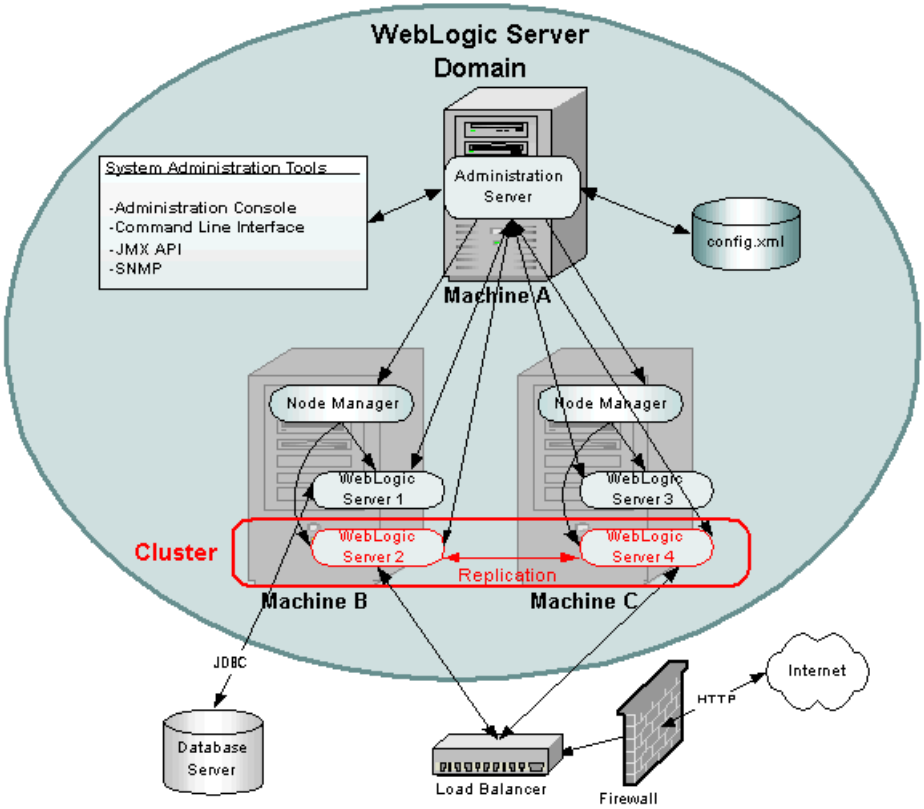


Figure 1-1 depicts a possible configuration of a WebLogic Server Domain—one of many possible configurations.

In the depicted domain, there are three physical machines:

Machine A hosts one instance of WebLogic Server, the Administration Server. The System Administration Tools communicate with the Administration Server to perform configuration and monitoring of the servers and applications in the domain. The Administration Server

communicates with each Managed Server on behalf of the System Administration Tools. The configuration for all the servers in the domain is stored in the configuration repository, the `config.xml` file, which resides on the machine hosting the Administration Server.

Machines B and C each host two instances of WebLogic Server, WebLogic Servers 1 through 4. These instances are called *Managed Servers*. The Administration Server communicates with an instance of Node Manager running on each machine to control startup and shutdown of the Managed Servers.

WebLogic Servers 2 and 4 are part of a *WebLogic Cluster* (outlined in red). This cluster is running an application that responds to HTTP requests routed to the cluster from a hardware load balancer. (An instance of WebLogic Server or a third-party Web server with one of the WebLogic Server plug-ins can also provide load balancing.) The load balancer processes HTTP requests from the Internet after they have passed through a firewall. The load balancer and firewall are not part of the domain. A replicated copy of objects such as HTTP sessions is passed between the two cluster members to provide failover capability.

WebLogic Server 1 runs an application that uses Java Database Connectivity (JDBC) to access a database server running on another physical machine that is not part of the WebLogic Domain.

Note: The pictured domain is only intended to illustrate the concepts of a WebLogic Server domain and how you manage the domain. Many possible configurations of servers, clusters, and applications are possible in a WebLogic Server domain.

System Administration Infrastructure

System administration infrastructure in WebLogic Server is implemented using the Java Management Extension (JMX) specification from Sun Microsystems. The JMX API models system administration functions with Java objects called MBeans. Knowledge of this implementation as described in this discussion of system administration infrastructure is *not* necessary to manage a WebLogic Server domain.

There are three types of MBeans used to manage a WebLogic Server domain: *administration*, *configuration,* and *runtime* Mbeans.

Administration Mbeans contain a set of *attributes* that define configuration parameters for various management functions. All attributes for administration MBeans have pre-set default values. When the Administration Server starts, it reads the domain configuration file (called `config.xml)` and overrides the default attribute values of the administration MBeans with any attribute values found in the `config.xml` file.

The `config.xml` file, located on the machine that hosts the Administration Server, provides persistent storage of Mbean attribute values. Every time you change an attribute using the system administration tools, its value is stored in the appropriate administration MBean and written to the `config.xml` file. Each WebLogic Server domain has its own `config.xml` file.

Each time the Administration Server starts successfully, and each time you modify the configuration, a backup configuration file is created. You can configure the number of back up copies of `config.xml` retained by the Administration Server. For more information, see "Backing up config.xml" on page 10-6.

If you set any configuration attributes on the command line when you start the Administration Server with `-D` arguments, these values override the values set by the defaults or those read from the `config.xml` file. These overridden values are also persisted to `config.xml` file by the Administration Server. For more information about these command-line arguments, see Configuring Servers in the *Administration Console Online Help*.

Configuration Mbeans are copies of Administration Mbeans that each Managed Server uses to initialize its configuration. When you start a Managed Server, the server receives a copy of all the administration MBeans configured on the Administration Server and stores them in memory as configuration MBeans. If you override any configuration attributes when starting a Managed Server, those values override the values received from the Administration Server but are not written to the `config.xml` file. For more information about starting a Managed Server, see Starting Managed Servers in the *Administration Console Online Help*.

Runtime Mbeans contain sets of attributes consisting of runtime information for active WebLogic Servers instances and applications. By retrieving the values of attributes in these runtime MBeans, you can monitor the running status of a WebLogic Server domain.

Mbeans may also contain *operations* used to execute management functions.

Although users with a knowledge of these Mbeans and the JMX API can create their own customized management system, most users prefer to use the system administration tools provided with WebLogic Server to perform these tasks. These tools do not require knowledge of the JMX API. For more information, see "System Administration Tools" on page 1-7.

The Administration Server and Managed Servers

One instance of WebLogic Server in each domain acts as an *Administration Server*. The Administration Server provides a central point for managing a WebLogic Server domain. All other WebLogic Server instances in a domain are called *Managed Servers*. In a domain with only a single WebLogic Server instance, that server functions both as Administration Server and Managed Server.

For a typical production system, BEA recommends that you deploy your applications only on Managed Servers. This practice allows you to dedicate the Administration Server to configuration and monitoring of the domain, while one or more Managed Servers service your applications.

For more information, see Starting and Stopping Servers in the *Administration Console Online Help*.

Recovery of a Failed Administration Server

A running Administration Server is always required to configure and monitor a domain. By maintaining backups of the `config.xml` file and certain other resources for a domain, you can replace a failed Administration Server with a backup WebLogic Server instance that can assume the role of Administration Server. For more information, see Starting Administration Servers and "Recovering Failed Servers" on page 10-1.

Managed Server Independence

To prevent the Administration Server from becoming a single point of failure, Managed Servers can always function without the presence a running Administration Server. When a Managed Server starts, it contacts the Administration Server to retrieve its configuration information. If a Managed Server is unable to connect to the specified Administration Server during startup, it can retrieve its configuration directly by reading a copy of the `config.xml` file and other files located on the Managed Server's own file system.

A Managed Server that starts in this way is running in *Managed Server Independence* mode. In this mode, a server uses cached application files to deploy the applications that are targeted to the server. You cannot change a Managed Server's configuration until it is able to restore communication with the Administration Server. For more information, see "Recovering Failed Servers" on page 10-1. Failover for Managed Servers is applicable to both clustered and non-clustered servers.

Domain-Wide Administration Port

You can enable an administration port for use with servers in a domain. The administration port is optional, but it provides two important capabilities:

- You can start a server in standby state. While in the standby state, the administration port remains available to activate or administer the server. However, the server's other network connections are unavailable to accept client connections. See Starting and Stopping

WebLogic Server in the *Administration Console Online Help* for more information on the standby state.

- You can separate administration traffic from application traffic in your domain. In production environments, separating the two forms of traffic ensures that critical administration operations (starting and stopping servers, changing a server's configuration, and deploying applications) do not compete with high-volume application traffic on the same network connection.

  For more information, see "Enabling the Domain-Wide Administration Port" in *Administration Console Online Help*

Service Packs and WebLogic Server Instances

All WebLogic Server instances in a domain must run the same version of the WebLogic Server software. 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 8.1, Service Pack 1 while the Managed Servers are running version 8.1 without Service Pack 1.

System Administration Tools

Using JMS as the underlying architecture, System administration tools are provided for a variety of management functions. An Administration Server must be running when you use system administration tools to manage a domain. The following tools are discussed in the next sections:

- "Logs" on page 1-12
- "Editing config.xml" on page 1-13

Security Protections for System Administration Tools

All system administration operations are protected based on the user name employed to access a system administration tool. A user (or the group a user belongs to) must be a member of one of four security roles. These roles grant or deny a user access to various sets of system administration operations. The roles are Admin, Operator, Deployer, and Monitor. You can also set a security policy on WebLogic Server instances in a domain. For more information, see "Protecting User Accounts" in *Managing WebLogic Security*.

System Administration Console

The Administration Console is a Web Application hosted by the Administration Server. You access the Administration Console from any machine on the local network that can communicate with the Administration Server through a Web browser (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, clusters, and applications. The management capabilities include:

- Configuration
- Stopping and starting servers
- Monitoring server health and performance
- Monitoring application performance
- Viewing server logs
- Assistants, which step you through the following tasks:
  - Creating JDBC connection pools and DataSources
  - Deploying your applications
  - Configuring SSL

Through 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.

See:

- "Starting the Administration Console" on page 1-20
- Administration Console Online Help. (The online help is also available from the Administration Console by clicking on the "?" icon located in the upper right portion of the console.)

Command-Line Interface

You use the command-line interface to manage a WebLogic Server domain when using the Administration Console is not practical or desired—such as when you want to manage your domain with scripts, when you cannot use a Web browser to access the Administration Console, or if you prefer using the command-line interface over a graphical user interface. You can use only the command-line interface to manage your domain, or use the command-line interface in combination with other system administration tools such as the Administration Console.

The command line interface invokes a Java class called `weblogic.Admin`. Arguments for this class enable you to perform many common management functions without knowing the JMX API. See:

- weblogic.Admin Command-Line Reference in the *WebLogic Server Command Reference*.
- WebLogic Server API Reference (in Javadocs, the `weblogic.management` packages)

If you require more fine-grained control than the `weblogic.Admin` management functions provide you can also use the command line interface to perform *`set`* or *`get`* operations directly on Mbean attributes. This feature requires knowledge of the WebLogic Server Mbean architecture. For more information, see the following resources:

- Commands for Managing WebLogic Server MBeans
- Javadocs for WebLogic Server classes.
  - Select the `weblogic.management.configuration` package for configuration MBeans (to configure a WebLogic Server Domain)
  - Select the `weblogic.management.runtime` package for runtime MBeans (for monitoring).
- A reference of Mbeans and attributes is provided in the BEA WebLogic Server Configuration Reference.

JMX

Advanced Java programmers with knowledge of the JMX API from Sun Microsystems Inc. and WebLogic Server Mbeans can write their own management components as a Java class.

See:

- Programming WebLogic JMX Services
- WebLogic Server API Reference (in Javadocs, the `weblogic.management` packages.)

Configuration Wizard

The Configuration Wizard is a tool for creating a new WebLogic Server domain or modifying an existing, inactive domain. Use the Configuration Wizard to:

- Create a new WebLogic Server domain—for stand-alone servers, Administration Servers with Managed Servers, and clustered servers.
- Create a new WebLogic Server domain based on a template. Templates allow you to re-create an existing domain for use in another context, such as migrating a domain containing an application under development to a production environment. See "Configuration Template Builder" on page 1-11.
- Add JMS Configurations off-line to an existing domain.
- Add JDBC Connection pools, MultiPools, and DataSources, off-line, to an existing domain.

The Configuration Wizard creates the appropriate directory structure for your domain, a basic `config.xml` file, and scripts you can use to start the servers in your domain. Depending on the options you select in the wizard, other files may also be created.

You can run the Configuration Wizard either using a graphical user interface (GUI) or in a text-based command line environment (this is called *console mode*—do not confuse this mode with the Administration *Console*). You can also create user-defined domain configuration templates for use by the Configuration Wizard.

See:

- "Creating and Configuring Domains Using the Configuration Wizard" on page 3-1
- Configuring WebLogic Platform

Configuration Template Builder

A configuration template defines the full set of resources within a domain, including infrastructure components, applications, services, security options, and general environment and operating system options. BEA provides a number of templates and template extensions as part of the WebLogic Platform product. The templates delivered with your installation are described in the Template Reference.

The Configuration Template Builder provides the capability to easily create your own templates, to enable, for example, the definition and propagation of a standard domain across a development project, or to enable the distribution of a domain along with an application that has been developed to run on that domain. The templates you create with the Configuration Template Builder are used as input to the Configuration Wizard as the basis for creating a domain that is customized for your target environment. See Overview of the WebLogic Configuration Wizard and Configuration Template Builder in *Configuring WebLogic Platform*.

Java Utilities

Utility programs are provided for common tasks such as deploying applications and testing DBMS configurations. For more information, see Using the WebLogic Java Utilities in the *WebLogic Server Command Reference*.

Ant Tasks

You can use two Ant tasks provided with WebLogic Server to help you perform common configuration tasks in a development environment. Ant is a Java-based build tool similar to Make.The configuration tasks enable you to start and stop WebLogic Server instances as well as create and configure WebLogic Server domains. When combined with other WebLogic Ant tasks, you can create powerful build scripts for demonstrating or testing your application with custom domains.

See:

Using Ant Tasks to Configure a WebLogic Server Domain in the *WebLogic Server Command Reference*.

Node Manager

Node Manager is a Java program provided with WebLogic Server that enables you to start, shut down, restart, and monitor remote WebLogic Server instances. To enable these capabilities, you run an instance of Node Manager on each physical machine in your domain.

See:

- "Overview of Node Manager" on page 4-1
- "Configuring, Starting, and Stopping Node Manager" on page 5-1
- "Creating and Configuring Domains Using the Configuration Wizard" on page 3-1

SNMP

WebLogic Server includes the ability to communicate with enterprise-wide management systems using Simple Network Management Protocol (SNMP). The WebLogic Server SNMP capability enables you to integrate management of WebLogic Servers into an SNMP-compliant management system that gives you a single view of the various software and hardware resources of a complex, distributed system.

See:

- WebLogic SNMP Management Guide
- WebLogic SNMP MIB Reference

Logs

Many WebLogic Server operations generate logs of their activity. Each server has its own log as well as a standard HTTP access log. These log files can be configured and used in a variety of ways to monitor the health and activity of your servers and applications.

See:

- Server Log in the *Administration Console Online Help*
- "Setting Up HTTP Access Logs" on page 8-15
- Using WebLogic Logging Services

You can also configure a special *domain log* that contains a definable subset of log messages from all WebLogic Server instances in a domain. You can modify which logged messages from a local server appear in the domain log using the system administration tools. You can view this domain log using the Administration Console or a text editor/viewer.

For more information, see Domain Log Filters in the *Administration Console Online Help*.

Editing config.xml

You can manage a WebLogic Server domain by manually editing the persistent store for configuration, the `config.xml` file. (Other system administration tools automatically save the configuration to the `config.xml` file.) Because of the difficulty of correctly editing the XML syntax required in this file, this method of configuration is not recommended but may provide advantages in limited situations.

Note: Do not edit the `config.xml` file while the Administration Server is running.

For more information, see BEA WebLogic Server Configuration Reference.

Resources You Can Manage in a WebLogic Server Domain

This section discusses the domain resources you manage with the system administration tools.

Servers

The administrative concept of a *server* represents an instance of WebLogic Server in your domain. Using the system administration tools you can:

- Start and stop servers. (To start and stop servers on a remote machine, you must have Node Manager installed on the remote machine.) For more information see "Node Manager" on page 1-11.
- Configure a server's Listen address, listen port, HTTP settings, and time outs.
- Configure HTTP server functionality and Virtual Hosts
- Configure logging and view logs
- Configure Secure Sockets Layer (SSL)
- Deploy applications to specific servers
- Configure WebLogic Server resources active on the server, such as JDBC Connection Pools and startup classes.
- Tune the server for best performance

For more information, see:

- Creating, Configuring, and Monitoring Servers, in the *Administration Console Online Help*
- Starting and Stopping Servers, in the *Administration Console Online Help*

- weblogic.Server Command-Line Reference in the *WebLogic Server Command Reference*

Clusters

WebLogic Server clusters allow you to distribute the work load of your application across multiple WebLogic Server instances. Clusters can improve performance and provide fail-over should a server instance become unavailable. For example, clusters provide several ways to replicate objects used in your applications so that data is not lost in the event of hardware failure.

You can architect combinations of clusters to distribute the work load in a way that provides the best performance for your applications.

Some services that are hosted on a single instance of WebLogic Server can be migrated from one server to another in the event of server failure. The system administration tools allow you to control these migrations.

See:

- Using WebLogic Server Clusters
- Clusters, in the *Administration Console Online Help*

Machines

The administrative concept of a *machine* represents the computer that hosts an instance of WebLogic Server. WebLogic Server uses machine names to determine the optimum server in a cluster to which certain tasks, such as HTTP session replication, are delegated.

Using the system administration tools you can define one or more machines, configure Node Manger for those machines, and assign servers to the machines. For UNIX machines, you can configure UID and GID information.

See:

- Using WebLogic Server Clusters
- Machines, in the *Administration Console Online Help*

Network Channels

A network channel is a configurable resource that defines the attributes of a network connection to WebLogic Server. You can use network channels to manage quality of service, meet varying connection requirements, and improve utilization of your systems and network resources. For more information, see "Configuring Network Resources" on page 11-1.

JDBC

Java Database Connectivity (JDBC) allows Java programs to interact with common DBMSs such as Oracle, Microsoft SQL Server, Sybase, DB2, MySQL, and others.

Using the System Administration tools you can manage and monitor connectivity between WebLogic Server and your database management system. Connectivity is usually established through connection pools or DataSources.

See:

- JDBC, in the *Administration Console Online Help*
- Configuring JDBC Connection Pools, in the *Administration Console Online Help*
- Configuring JDBC DataSources, in the *Administration Console Online Help*
- Configuring JDBC MultiPools, in the *Administration Console Online Help*

JMS

The Java Message Service (JMS) is a standard API for accessing enterprise messaging systems that allow communication between applications.

Using the system administration tools, you can define configuration attributes to:

- Enable JMS
- Create JMS servers
- Create and/or customize values for JMS servers, connection factories, destinations (physical queues and topics), distributed destinations (sets of physical queue and topic members within a cluster), destination templates, destination sort order (using destination keys), persistent stores, paging 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, message paging, flow control, and load balancing for distributed destinations.

For more information, see:

- Configuring JMS, in the *Administration Console Online Help*

- Monitoring JMS, in the *Administration Console Online Help*
- Tuning JMS, in the Administration *Console Online Help*

WebLogic Messaging Bridge

A *Messaging Bridge* transfers messages between two messaging providers. The providers may be another implementation of WebLogic JMS or a third-party JMS provider.

For more information, see Messaging Bridge, in the *Administration Console Online Help*.

Web Servers and Web Components

WebLogic Server can perform as a fully functional Web server. WebLogic Server can serve both static files such as HTML files and dynamic files such as Java servlets or Java ServerPages (JSP). Virtual hosting is also supported.

For more information on managing Web server functionality in WebLogic Server, see:

- "Configuring Web Server Functionality for WebLogic Server" on page 8-1.
- Virtual Hosts, in the *Administration Console Online Help*

You can also use Web Servers such as Apache, Microsoft IIS, and Netscape with WebLogic Server. For more information, see "Using WebLogic Server with Web Servers" on page 1-21.

Applications

You deploy, configure, and manage the applications in your domain using the system administration tools to:

- Deploy applications to one or more WebLogic Servers or clusters in a domain. WebLogic Server uses a two-phase deployment model that gives you fault-tolerant control over the deployment process.
- Configure runtime parameters for the applications.
- Monitor application performance.
- Configure security parameters.
- View an application's deployment descriptor.
- Edit selected runtime elements of deployment descriptors. (Administration Console only). You can also use a text editor, an XML editor, or the WebLogic Builder tool to edit

deployment descriptors. For more information, see "Editing and Creating Deployment Descriptors with WebLogic Builder" on page 1-18.

- Protect access to an application based on security roles or a security policy. For more information see Securing WebLogic Resources.

See:

- WebLogic Server Deployment
- Deploying Applications and Modules, in the *Administration Console Online Help*

Application Formats

You deploy applications in one or more of the following J2EE application formats:

- Web applications
- Enterprise JavaBeans (EJB)
- Enterprise applications
- J2EE connectors
- Web services. Web services are deployed as a Web Application or Enterprise Application that includes a special deployment descriptor that configures the Web Service.

For more information, see:

- Developing Applications
- Assembling and Configuring Web Applications
- Deploying Applications and Modules
- Developing WebLogic Server Applications
- Programming WebLogic Enterprise Java Beans
- Programming WebLogic J2EE Connectors
- Programming WebLogic Web Services
- Defining a Security Policy
- Setting Protections for WebLogic Resources

Editing and Creating Deployment Descriptors with WebLogic Builder

In addition to using the Administration Console to edit selected deployment descriptor elements you can also use the more robust WebLogic Builder tool that is included with your WebLogic Server distribution. WebLogic Builder is a stand-alone graphical tool for assembling a J2EE application, creating and editing deployment descriptors, and deploying an application on WebLogic Server. For more information, see WebLogic Builder Online Help.

Startup and Shutdown Classes

A startup class is a Java program that is automatically loaded and executed when a WebLogic Server instance is started or restarted and after other server initialization tasks have completed. A shutdown class is automatically loaded and executed when a WebLogic Server instance is shut down either from the Administration Console or using the `weblogic.Admin` shutdown command.

You use the system administration tools to register and manage startup and shutdown classes.

For more information, see Startup and Shutdown Classes, in the *Administration Console Online Help*.

JNDI

The Java Naming and Directory Interface (JNDI) API enables applications to look up objects—such as DataSources, EJBs, JMS, and MailSessions—by name. You can view the JNDI tree through the Administration Console.

See:

- JNDI, in the *Administration Console Online Help*
- Programming WebLogic JNDI

Transactions

You use the system administration tools to configure and enable the WebLogic Server Java Transaction API (JTA). The transaction configuration process involves configuring:

- Transaction time outs and limits
- Transaction Manager behavior

See:

- JTA, in the *Administration Console Online Help*
- Programming WebLogic JTA

XML

The XML Registry is a facility for configuring the XML resources of a WebLogic Server domain. XML resources in WebLogic Server include the parser used by an application to parse XML data, the transformer used by an application to transform XML data, external entity resolution, and caching of external entities.

See:

- Administering WebLogic Server XML
- XML, in the *Administration Console Online Help*

Security

The WebLogic Server security subsystem allows you to plug in third-party security solutions and also provides out-of-the box implementations for many common security systems. You can also create your own security solution and implement it in WebLogic Server.

For backwards compatibility, the security functionality available in version 6.0 and 6.1 of WebLogic Server is also supported when the domain is running in Compatibility Security Mode.

Using the administration tools, you can define realms, users, groups, passwords, and other security features.

See:

- Managing WebLogic Security
- Using Compatibility Security in *Managing WebLogic Security*
- "Security" in the *Administration Console Online Help*
- Security Index Page

WebLogic Tuxedo Connector

WebLogic Tuxedo Connector provides interoperability between WebLogic Server applications and Tuxedo services. The connector allows WebLogic Server clients to invoke Tuxedo services

and Tuxedo clients to invoke WebLogic Server Enterprise Java Beans (EJBs) in response to a service request.

See:

- WebLogic Tuxedo Connector
- WebLogic Tuxedo Connector (WTC), in the *Administration Console Online Help*

Jolt

Jolt is a Java-based client API that manages requests to BEA Tuxedo services via a Jolt Service Listener (JSL) running on a Tuxedo server.

See:

- BEA Jolt
- Jolt, in the *Administration Console Online Help*

Mail

WebLogic Server includes the JavaMail API version 1.1.3 reference implementation from Sun Microsystems. Using the JavaMail API, you can add email capabilities to your WebLogic Server applications. JavaMail provides access from Java applications to IMAP- and SMTP-capable mail servers on your network or the Internet. It does not provide mail server functionality; so you must have access to a mail server to use JavaMail.

See:

- "Using JavaMail with WebLogic Server Applications" under Programming Topics.
- Mail, in the *Administration Console Online Help*

Starting the Administration Console

This section contains instructions for starting the Administration Console.

To use the Administration Console, use one of the supported Web Browsers for your environment. See Browser Support for the WebLogic Server Console. If your Web browser is not on this list of supported browsers, you may experience functional or formatting problems when using the Administration Console.

To start the Administration Console:

1. Start an Administration Server. See Starting Administration Servers in the *Administration Console Online Help*.
2. Open one of the supported Web browsers and open the following URL:

   `http://hostname:port/console`

   Where *hostname* is the DNS name or IP address of the Administration Server and *port* is the listen port on which the Administration Server is listening for requests (port 7001 by default). If you have configured a domain-wide Administration port, use that port number. If you configured the Administration Server to use Secure Socket Layer (SSL) you must add `s` after `http` as follows:

   `https://hostname:port/console`

   Note: A domain-wide administration port always uses SSL. See Server --> Configuration --> Keystores and SSL in the *Administration Console Online Help*.
3. When the login page appears, enter the user name and the password you used to start the Administration Server (you may have specified this user name and password during the installation process) or enter a user name that belongs to one of the following security groups: Administrators, Operators, Deployers, or Monitors. These groups provide various levels of access to system administration functions in the Administration Console. See "Protecting User Accounts" in *Managing WebLogic Security*.

   Using the security system, you can add or delete users to one of these groups to provide controlled access to the console. See "Protecting User Accounts" in *Managing WebLogic Security*.

Note: 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.

For information on using the Administration Console, see Using the Administration Console in the *Administration Console Online Help*.

Using WebLogic Server with Web Servers

You can proxy requests from popular Web servers to an instance of WebLogic Server or a cluster of WebLogic Servers by using one of the Web server plug-ins. Plug-ins are available for the following Web servers:

- Sun One Web Server or IPlanet
- Microsoft Internet Information Server

- Apache

Because these plug-ins operate in the native environment of the Web server, you manage the plug-ins through the administration facilities of that Web server.

For more information, see Using WebLogic Server with Plug-ins.

Special servlets are also included with the WebLogic Server distribution to proxy requests from an instance of WebLogic Server to another instance of WebLogic Server or to a cluster of WebLogic Servers. For more information, see:

- Configure Proxy Plug-ins in *Using Web Server Plug-Ins With WebLogic Server*
- Proxying Requests to a WebLogic Cluster in *Using WebLogic Server Clusters*

Monitoring

The system administration tools contain extensive capabilities for monitoring WebLogic Servers, domains, and resources. Using the tools you can monitor:

- Server health and performance:
  - Execute Queues
  - Connections
  - Sockets
  - Threads
  - Throughput
  - Memory Usage
- Security:
  - Locked-out users
  - Invalid Logins
  - Login attempts
- Transactions:
  - Committed transactions
  - Rolledback transactions
- JMS connections and servers

- WebLogic Messaging Bridge
- Applications:
  - Servlet sessions
  - Connector connection pools
  - EJB performance
- JDBC connections and connection pools

For more information, see "Monitoring a WebLogic Server Domain" on page 9-1.

Licenses

WebLogic Server requires a valid license to function. An evaluation copy of WebLogic Server is enabled for a limited time period so you can start using WebLogic Server immediately. To use WebLogic Server beyond the 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 five connections to the server.

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.

See "Installing and Updating WebLogic Platform License Files" in *Installing WebLogic Platform*.

CHAPTER 2

Overview of WebLogic Server Domains

The following sections describe WebLogic Server domains:

What Is a Domain?

A *domain* is the basic administration unit for WebLogic Server instances. A domain consists of one or more WebLogic Server instances (and their associated resources) that you manage with a single Administration Server. You can define multiple domains based on different system administrators' responsibilities, application boundaries, or geographical locations of servers. Conversely, you can use a single domain to centralize all WebLogic Server administration activities.

Each domain's configuration is stored in a separate configuration file called `config.xml`, which is stored on the Administration Server along with other files such as logs and security files. When you use the Administration Server to perform a configuration task, the changes you make apply only to the domain managed by that Administration Server. To manage another domain, use the Administration Server for that domain. For this reason, the servers instances, applications, and resources in one domain should be treated as being independent of servers, applications, and resources in a different domain.You cannot perform configuration or deployment tasks in multiple domains at the same time.

For more information, see

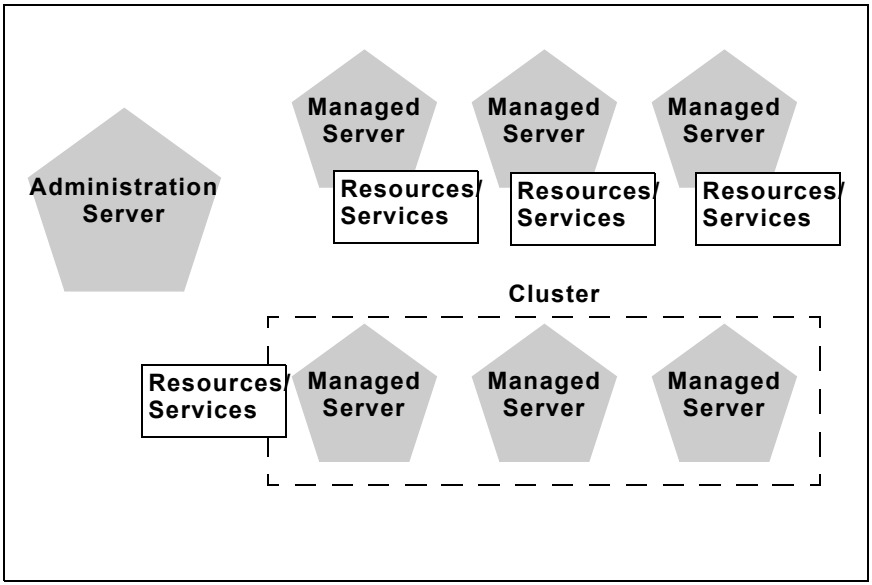
- Starting and Stopping Servers in the *Administration Console Online Help*
- "Starting the Administration Console" on page 1-20

Contents of a Domain

A domain can include multiple WebLogic Server clusters and non-clustered WebLogic Server instances. A minimal domain can contain only one WebLogic Server instance, which functions both as an Administration Server, and as a Managed server—such a domain can be useful while developing applications, but is not recommended for use in a production environment. Although the scope and purpose of a domain can vary significantly, most WebLogic Server domains contain the components described in this section.

Figure 2-1 shows a production environment that contains an Administration Server, three standalone Managed Servers, and a cluster of three Managed Servers.

Figure 2-1 WebLogic Server Domain



Production and Development Modes

You can configure servers in your domain to start in one of two modes, development or production. You use development mode while you are developing your applications. Development mode uses a relaxed security configuration and enables you to auto-deploy applications. You use production mode when your application is running in its final form. A production domain uses full security and may use clusters or other advanced features. For more information on the differences between development mode and production mode, refer to "Differences Between Configuration Startup Modes."

The runtime mode is a domain-wide setting. As each Managed Server starts, it refers to the mode of the Administration Server to determine its runtime mode. By default, all servers run in development mode. If you configure the domain to run in production mode, the Administration Server saves this setting to the domain's `config.xml` file.

BEA recommends that you set the mode when you first create a domain. For information on creating a domain that runs in production mode, see "Example: Creating a Domain with Administration Server and Clustered Managed Servers" on page 3-6.

It is possible, but not recommended, to change the mode of an existing domain. See Changing the Runtime Mode in *Administration Console Online Help*.

Creating a New Domain

This section discusses several ways to create a new domain. For information about restrictions for creating and using domains, see "Domain Restrictions" on page 2-10.

Server Name Considerations

Each server instance in your WebLogic environment must have a unique name, regardless of the domain or cluster in which it resides, or whether it is an Administration Server or a Managed Server. 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.

For WebLogic JMS, this strict unique naming rule also applies to JMS resources, such as JMS servers and stores in multi-domain environments when using the WebLogic Messaging Bridge or the Foreign JMS Server feature for intra-domain operability. For additional naming requirements, see "JMS Resource Naming Rules for Domain Interoperability" in the *Administration Console Online Help*.

JTA transactions are also subject to strict unique naming rules for inter-domain transactions. For more information see "Configuring Domains for Inter-Domain Transactions" in the *Administration Console Online Help*.

Creating a Domain Using the Configuration Wizard

The Configuration Wizard is the easiest and recommended way to create a new WebLogic Server domain. The configuration Wizard interactively queries you about the type of domain you want to create and creates the `config.xml` file, start scripts, and other files used in a domain. Using the Configuration Wizard you can:

- Create a domain with a single WebLogic Server instance for use in a development environment
- Create a domain with one or more Managed Servers
- Create a domain with one or more clusters
- Configure JDBC connection pools, DataSources, and MultiPools
- Configure JMS
- Add users to the security realm

- Create a domain based on a configuration template. A configuration template defines the full set of resources within a domain, including infrastructure components, applications, services, security options, and general environment and operating system options. You can use one of the provided configuration templates or create your own template. See "Configuration Template Builder" on page 1-11.

For more information, see :

- "Creating and Configuring Domains Using the Configuration Wizard" on page 3-1.
- Configuring WebLogic Platform

Creating a Domain Using the weblogic.Server Command

If you start a WebLogic Server instance from a directory that does not contain a `config.xml` file, the server automatically creates a default configuration, including start scripts and the `config.xml` file, in the directory from which you start the server instance. You can then modify the domain configuration using the Administration Console or other system administration tools.

To create a domain using the `weblogic.Server` command:

1. Open a command shell.
2. Set the `CLASSPATH` to include the WebLogic Server classes. The easiest way to set your `CLASSPATH` is to run the `setWLSEnv.cmd` (Windows) or the `setWLSEnv.sh` (UNIX) script. The script is located in the WebLogic Server installation at:

   ```
   BEA_Home/weblogic81/server/bin
   ```

3. Run the following command:

   ```
   java weblogic.Server
   ```

   When you are prompted for a username and password, enter any values you choose. You will be prompted with:

   ```
   Would you like the server to create a default configuration and boot?
   (y/n)
   ```

   Note: If you want to specify a name for your domain add the `-Dweblogic.Domain` option to the start command. For example, the command

   ```
   java -Dweblogic.Domain=Chicago weblogic.Server
   ```

   creates a domain called "Chicago".

4. Answer `Y`. You will be asked to confirm the password.

5. Enter the same password you entered in step 3. The server starts and creates a default `config.xml` file in the directory from which you ran the `java weblogic.Server` command.

The server creates a Boot Identity file (called `boot.properties`) that contains the username and password you entered in step 3. When this file is present in the domain's root directory, the server does not prompt for a username and password during startup.

Creating a Domain Using Ant Tasks

You can create a new domain configuration using Ant scripts by running the `wlserver` and `wlconfig` Ant tasks. For more information, see Using Ant Tasks to Configure a WebLogic Server Domain in the *WebLogic Server Command Reference*.

Administration Server

Each WebLogic Server domain must have one server instance that acts as the Administration Server. You use the Administration Server—programmatically, via the Administration Console, or using the command line—to configure all other server instances and resources in the domain.

Role of the Administration Server

Before you start the Managed Servers in a domain, start the Administration Server. When you start a standalone or clustered Managed Server, it contacts 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.

You can invoke the services of the Administration Server in the following ways:

- WebLogic Server Administration Console—The Administration Console is a browser-based graphical user interface (GUI) you use to configure a domain. For more information, see Using the Administration Console in *Administration Console Online Help*.
- WebLogic Server Application Programming Interface (API)—You can write a Java class to modify configuration attributes using the API provided with WebLogic Server. For more information, see Programming WebLogic Management Services with JMX.
- WebLogic Server command-line utility—This utility (`weblogic.Admin`) allows you to create scripts to automate domain management. For more information, see WebLogic Server Command Reference.

- SNMP—You can use Simple Network Management Protocol, versions SNMPv1 and SNMPv2 to monitor a WebLogic Server domain. For more information, see WebLogic SNMP Management Guide.

Whichever method is used, the Administration Server for a domain must be running to modify the domain configuration.

When the Administration Server starts, it loads the `config.xml` for the domain. It looks for `config.xml` in the current directory. When you create a new domain using the Configuration Wizard, you can specify the domain location, or by default the domain is placed at:

`BEA_HOME/user_projects/domains/mydomain`

where *mydomain* is a domain-specific directory, with the same name as the domain.

Each time the Administration Server starts successfully, and each time you modify the configuration, a backup configuration file is created. You can configure the number of backup copies of `config.xml` retained by the Administration Server. For more information, see "Backing up config.xml" on page 10-6.

What Happens If the Administration Server Fails?

The failure of an Administration Server for a domain does not affect the operation of Managed Servers in the domain. If an Administration Server for a domain becomes unavailable while the server instances it manages—clustered or otherwise—are up and running, those Managed Servers continue to run. If the domain contains clustered server instances, the load balancing and failover capabilities supported by the domain configuration remain available, even if the Administration Server fails.

If an Administration Server fails because of a hardware or software failure on its host machine, other server instances on the same machine may be similarly affected. However, the failure of an Administration Server itself does not interrupt the operation of Managed Servers in the domain.

For more information, see "Recovering Failed Servers" on page 10-1.

Managed Servers and Clustered Managed Servers

In a domain, server instances other than the Administration Server are referred to as Managed Servers. Managed Servers host the components and associated resources that constitute your applications—for example, JSPs and EJBs. When a Managed Server starts up, it connects to the domain's Administration Server to obtain configuration and deployment settings.

Note: Managed Servers in a domain can start up independently of the Administration Server if the Administration Server is unavailable. See "Recovering Failed Servers" on page 10-1 for more information.

Two or more Managed Servers can be configured as a WebLogic Server cluster to increase application scalability and availability. In a WebLogic Server cluster, most resources and services are deployed to each Managed Server (as opposed to a single Managed Server,) enabling failover and load balancing. To learn which component types and services can be clustered, see "What Types of Objects Can Be Clustered?" in *Using WebLogic Server Clusters*.

You create a non-clustered Managed Server and add it to a cluster by configuring pertinent configuration parameters for the server instance and the cluster using the Administration Console or other system administration tools. Conversely, you can remove a Managed Server from a cluster by re-configuring the parameters appropriately. The key difference between clustered and non-clustered Managed Servers is support for failover and load balancing—these features are available only in a cluster of Managed Servers.

Your requirements for scalability and reliability drive the decision on whether or not to cluster Managed Servers. For example, if your application is not subject to variable loads, and potential interruptions in application service are acceptable, clustering may be unnecessary.

For more information about the benefits and capabilities of a WebLogic Server cluster, see "Introduction to WebLogic Server Clustering" in *Using WebLogic Server Clusters*. A single domain can contain multiple WebLogic Server clusters, as well as Managed Servers that are not configured as clusters.

Resources and Services

In addition to the Administration Server and Managed Servers, a domain also contains the resources and services required by Managed Servers and hosted applications deployed in the domain.

Examples of domain-level resources include:

- Machine definition to identify a specific computer. A Machine definition is used to associate a computer with the Managed Server(s) it hosts. This information is used by Node Manager in restarting a failed Managed Server, and by a clustered Managed Server in selecting the best location for storing replicated session data. For more information about Node Manager, see "Overview of Node Manager" on page 4-1.
- Network channels, an optional resource that can be used to define default ports, protocols, and protocol settings. After creating a network channel, you can assign it to any number of

Managed Servers and clusters in the domain. See "Configuring Network Resources" on page 11-1 for more information.

Managed Servers in the domain host their own resources and services. You can deploy resources and services to selected Managed Servers or to a cluster. Examples of deployable resources include:

- Application components, such as EJBs, Web Applications, J2EE Connectors and Enterprise Applications
- Startup and shutdown classes
- JDBC connection pools
- JMS servers
- Virtual Hosts

Common Domain Types

There are two basic types of domains:

- **Domain with Managed Servers**: A simple production environment can consist of a domain with several Managed Servers that host applications, and an Administration Server to perform management operations. In this configuration, applications and resources are deployed to individual Managed Servers; similarly, clients that access the application connect to an individual Managed Server.

  Production environments that require increased application performance, throughput, or availability may configure two or more of Managed Servers as a cluster. Clustering allows multiple Managed Servers to operate as a single unit to host applications and resources. For more information about the difference between a standalone and clustered Managed Servers, see "Managed Servers and Clustered Managed Servers" on page 2-7.

- **Standalone Server Domain**: For development or test environments, you may want to deploy a single application and server independently from servers in a production domain. In this case, you can deploy a simple domain consisting of a single server instance that acts as an Administration Server and also hosts the applications you are developing. The examples domain that you can install with WebLogic Server is an example of a standalone server domain.

Note: In production environments, deploy applications only on Managed Servers in the domain. The Administration Server should be reserved for management tasks. For more

information, see Chapter 3, "Creating and Configuring Domains Using the Configuration Wizard."

Domain Restrictions

Many WebLogic Server installations consist of a single domain that includes all the Managed Servers required to host applications. Note the following restrictions for domains:

- Do not create a domain named "weblogic". This name is reserved for internal use by WebLogic Server.
- If you create more than one domain:
  - Each domain requires its own Administration Server for performing management activities. To manage another domain, start the Administration Console hosted by the Administration Server of that domain, or use the URL of the Administration Server in the command line arguments.
  - All Managed Servers in a cluster must reside in the same domain; you cannot "split" a cluster over multiple domains.
  - You cannot share a configured resource or subsystem between domains. For example, if you create a JDBC connection pool in one domain, you cannot use it with a Managed Server or cluster in another domain. (Instead, you must create a similar connection pool in the second domain.)
- Each server instance in your WebLogic environment must have a unique name, regardless of the domain or cluster in which it resides, or whether it is an Administration Server or a Managed Server. 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. For more information see, "Server Name Considerations" on page 2-4.

Directory Structure

This section discusses the directory structure for a domain and for server instances in a domain.

The config.xml file

The configuration of a domain is stored in the `config.xml` file for the domain. (You can specify another name for this file by specifying a command line option when you start the Administration server.) The `config.xml` file specifies the name of the domain and the configuration parameter settings for each server instance, cluster, resource, and service in the domain. The `config.xml`

file must exist in the Server Root Directory of the Administration Server. (See "Server Root Directory for an Administration Server" on page 2-13.)

This directory also contains default script files that you can use to start the Administration Server and Managed Servers in the domain. For more information about these scripts and other methods of starting a server instance, see "Starting and Stopping WebLogic Server in the *Administration Console Online Help*.

Domain Directory Structure

In releases of WebLogic Server prior to version 7.0, domain directories were created within the directory structure of the Weblogic Server installation. With WebLogic Server 8.1, you can set up domain directories outside the product installation directory tree, in any location on the file system that has access to the Weblogic Server installation and the JDK.

When you use the Configuration Wizard to create a domain, by default it creates the `user_projects/domains` directory in the BEA Home directory as a container for your domains (you can also specify a different directory anywhere on the file system). It also creates a root directory for the new domain and any other directories specified in the domain template that you select to create the domain.

The domain directory structure created by the Configuration Wizard contains:

- A domain root directory with the same name as the domain, such as `mydomain` or `petstore`. This directory contains the following:
  - The `config.xml` file for the domain
  - Scripts you use to start server instances and establish your environment
  - A subdirectory for storing applications for the domain, typically named `applications`.

  Note: If you plan to use the WebLogic Server's auto-deployment feature—available only when a domain is running in development mode—the subdirectory for applications *must* be named `applications`. For information about auto-deployment, see "Auto-Deployment" in *Developing WebLogic Server Applications*.

When you start a server instance in a domain for the first time, Weblogic Server creates the following subdirectories in the domain directory:

- Files containing security information.
- A`logs` directory for storing domain-level logs.

- A directory for each server running in the domain, for storing server logs and HTTP access logs. For more information, see "A Server's Root Directory" on page 2-12.

You can create other directories within the domain directory structure, as desired.

A Server's Root Directory

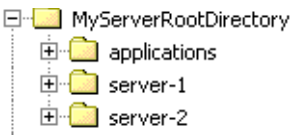
All instances of WebLogic Server use a *server root directory* to store runtime data and to provide the context for any relative pathnames in the server's configuration. When you start a WebLogic Server instance for the first time, a subdirectory is created for each server instance under the server root directory. These subdirectories contain log files, security data, administrative data, and other files used by the server instance.

You can specify the path and name of the server root directory for each server instance. You can specify a common server root directory for multiple server instances hosted on a single computer or you can specify a different server root directory for each server. A domain may have one or more server root directories.

The server root directory may also contain a subdirectory called `applications` that is used for auto-deployment of applications when a domain is running in development mode.

For example, in Figure 2-2, the server root directory is called `MyServerRootDirectory`. WebLogic Server instances named `server-1` and `server-2` each have their own subdirectory under the server root directory. The name of these subdirectories is the name of the server as defined in the domain's configuration.

Figure 2-2 Server Root Directory



Specifying a Server Root Directory

You can specify the path for the server root directory by one of the following means:

- Use the `-Dweblogic.RootDirectory=`*`path`* option when starting a WebLogic Server instance from command line. For example the following command:

  ```
  java -Dweblogic.RootDirectory=c:\MyServerRootDirectory weblogic.Server
  ```

  starts a WebLogic Server instance and uses `c:\MyServerRootDirectory` as the server root directory.

- If you use Node Manager to start a WebLogic Server instance, you can specify a server root directory with the Root Directory attribute in the Administration Console on the Server --> Configuration --> Remote Start tab.

If you do not use one of the above means to specify a server root directory, the path and name of the server root directory depend on whether a server instance is a Managed Server or the Administration Server and whether or not you use Node Manager to start the server instance. These variations are discussed in the next sections.

Server Root Directory for an Administration Server

An Administration Server uses its server root directory as a repository for the domain's configuration data (such as `config.xml`) and security resources (such as the default, embedded LDAP server).

To determine the root directory for an Administration Server, WebLogic Server does the following:

- If the server's startup command includes the `-Dweblogic.RootDirectory=`*`path`* option, then the value of *`path`* is the server root directory.
- If `-Dweblogic.RootDirectory=`*`path`* is not specified, and if the working directory (that is, the directory from which you issue the startup command) contains a `config.xml` file, then the working directory is the server root directory.
- If neither of the previous statements is true, then the server looks for a `config.xml` file in *`working-directory`*`/config/`*`domain-name`*. If it finds `config.xml` in this directory, then *`working-directory`*`/config/`*`domain-name`* is the server root directory.
- If WebLogic Server cannot find a `config.xml` file, then it offers to create one. You can use this method to create a new domain. For more information, see Using the weblogic.Server Command Line to Create a Domain in the *WebLogic Server Command Reference*.

Server Root Directory for a Managed Server Started with Node Manager

If you use the Node Manager to start a Managed Server, the root directory is located on the computer that hosts the Node Manager process. To determine the location of the server's root directory, WebLogic Server does the following:

- If you specified a root directory in the Administration Console on the Server --> Configuration --> Remote Start tab, then the directory you specified is the server root directory.

- If you did not specify a root directory in the Administration Console, then the server root directory is:
  *BEA_HOME**WL_HOME*\common\nodemanager
  where:
  - *BEA_HOME* is the directory where you installed one or BEA products, including WebLogic Server
  - *WL_HOME* is the directory in which you installed WebLogic Server on the Node Manager's host computer (weblogic81 by default)

The server root directory for a Managed Server started with Node Manager directory contains a subdirectory for each Managed Server instance. The name of the subdirectory is the name of the server as defined in the domain configuration.

Server Root Directory for a Managed Server Not Started with Node Manager

If you do not use the Node Manager to start a Managed Server (and therefore use the java weblogic.Server command or a script that calls that command), WebLogic Server does the following to determine the root directory:

- If the server's startup command includes the -Dweblogic.RootDirectory=*path* option, then the value of *path* is the server's root directory.
- If -Dweblogic.RootDirectory=*path* is not specified, then the working (current) directory is the root directory. For example, if you run the weblogic.Server command from c:\config\MyManagedServer, then c:\config\MyManagedServer is the root directory.

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

CHAPTER 3

Creating and Configuring Domains Using the Configuration Wizard

When you are ready to start developing or testing applications, use the Configuration Wizard to create and configure a domain. A *domain* is a collection of resources, such as servers, clusters, database connections, security services, and J2EE applications, that you manage as a unit. A domain must contain at least a definition for an Administration Server and an administrative user. You can define additional resources as part of the domain-creation process or at any time after creating the domain.

To create a domain, start the Configuration Wizard and choose a domain configuration template. Like a standard template, a *domain configuration template* provides a basic structure for a domain that you modify as your needs require.

The following sections describe creating and configuring domains:

For more information, refer to:

- "Extending Domains" in the *Creating WebLogic Configurations Using the Configuration Wizard* guide (adding resources to a domain that is inactive)
- "Creating, Configuring, and Monitoring Servers" in the Administration Console Online Help (adding resources to a domain that is currently active)

Starting the Configuration Wizard

You can start the Configuration Wizard any time after installing WebLogic Server. The wizard creates domains only on the computer on which it is installed; it cannot connect to remote hosts and create domains there. (After you create the domain, you can log on to a remote WebLogic Server host and use that host to run one or more servers that you created in the domain. For more information about starting servers, refer to "Starting and Stopping Servers" in the Administration Console Online Help.)

The installation program prompts you to start the Configuration Wizard. In addition, you can start the wizard by completing either of the following tasks:

- "Starting in a GUI Environment" on page 3-2
- "Starting in a Text-Based Environment" on page 3-3

Starting in a GUI Environment

To start and run the Configuration Wizard on a Windows computer, select the Configuration Wizard option from the BEA program group in the Windows Start Menu.

To start and run the Configuration Wizard in a GUI environment on a UNIX computer (or from a Windows command prompt):

1. Log in to a Windows or UNIX system on which the WebLogic Server software is installed.
2. Open a command-line shell.
3. Go to the following directory: `WL_HOME/common/bin`

   where `WL_HOME` is the directory in which you installed WebLogic Server. For example:

   ```
   cd ~/bea/weblogic810/common/bin
   ```
4. Invoke the `config.cmd` or `config.sh` script.

If you try to start the Configuration Wizard on a system that cannot support the graphical display, the wizard automatically starts in console mode.

Starting in a Text-Based Environment

To start the Configuration Wizard in a text-based environment (console mode):

1. Log in to a Windows or UNIX system on which the WebLogic Server software is installed.
2. Open a command-line shell.
3. Go to the following directory: `WL_HOME/common/bin`

   where `WL_HOME` is the directory in which you installed WebLogic Server. For example:

   ```
   cd ~/bea/weblogic810/common/bin
   ```

4. Invoke the `config.cmd` or `config.sh` script with the `-mode=console` argument. For example, in a `bash` shell on UNIX:

   ```
   . config.sh -mode=console
   ```

Choosing a Domain Configuration Template

On the first page of the Configuration Wizard, choose Create new WebLogic configuration and then click Next.

The second page of the Configuration Wizard asks you to choose a domain configuration template. A *domain configuration template* provides a basic structure for a domain that you modify as your needs require. You can choose templates that BEA provides or templates that your organization provides.

BEA Templates

Table 3-1 provides an overview of the WebLogic Server domain configuration templates. For more information about these and the templates that other BEA products provide, see "Template Reference" in the *Creating WebLogic Configurations Using the Configuration Wizard* guide.

Table 3-1 WebLogic Server Templates

Template	Description
Basic WebLogic Server Domain	Creates a basic WebLogic Server domain without the installation of any sample applications.
WebLogic Server Examples Domain	Creates the WebLogic Server Examples domain outside the installed kit. The WebLogic Server Examples domain contains a collection of examples that illustrate best practices for coding individual J2EE APIs.
Avitek Medical Records Sample Domain	Creates the Avitek Medical Records domain outside the installed kit. Avitek Medical Records is a WebLogic Server sample application suite that concisely demonstrates all aspects of the J2EE platform.

Additional Templates

Your organization can provide templates as well. For example, each development team can create a domain configuration template that defines JDBC connection pools and other resources that are specific to the application being developed.

To choose from additional templates, under Template Locations, click the Browse button and choose the directory in which your organization has located additional templates.

For information about creating templates, refer to "Creating Configuration Templates Using the Template Builder" and "Creating Extension Templates Using the Template Builder" in the *Creating WebLogic Configurations Using the Configuration Wizard* guide.

Completing the Remaining Steps of the Configuration Wizard

After you choose a domain configuration template, the Configuration Wizard prompts you to choose one of the following:

- Express Configuration, which accepts all default settings in the template.
- Custom Configuration, which enables you to modify the default settings.

For more information about completing the remaining steps, refer to the Configuration Wizard online help or to "Creating a New WebLogic Domain" in the *Creating WebLogic Configurations Using the Configuration Wizard* guide.

Example: Creating a Domain with a Single Server Instance

If you are in a development environment and want the simplest possible domain for developing your applications, you can use the Configuration Wizard to create a domain with a single server instance. In this domain, the single server you create is an Administration Server. In a development environment it is acceptable to deploy applications on the Administration Server; in a production environment, BEA recommends that you use Administration Servers only for managing the domain and you deploy applications only on Managed Servers.

To create a new domain with a single WebLogic Server instance:

1. Start the Domain Configuration Wizard. For more information, refer to "Starting the Configuration Wizard" on page 3-2.

   The Domain Configuration Wizard displays the Create or Extend a Configuration page.

   Note: The remaining instructions in this section assume that you are running the Domain Configuration Wizard in GUI mode.

2. On the Create or Extend a Configuration page, click Create a new WebLogic configuration button. Then click Next.
3. On the Select a Configuration Template page, choose the Basic WebLogic Server Domain template. Then click Next.
4. On the Choose Express or Custom Configuration page, choose Express. Then click next.
5. On the Configure Administrative Username and Password page, enter a user name and password. This user becomes the initial administrative user for the domain. Then click Next.

   See "Specifying an Initial Administrative User for a Domain" in the Administration Console Online Help.

6. On the Configure Server Start Mode and Java SDK page, accept the default selections and click Next.

   For information on server modes, see "Production and Development Modes" on page 2-3.

7. On the Create WebLogic Configuration page, click Create.

By default, the wizard creates a domain with the following characteristics:

- The domain is named mydomain.

- The domain contains a single server named myserver. This server functions as the Administration Server.
- The domain is created for a development environment.
- The root directory for the Administration Server is located in `WL_HOME\user_projects\domains\mydomain`.
- On Windows, you can start the Administration Server from the Start menu.
- The wizard does not install the server as a Windows service.
- The domain includes a `boot.properties` file.

To start the server in this domain, open a command prompt, change to `WL_HOME\user_projects\domains\mydomain`, and run the `startWebLogic.cmd` (Windows) or `startWebLogic.sh` (UNIX) script. For more information about starting servers, refer to "Starting and Stopping Servers" in the Administration Console Online Help.

To access the Administration Console for the domain, use any of the following URLs:

- From the same computer on which the server is running:
  `http://localhost:7001/console`
- From any computer:
  `http://`*`DNS-name`*`:7001/console`
  where *`DNS-name`* is the DNS name of the computer on which the server is running.
- From any computer:
  `http://`*`IP-Address`*`:7001/console`
  where *`IP-Address`* is an IP address of the computer on which the server is running.

For information about adding resources to this simple domain, refer to:

- "Extending Domains" in the *Creating WebLogic Configurations Using the Configuration Wizard* guide (adding resources to a domain that is inactive).
- "Creating, Configuring, and Monitoring Servers" in the Administration Console Online Help (adding resources to a domain that is currently active).

Example: Creating a Domain with Administration Server and Clustered Managed Servers

This type of domain configuration is recommended for production environments that require the improved availability and performance provided by failover and load balancing in a WebLogic

Server cluster. For more information about clusters, refer to "Introduction to WebLogic Clustering" in the *Using WebLogic Server Clusters* guide.

To create a new domain with a cluster of Managed Servers and an Administration Server (which is not part of the cluster):

1. Start the Domain Configuration Wizard. For more information, refer to "Starting the Configuration Wizard" on page 3-2.

   The Domain Configuration Wizard displays the Create or Extend a Configuration page.

   Note: The remaining instructions in this section assume that you are running the Domain Configuration Wizard in GUI mode.

2. On the Create or Extend a Configuration page, click Create a new WebLogic configuration button. Then click Next.
3. On the Select a Configuration Template page, choose the Basic WebLogic Server Domain template. Then click Next.
4. On the Choose Express or Custom Configuration page, choose Custom. Then click next.
5. On the Administration Server Configuration page, configure the Administration Server as follows:
   - **Name:** Enter an alphanumeric name that is unique for all configuration objects in the domain. Within a domain, each server, machine, cluster, JDBC connection pool, virtual host, and any other resource type must be named uniquely and must not use the same name as the domain.

     This field will not accept spaces.

     The server name is not used as part of the URL for applications that are deployed on the server. It is for your identification purposes only. The server name displays in the Administration Console, and if you use WebLogic Server command-line utilities or APIs, you use this name to identify the server.
   - **Listen Address**—To accept the default behavior, leave the `All Local Addresses` value. By default, remote processes can access server instances through the IP address or DNS name of the host machine. Local processes can access the server with the IP address, DNS name, or the `localhost` string.

     To change the default behavior, click in the field and choose a value from the drop-down list. See "Configuring the Listen Address" in the Administration Console Online Help.
   - **Listen Port:** Enter a numeric value for the listen port. The range is 1 to 65535.

See "Configuring the Listen Ports" in the Administration Console Online Help.

- **SSL Enabled**. Click in this box to enable SSL.

  By default, the server instance uses demonstration certificates for SSL communication. In a production environment, you must configure the server to use certificates from a certificate authority. See "Configuring the Listen Ports" in the Administration Console Online Help.

- **SSL Listen Port:** Enter a numeric value for SSL listen port. The range is 1 to 65535.

6. Click Next.
7. On the Managed Servers, Clusters and Machine Options page, click Yes. Then click Next.
8. On the Configure Managed Servers page, click Add and fill in the fields of the Add Server row as follows:

   - **Name**—Enter an alphanumeric name that is unique for all configuration objects in the domain. Within a domain, each server, machine, cluster, JDBC connection pool, virtual host, and any other resource type must be named uniquely and must not use the same name as the domain.

     This field will not accept spaces.

     Each server within a domain must have a unique name.

   - **Listen Address**—To accept the default behavior, leave the `All Local Addresses` value. By default, remote processes can access server instances through the IP address or DNS name of the host machine. Local processes can access the server with the IP address, DNS name, or the `localhost` string.

     To change the default behavior, choose a value from the drop-down list.

     For guidelines on supplying addressing information for a cluster and its members, see "Identify Names and Addresses" in the *Using WebLogic Server Clusters* guide.

   - **Listen Port:** Enter a numeric value for the listen port. The range is 1 to 65535.

     See "Configuring the Listen Ports" in the Administration Console Online Help.

   - **SSL Enabled**. Click in this box to enable SSL.

     By default, the server instance uses demonstration certificates for SSL communication. In a production environment, you must configure the server to use certificates from a certificate authority. See "Configuring the Listen Ports" in the Administration Console Online Help.

   - **SSL Listen Port:** Enter a numeric value for SSL listen port. The range is 1 to 65535.

9. Repeat step 8. to add additional Managed Servers, then click Next to move to the Cluster Configuration page.
10. On the Configure Clusters page, click Add and fill in the fields of the Add Cluster row as follows:
    - **Cluster Name:** Enter an alphanumeric name that is unique for all configuration objects in the domain. Within a domain, each server, machine, cluster, JDBC connection pool, virtual host, and any other resource type must be named uniquely and must not use the same name as the domain. This field will not accept spaces.
    - **MultiCast Address:** Enter a multicast address for the cluster. A multicast address is an IP address in the range from 224.0.0.0 to 239.255.255.255.
    - **MultiCast Port:** Enter a numeric value for the multicast port. The range of values is 1 to 65535.
    - **Cluster Address:** (Optional) Enter the cluster address. The cluster address forms the host name portion of URLs for requests directed to the cluster. If the cluster address is not set, EJB handles may not work properly. See "Cluster Address" in *Using WebLogic Server Clusters*.

      For production use, enter a DNS name that maps to the individual IP addresses of the Managed Servers in the cluster.

      For testing or development purposes, use a comma-separated list of the IP addresses and ports assigned to the Managed Servers (this is the default entry). For example `IPaddress1:port1,IPaddress2:port2,IPaddress3:port3`
11. Click Next to display the Assign Servers to Clusters page.
12. On the Assign Servers to Clusters page:
    a. In the Target list, select the cluster you created.
    b. In the Source list, select Managed Servers.

       Administration Servers cannot be part of a cluster.
    c. Click the right arrow button to assign the server to the cluster.
    d. Click Next to display the Configure Machines page.
13. On the Configure Machines page, complete the following steps for each WebLogic Server host that will run servers in the domain:
    a. Click the Add button.

b. In the Name column provide a name that identifies the computer that will run a server instance.

c. If you plan to use the Node Manager on the computer, in the Node Manager Listen Address column, enter the DNS name of the computer. In Node Manager Listen Port, enter the port on which the Node Manager listens for requests.

   The Administration Server uses the listen address and listen port to connect to the Node Manager running on the remote computer. The Administration Server can then ask the Node Manager to start a server instance on the computer.

For servers that are in a cluster, WebLogic Server uses the Machines that you configure to determine the optimum server to which certain tasks, such as HTTP session replication, are delegated.

14. Click Next to display the Assign Servers to Machines page.
15. On the Assign Servers to Machines page:
    a. In the Target list, select a Machine that you created.
    b. In the Source list, select a server.
    c. Click the right arrow to assign the server to the machine.
    d. When you complete all server associations, click Next.
16. Skip subsequent pages until the wizard displays the Configure Administrative Username and Password page.
17. On the Configure Administrative Username and Password page, enter a user name and password. This user becomes the initial administrative user for the domain. Then click Next.

    See "Specifying an Initial Administrative User for a Domain" in the Administration Console Online Help.
18. For Windows systems, the Configuration Wizard prompts you to create an item on the Windows Start menu. Click Yes if you want the wizard to lead you through the process of configuring a shortcut for the servers in your domain.
19. For Windows systems, the wizard also prompts you to install the Administration Server as a Windows service. Select Yes if you want the Administration Server to start automatically when you boot the Windows computer. See "Setting Up a WebLogic Server Instance as a Windows Service" on page 6-1.

Click Next to continue.

20. On the Configure Server Start Mode and Java SDK page, select Production Mode. Then click Next.

    For information on server modes, see "Production and Development Modes" on page 2-3.

21. On the Create WebLogic Configuration page, click Create.

By default, the wizard creates a domain with the following characteristics:

- The domain is named mydomain.
- The domain's Administration Server is named myserver.
- The root directory for the Administration Server is located in `WL_HOME\user_projects\domains\mydomain`.

To start the Administration Server in this domain, open a command prompt, change to `WL_HOME\user_projects\domains\mydomain`, and run the `startWebLogic.cmd` (Windows) or `startWebLogic.sh` (UNIX) script. To start Managed Servers, use the Node Manager or the `startManagedWebLogic` script. See "Starting and Stopping Servers" in the Administration Console Online Help.

To access the Administration Console for the domain, use any of the following URLs:

- From the same computer on which the server is running:
  `http://localhost:7001/console`
- From any computer:
  `http://`*`DNS-name`*`:7001/console`
  where *`DNS-name`* is the DNS name of the computer on which the server is running.
- From any computer:
  `http://`*`IP-Address`*`:7001/console`
  where *`IP-Address`* is an IP address of the computer on which the server is running.

CHAPTER 4

Overview of Node Manager

The Managed Servers in a production WebLogic Server environment are often distributed across multiple machines and geographic locations.

Node Manager is a Java utility that runs as separate process from WebLogic Server and allows you to perform common operations tasks for a Managed Server, regardless of its location with respect to its Administration Server. While use of Node Manager is optional, it provides valuable benefits if your WebLogic Server environment hosts applications with high availability requirements.

If you run Node Manager on a machine that hosts Managed Servers, you can start and stop the Managed Servers remotely using the Administration Console or from the command line. Node Manager can also automatically restart a Managed Server after an unexpected failure.

The following sections provide an overview of Node Manager.

To configure and use Node Manager, see "Configuring, Starting, and Stopping Node Manager" on page 5-1.

Node Manager Environment

These sections describe how Node Manager fits into your WebLogic Server environment.

Run Node Manager on Each Machine that Hosts Managed Servers

To take advantage of Node Manager capabilities, you must run a Node Manager process on each machine that hosts Managed Servers. You can manage multiple Managed Servers on a single machine with one Node Manager process—in Figure 4-1, the two Managed Servers on Machine C can be controlled by a single Node Manager process.

You cannot use Node Manager to start or stop an Administration Server. In a production environment, there is no need to run Node Manager on a machine that runs an Administration Server, unless that machine also runs Managed Servers. In a development environment, you may wish to run Node Manager on a machine that hosts an Administration Server and one or more Managed Servers, because doing so allows you to start the Managed Servers using the Administration Console.

Run Node Manager as an Operating System Service

The WebLogic Server installation process installs Node Manager to run as an operating system service: a daemon on UNIX machines, or a Windows service on Windows-based machines. An operating system service starts up automatically each time the operating system boots.

A key Node Manager feature is the ability to restart Managed Servers after a failure. If the failure is a machine crash, running Node Manager as a service ensures that Node Manager starts up automatically when the machine reboots, and is available to restart Managed Servers on that machine.

Node Manager is Domain-Independent

A Node Manager process is not associated with a specific WebLogic domain. Node Manager resides outside the scope of a domain, and you can use a single Node Manager process to start Managed Servers in any WebLogic Server domain that it can access—in Figure 4-1, Managed Server 2 and Managed Server 3 could be in separate domains, and controlled by a single Node Manager process. Node Manager is associated with Managed Servers using a Machine resource, as described in "Configure a Machine to Use Node Manager" on page 5-5.

Invoking Node Manager

You can invoke Node Manager's capabilities through the WebLogic Server Administration Console or JMX utilities, either the `weblogic.Admin` command-line utility or those you write

yourself. For more information about JMX, see "Programming WebLogic Management Services with JMX".

Typically, Node Manager is called by an Administration Server on a remote machine. However, Node Manager can also be called by a Administration Server on the same machine—allowing you to start co-resident Managed Servers using the Administration Console. When Node Manager is not running on a machine, the Start command on the Server—>Control tab page for Managed Servers on that machine is not enabled.

Node Manager Uses SSL

Node Manager communicates with Managed Servers and Administration Servers using two-way SSL to verify the identity of server instances that it communicates with. You cannot use Node Manager with an unsecured communication protocol.

The connection between Node Manager and a Managed Server is an always-open SSL connection. Node Manager uses this connection to send GETSTATE messages to the Managed Server. The Managed Server uses the connection to respond to each GETSTATE message.

The connection between Node Manager and an Administration Server is a short-lived request-response SSL connection.

Native Support for Node Manager

BEA provides native Node Manager libraries for Windows, Solaris, HP UX, Linux on Intel, Linux on Z-Series, and AIX operating systems. In addition, non-native Node Manager libraries are supported on these platforms.

For other UNIX and Linux operating Systems since support for only non-native Node Manager libraries is available, you must set the `weblogic.nodemanager.nativeVersionEnabled` option to `false` at the command line when starting Node Manager to use the pure Java version. For more information, see "Node Manager Properties" on page 5-10.

Native Node Manager is not supported on Open VMS, OS/390, AS400, UnixWare, or Tru64 UNIX.

Node Manager Capabilities

Node Manager enables you to perform these tasks:

- Start and stop remote Managed Servers.

- Monitor the self-reported health of Managed Servers and automatically kill server instances whose health state is “failed”.
- Automatically restart Managed Servers that have the “failed” health state, or have shut down unexpectedly due to a system crash or reboot.

The following sections describe these capabilities in more detail.

Start Managed Servers

Requests from the Administration Console (or JMX utilities such as `weblogic.Admin`) to start a Managed Server using Node Manager are issued to the Administration Server for the domain that contains the Managed Server. The Administration Server dispatches the start command to the Node Manager process on the machine that hosts the target Managed Server. Node Manager executes the start command and creates a Managed Server process. If the Managed Server does not respond within 60 seconds, the Node Manager sets the state of the Managed Server to `UNKNOWN`. Node Manager does not retry the start command. If the Managed Server successfully starts and establishes a connection with Node Manager, the state of the Managed Server is updated appropriately.

Node Manager starts a Managed Server using the startup arguments configured for the Managed Server in the Server—>Configuration—>Remote Start tab. The Managed Server is started in the startup mode configured on the advanced options portion of its Server—>Configuration—>General tab. By default the startup mode is `RUNNING`.

Note: Node Manager uses the same command arguments that you supply when starting a Managed Server using a script or at the command line. For information about startup arguments, see “weblogic.Server Command-Line Reference” in *WebLogic Server Command Reference*.

If you do not specify startup arguments for a Managed Server in its Remote Start tab, Node Manager uses its own properties as defaults to start the Managed Server. (See “Node Manager Properties” on page 5-10.) Although the Node Manager property values may suffice to boot a Managed Server, to ensure a consistent and reliable boot process, you should configure startup arguments for each Managed Server.

Node Manager starts a Managed Server in a dedicated process on the target machine, separate from the Node Manager and Administration Server processes, in the same directory where the Node Manager process is running. To run the Managed Server in a different directory, set the `Root Directory` attribute in the Server—>Configuration—>Remote Start tab.

The messages that would otherwise be output to `STDOUT` or `STDERROR` when starting a Managed Server are instead displayed in the Administration Console and written to the Node Manager log file for that server instance. For more information, see "Managed Server Log Files" on page 5-17.

Suspend or Stop Managed Servers

Requests from the Administration Console (or JMX utilities such as `weblogic.Admin)` to stop or suspend a Managed Server using Node Manager are issued to the Administration Server for the domain that contains the Managed Server. The Administration Server dispatches the command directly to the target Managed Server. If the Administration Server cannot reach the target Managed Server, the command is dispatched to the Node Manager process running on the target Managed Server's machine. The Node Manager forwards the request to the target Managed Server for execution. If the Managed Server fails to respond to a shutdown request from the Node Manager, the Node Manager process itself performs the shutdown.

Shut Down Failed Managed Servers

Node Manager periodically checks the self-reported heath status of Managed Servers that it has started. (For information about how a Managed Server monitors its health, see "Server Self-Health Monitoring" on page 9-2.) By default, Node Manager issues a health query to a Managed Server every 180 seconds.

Node Manager automatically kills a Managed Server that reports its health state as "failed", if the Managed Server's `Auto Kill If Failed` attribute is true. By default, the `Auto Kill If Failed` attribute is false. If you want Node Manager to restart a Managed Server that is hung, set `Auto Kill If Failed` to true, on the Server—>Configuration—>Health Monitoring tab for the Managed Server.

If a Managed Server does not respond to three consecutive health queries in a row, Node Manager considers the Managed Server to be "failed", and shuts it down, if `Auto Kill If Failed` is set.

For instructions on controlling the frequency with which Node Manager checks the health state of a Managed Server, see "Configure Monitoring, Shutdown, and Restart for Managed Servers" on page 5-7.

Restart of Crashed and Failed Managed Servers

By default, Node Manager automatically restarts Managed Servers that crash, and Managed Servers that Node Manager killed because their health state was "failed".

By default, Node Manager restarts a Managed Server no more than twice within a one-hour period. You can configure how many times Node Manager will restart a Managed Servers it controls, and the period of time over which it will do the restarts. For instructions, see "Configure Monitoring, Shutdown, and Restart for Managed Servers" on page 5-7.

Note: If you stop a Node Manager process that is currently monitoring Managed Servers, do not shut down those Managed Servers while the Node Manager process is shut down. Node Manager will be unaware of shutdowns performed on Managed Servers while it was down. When Node Manager is restarted, if a Managed Server it was previously monitoring is not running, it will automatically restart it.

Prerequisites for Automatic Restart of Managed Servers

For Node Manager to restart failed Managed Servers, the behavior must be configured appropriately, as described in "Configure Monitoring, Shutdown, and Restart for Managed Servers" on page 5-7. In addition, the following prerequisites apply:

- A Node Manager process can automatically monitor, shutdown, and restart only those Managed Servers that it started. It cannot provide these services for Managed Servers booted directly from the command line.
- If the event that caused a Managed Server to fail also causes Node Manager to fail, the Node Manager process on the machine where the Managed Server runs must be restarted, either by the operating system or manually, in order for Node Manager to restart the failed Managed Server. For this reason, BEA recommends that you run Node Manager as an operating system service.
- The Administration Server for the domain that contains the Managed Servers must run as a windows service, or on Unix systems, a daemon. For instructions, see "Setting Up a WebLogic Server Instance as a Windows Service" in *Configuring and Managing WebLogic Server*.

Node Manager Communications for Lifecycle Operations

This section describes the communications between a Node Manager client, a Node Manager process, and the Managed Servers managed by the Node Manager process.

Figure 4-1 and Table 4-1, "Communications Between Node Manager Process, Node Manager Client, and Managed Servers," on page 4-9 illustrate and describe the key communications that occur during Node Manager operation.

The interactions associated with specific lifecycle events are described in:

Figure 4-1 Node Manager Communications

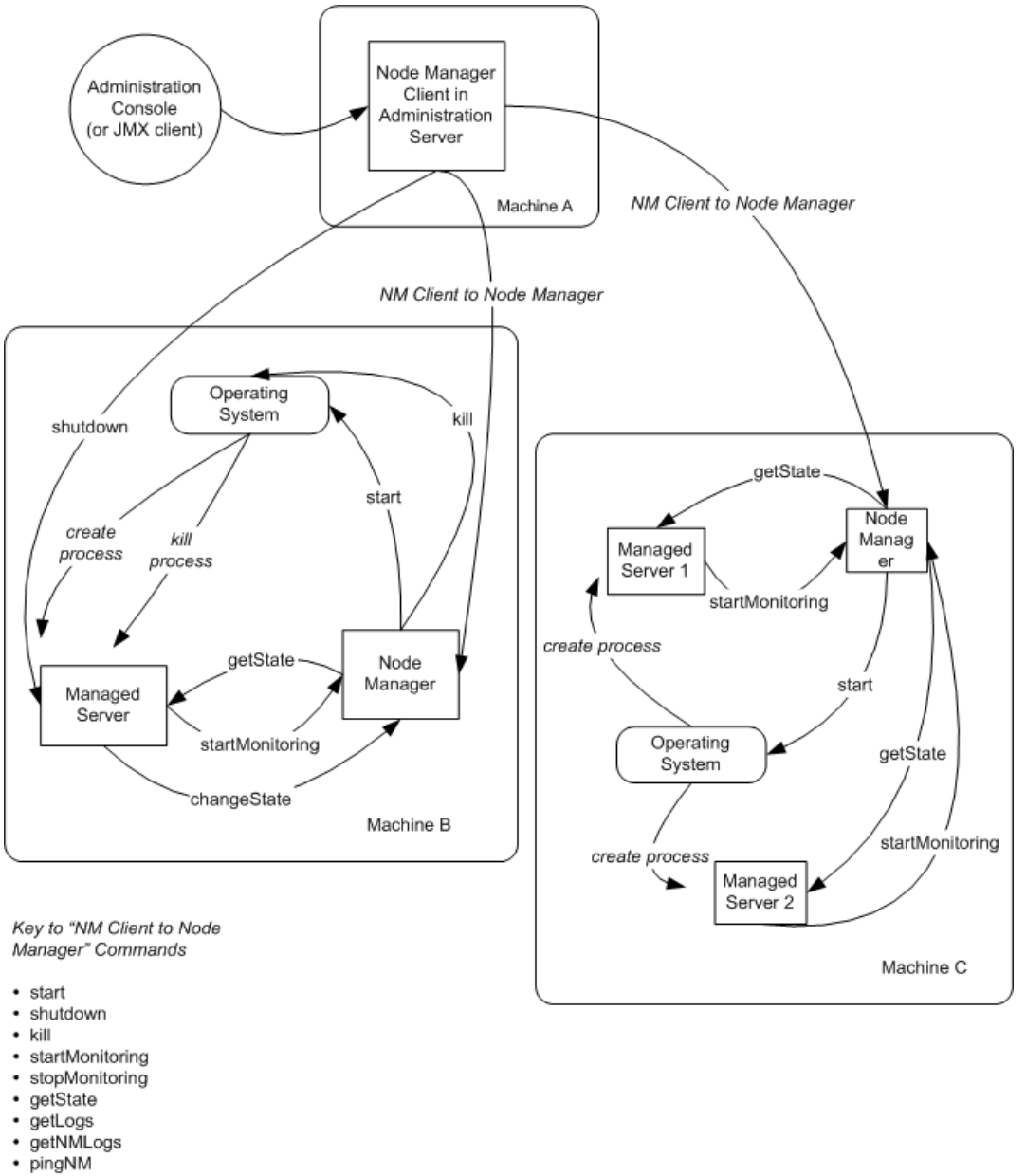


Table 4-1 Communications Between Node Manager Process, Node Manager Client, and Managed Servers

This component...	Has these interactions with a Node Manager process...	Description

Node Manager client in the Administration Server	`start`
	`shutdown`
	`kill`
	`startMonitoring`
	`stopMonitoring`
	`getState`
	`getLogs`
	`getNMLogs`
	`pingNM`
		Node Manager functions for the Managed Servers on a machine are invoked by a Node Manager client, either the client in the Administration Server for the domain that contains the Managed Servers, or a JMX client. Communications from the Node Manager client to the Node Manager process on a machine include:
		• lifecycle commands
		• commands to determine the availability of the Node Manager process and the health state of the Managed Servers under Node Manager control
		• requests for log files

Operating System	`start`	As a result of receiving a command to start a Managed Server from the Node Manager client, the Node Manager process issues a command to the operating system to create the Managed Server process.

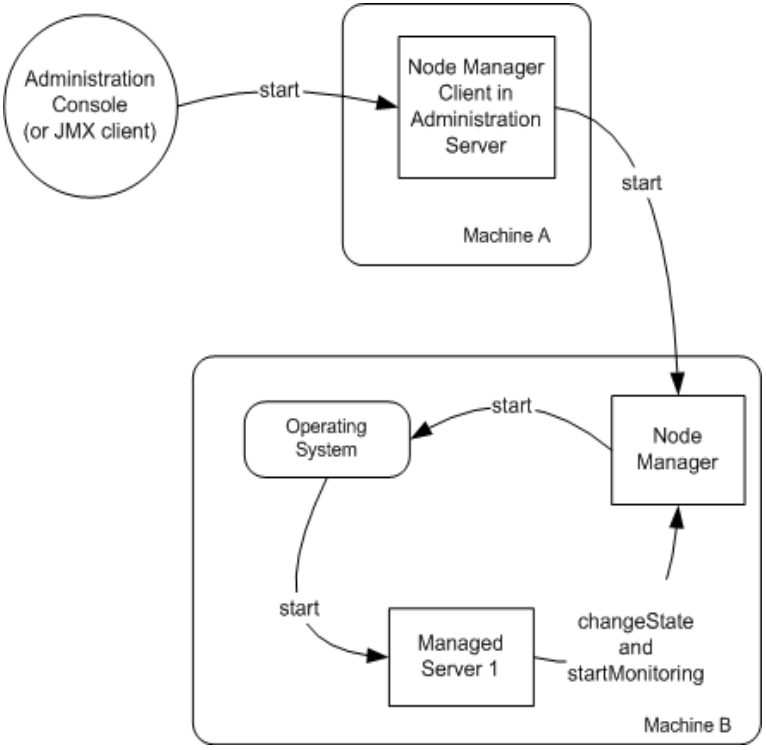
Managed Servers	`startMonitoring`
	`getState`
	`changeState`
	`shutdown`
		After the Managed Server process is created, and the Managed Server starts up, the Managed Server tells the Node Manager process that its state is `started`, and to commence monitoring.
		If the Managed Server determines that its health state has changed to `failed`, it informs the Node Manager process of the state change.
		Note: The Node Manager client sends shutdown commands directly to the target Managed Server. If the Managed Server does not respond, the shutdown command is sent to the Node Manager process that controls the Managed Server.

Node Manager Communications to Start a Managed Server

When the Node Manager process receives a command to start a Managed Server, it issues a request to the operating system to create the Managed Server process. When the Managed Server starts up, is tells the Node Manager process that it has started up, and to start monitoring its state.

If the Managed Server fails to connect back to the Node Manager process within the period of time specified by its `ScavengerDelaySeconds` attribute, the Node Manager process declares the state of the Managed Server to be `UNKNOWN`, and the task fails.

Figure 4-2 Starting a Managed Server with Node Manager



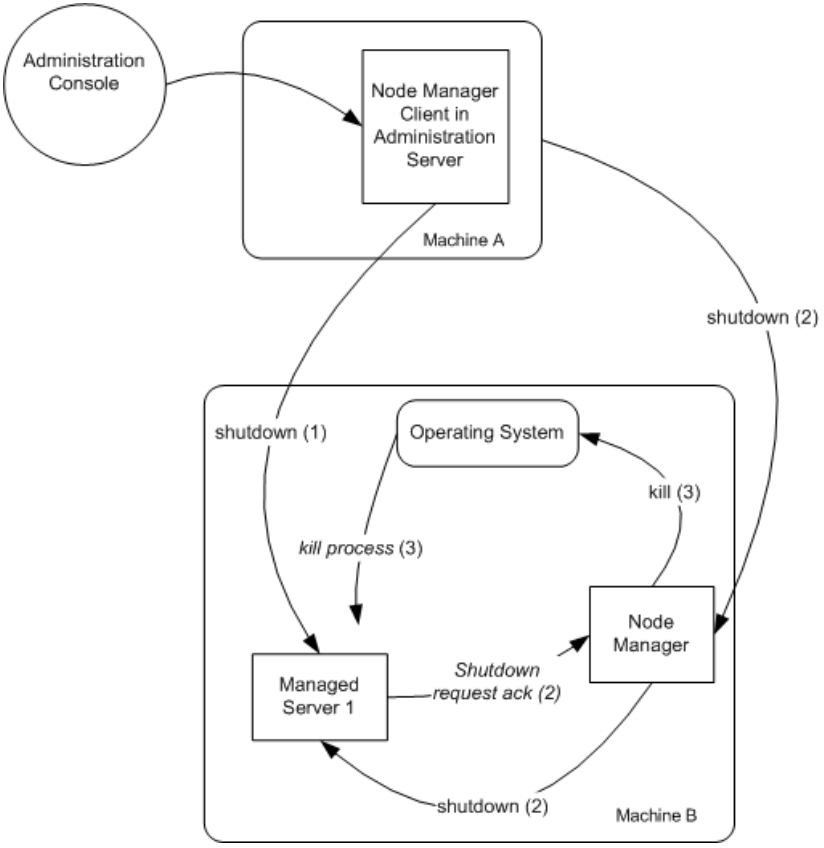
Node Manager Communications to Shut Down a Managed Server

Figure 4-3 illustrates the Node Manager communications involved in initiating a Managed Server shutdown. Depending on the state and availability of the Managed Server, Node Manager might need to try alternative strategies to successfully initiate the shutdown.

When an end-user issues chooses a shutdown action from the Administration Console, Node Manager proceeds in this fashion:

1. The Node Manager client in the Administration Server issues a `shutDown` directly to the Managed Server. If the Node Manager client successfully contacts the Managed Server, the Managed Server performs the shutdown sequence described in "Graceful Shutdown" on page 7-8. Otherwise, Node Manager tries the strategy described below in (2).
2. If the Managed Server is not reachable by the Node Manager client, the client issues the `shutDown` command to the Node Manager process on the machine that hosts the Managed Server. If the Managed Server is reachable by the Node Manager process, it acknowledges the command and performs the shutdown sequence described in "Graceful Shutdown" on page 7-8.Otherwise, Node Manager tries the strategy described below in (3).
3. If the Managed Server does not acknowledge the `shutDown`, the Node Manager issues a request to the operating system to kill the Node Manager process.

Figure 4-3 Shutting Down a Managed Server with Node Manager



Node Manager Communications to Restart a Managed Server

Figure 4-4 illustrates Node Manager communications involved in restarting a Managed Server whose state is `FAILED`.

Note: Node Manager only kills a Managed Server whose `AutoKillIfFailed` attribute is set to `true`. Node Manager only restarts a Managed Server whose `AutoRestart` attribute is set to `true`.

The Node Manager issues `GETSTATE` requests to the Managed Server periodically, as specified by the Managed Server's `HealthCheckIntervalSeconds` attribute.

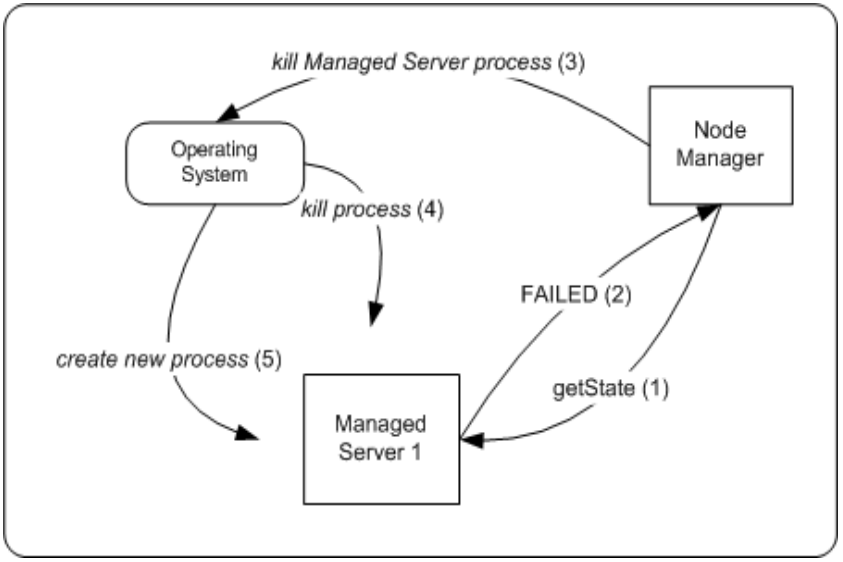
If the Managed Server reports its health as `FAILED` or if it does not respond to the `GETSTATE` within `ScavengerDelaySeconds`, the Node Manager process issues a request to the operating system to kill the Managed Server process.

After `RestartDelaySeconds`, the Node Manager process issues a request to the operating system to create a new Managed Server process.

When the Managed Server starts up, it contacts the Node Manager, as described in "Node Manager Communications to Start a Managed Server" on page 4-10.

If the Managed Server does not contact the Node Manager process, the Node Manager process will keep trying to create the process up to the number of attempts specified by the Managed Server's `RestartMax` attribute. The attempts occur periodically, as specified by the Managed Server's `RestartIntervalSeconds`.

Figure 4-4 Restarting a Managed Server with Node Manager



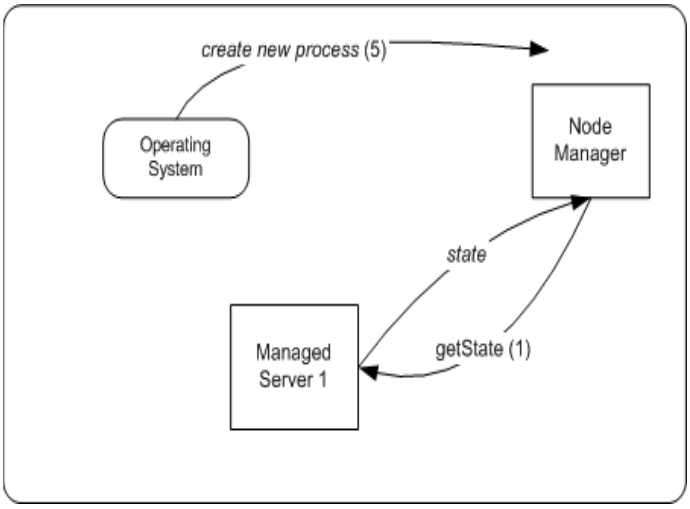
Node Manager Communications to Re-establish Communications After a Failure

Figure 4-4 illustrates Node Manager communications that occur when a Node Manager process starts up after it failed while monitoring a Managed Server.

After starting up, the Node Manager issues a `GETSTATE` requests to the Managed Server.

If the Managed Server reports status of `FAILED` or does not respond within `ScavengerDelaySeconds`, the Node Manager process proceeds as described in "Node Manager Communications to Restart a Managed Server".

Figure 4-5 Node Manager Process Re-establishing Communications After Failure



CHAPTER 5

Configuring, Starting, and Stopping Node Manager

For information about Node Manager features and functionality, see "Overview of Node Manager" on page 4-1. The following sections describe how to configure and use Node Manager.

Configuring Node Manager

This section describes the default configuration for Node Manager after installation, and the tasks required to configure Node Manager for a production environment.

Note: Each server instance in your WebLogic environment must have a unique name, regardless of the domain or cluster in which it resides, or whether it is an Administration Server or a Managed Server.

Default Configuration (Development Environment)

Node Manager is ready-to-run after WebLogic Server installation if you run Node Manager and the Administration Server on the same machine, and use the demonstration SSL configuration. By default, the following behaviors are configured:

- You can start a Managed Server using Node Manager through the Administration Console. If a Managed Server does not connect to the Node Manager within 180 seconds after Node Manager issues the start command, Node Manager sets the Managed Server's state to

UNKNOWN. Node Manager does not re-issue the start command. Even if the Managed Server takes longer than 180 seconds to establish a socket connection back, this does not affect the Managed Server's ability to communicate with Node Manager. When the Managed Server establishes the connection, Node Manager accepts it, and resets the Managed Server's state appropriately.

- Node Manager monitors the Managed Servers that it has started—it checks the self-reported health status of each Managed Server every 180 seconds. (For information about how a Managed Server monitors its health, see"Server Self-Health Monitoring" on page 9-2.) If a Managed Server does not respond to three consecutive health inquiries, Node Manager considers the Managed Server "failed".
- Automatic shutdown of Managed Servers is disabled. Node Manager will not kill Managed Server processes that are failed.
- Automatic restart of Managed Servers is enabled. Node Manager restarts:
  - Managed Servers that it killed or were killed by another method
  - Managed Servers that were running when a failure condition resulted in a system reboot up to two times within a 60 minute interval.

Configuration Checklist (Production Environment)

This section summarizes the tasks required to configure Node Manager for a production environment, and tailor its behavior.

- **Configure Node Manager to accept commands from remote hosts—**In a production environment, Node Manager must accept commands from remote hosts. Perform these tasks:
  a. "Set Up the Node Manager Hosts File" on page 5-3
  b. "Reconfigure Startup Service" on page 5-3
  c. "Review nodemanager.properties" on page 5-7
  d. "Configure a Machine to Use Node Manager" on page 5-5
  e. "Configure Managed Server Startup Arguments" on page 5-5
  f. "Ensure Administration Server Address is Defined" on page 5-6

- **Configure Node Manager for production security components**—By default Node Manager runs with demonstration security components. For information about using production security components, see "Configure SSL for Node Manager" on page 5-6.
- **Tailor monitoring and restart behavior—**To change Node Manager's monitoring, shutdown and restart behaviors, see "Configure Monitoring, Shutdown, and Restart for Managed Servers" on page 5-7.

Set Up the Node Manager Hosts File

Node Manager accepts commands from Administration Servers running on the same machine and on trusted hosts. Trusted hosts are identified by IP address or DNS name in the `nodemanager.hosts` file, which is created the first time you start Node Manager, in the directory where you run it.

Note: Each time you start Node Manager, it looks for `nodemanager.hosts` in the current directory, and creates the file if it does not exist in that directory.

You can specify the name and location of the trusted hosts file using the `trustedHosts` command-line argument. For more information, see "Node Manager Properties" on page 5-10.

By default, `nodemanager.hosts` is empty. To add trusted hosts, edit the file with a text editor, and add one line for each trusted host on which an Administration Server runs. If you want Node Manager to accept commands from any host, put an asterisk in the hosts file.

If you identify a trusted host by its DNS name, you must enable reverse DNS lookup when starting Node Manager. By default, reverse DNS lookup is disabled. Enable reverse DNS lookup in the `nodemanager.properties` file, or with the command-line argument:

```
ReverseDnsEnabled=true
```

Note: You do not have to restart Node Manager after changing the `nodemanager.hosts` file.

Reconfigure Startup Service

The WebLogic Server installation process installs Node Manager as an operating system service: a daemon on UNIX systems, or a Windows service on Windows systems. By default, the operating system service starts up Node Manager to listen on localhost:5555.

When you configure Node Manager to accept commands from remote systems, you must uninstall the default Node Manager service, then reinstall it to listen on a non-localhost Listen Address.

The directory *WL_HOME*\server\bin (where *WL_HOME* is the top-level directory for the WebLogic Server installation) contains uninstallNodeMgrSvc.cmd, a script for uninstalling the Node Manager service, and installNodeMgrSvc.cmd, a script for installing Node Manager as a service.

1. Delete the service using uninstallNodeMgrSvc.cmd.
2. Edit installNodeMgrSvc.cmd to specify Node Manager's Listen Address and Listen Port.

   Make the same edits to uninstallNodeMgrSvc.cmd as you make to installNodeMgrSvc.cmd, so that you can successfully uninstall the service in the future, as desired.
3. Run installNodeMgrSvc.cmd to re-install Node Manager as a service, listening on the updated address and port.

Daemonizing Node Manager For UNIX Systems

WebLogic Server does not provide command script for uninstalling and re-installing the Node Manager daemon process. Refer to your operating system documentation for instructions on uninstalling existing daemons, and setting up new ones.

1. Remove the Node Manager daemon process that the WebLogic Server installation process set up.
2. At the command line, or in a script, reinstall the Node Manager daemon. You may wish to view the contents of installNodeMgrSvc.cmd file before setting up the new daemon—although this command file is Windows-specific, it illustrates
   - Key environment and local variables that must be defined.
   - Validation steps you might want to include in a script that install Node Manager as a daemon.
   - Logic for setting default values for listen address and port.
3. To install Node Manager as a daemon, at the command line or in your script, you must, at a minimum:
   - Set WL_HOME
   - Set NODEMGR_HOME
   - Add the JDK and WebLogic directories to the system path.
   - Adds the JDK and WebLogic jars to the classpath.

- Set `LD_LIBRARY_PATH`
- Set `JAVA_VM`
- Set `NODEMGR_HOST`
- Set `NODEMGR_PORT`
- Set `PROD_NAME`=BEA WebLogic Platform 8.1

For variable definitions, see "Node Manager Environment Variables" on page 5-9.

Refer to your operating system documentation for operating system-specific settings that might be required.

Configure a Machine to Use Node Manager

In order for Node Manager to accept commands from remote Administration Servers, you must create a machine definition for each machine that runs a Node Manager process.

A machine definition associates a particular machine with the server instances it hosts, and specifies the connection attributes for the Node Manager process on that machine.

Create a machine definition on the Machine—>Configuration—>Node Manager tab in the Administration Console. Enter the DNS name or IP address upon which Node Manager listens in the Listen Address box.

Configure Managed Server Startup Arguments

Specify the startup arguments that Node Manager will use to start a Managed Server in the Server—>Configuration—>Remote Start tab for the Managed Server. If you do not specify startup arguments for a Managed Server in this fashion, Node Manager uses its own properties as defaults to start the Managed Server. Although these defaults are sufficient to boot a Managed Server, to ensure a consistent and reliable boot process, configure startup arguments for each Managed Server.

If you will run Node Manager as a Windows Service, as described in "Starting Node Manager as a Service" on page 5-8, you must configure the following JVM property for each Managed Server that will be under Node Manager control:

- `-Xrs` for the Sun JVM, or
- `-Xnohup` for the Jrockit

If you do not set this option, Node Manager will not be able to restart a Managed Server after a system reboot, due to this sequence of events:

1. A reboot causes a running Managed Server to be killed before the Node Manager and Administration Server operating system services are shutdown.
2. During the interval between the Managed Server being killed, and the Node Manager service being shutdown, Node Manager continues to monitor the Managed Server, detects that it was killed, and attempts to restart it.
3. The operating system does not allow restart of the Managed Server because the machine is shutting down.
4. Node Manager marks the Managed Server as failed, and it will not start this server when the machine comes up again.

Starting a Managed Servers with the `-Xrs` or `-Xnohup` option avoids this sequence of events by preventing the immediate shutdown of the Managed Server during machine shutdown.

For information about other startup arguments you can define for a server instance, see "Server—>Configuration—>Remote Start" in *Administration Console Online Help*.

Ensure Administration Server Address is Defined

Make sure that a Listen Address is defined for each Administration Server that will connect to the Node Manager process. If the Listen Address for a Administration Server is not defined, when Node Manager starts a Managed Server it will direct the target server to contact localhost for its configuration information.

Set the Listen Address using the Server-—>Configuration—>General tab in the Administration Console.

Configure SSL for Node Manager

Node Manager communicates with Administration Servers and Managed Servers using two-way SSL.

The default WebLogic Server installation includes demonstration Identity and Trust keystores that allow you to use SSL out of the box. The keystores—`DemoIdentity.jks` and `DemoTrust.jks`—are installed in *`WLSHome`*`/server/lib`. For testing and development purposes, the keystore configuration is complete.

Configuring SSL for a production environment involves obtaining identity and trust for the Node Manager and each Administration and Managed Server with which the Node Manager will be communicating and then configuring the Node Manager, the Administration Server, and any Managed Servers with the proper identity and trust. In addition, the use of host name verification

and the Administration port must be taken into consideration. To configure production SSL components, see "Configuring the SSL Protocol" in *Managing WebLogic Security*.

Review nodemanager.properties

In many environments, the SSL-related properties in `nodemanager.properties` may be the only Node Manager properties that you must explicitly define, as described in "Configure SSL for Node Manager" on page 5-6. However, `nodemanager.properties` also contains non-SSL properties in that you might need to specify, depending on your environment and preferences. For example:

- For a non-Windows installation, it might be appropriate to specify the `StartTemplate` and `NativeVersionEnabled` properties.
- If Node Manager runs on a multi-homed system, and you want to control which address and port it uses, define `ListenAddress` and `ListenPort`.
- To specify a non-default name and location for your trusted hosts file and node manager logs directory, define `TrustedHosts` and `SavedLogsDirectory`, respectively,

You can optionally set these and other options, including `JavaHome`, `WeblogicHome`, and `ReverseDNSEnabled` in `nodemanager.properties`. Review the property descriptions in "Node Manager Properties" on page 5-10 to determine properties that you wish to define.

As appropriate, update `nodemanager.properties` on each system on which Node Manager will run.

Note: `nodemanager.properties` is created in the directory where you start Node Manager the first time you start Node Manager after installation of WebLogic Server. Each time you start Node Manager, it looks for `nodemanager.properties` in the current directory, and creates the file if it does not exist in that directory.

You cannot access the file until Node Manager has started up once.

Configure Monitoring, Shutdown, and Restart for Managed Servers

Node Manager's default monitoring, shutdown and restart behaviors are described in "Default Configuration (Development Environment)" on page 5-1. To reconfigure the behavior, see "Configure Monitoring, Shutdown and Restart for Managed Servers" in *Administration Console Online Help*.

Note: These features are available when the conditions described in "Prerequisites for Automatic Restart of Managed Servers" on page 4-6 are met.

Starting and Stopping Node Manager

These sections describe methods of starting Node Manager, required environment variables, and command line arguments.

For prerequisite configuration steps see "Default Configuration (Development Environment)" on page 5-1 or "Configuration Checklist (Production Environment)" on page 5-2.

Starting Node Manager as a Service

The WebLogic Server installation process automatically installs Node Manager as a service, so that it starts up automatically when the system boots. By default, Node Manager will listen on localhost. If you want Node Manager accept commands from remote systems, you must uninstall the default Node Manager service, then reinstall it to listen on a non-localhost Listen Address. For more information, see "Reconfigure Startup Service" on page 5-3.

Starting Node Manager with Commands or Scripts

Although running Node Manager as an operating system service is recommended, you can also start Node Manager manually at the command prompt or with a script. The environment variables Node Manager requires are described in "Node Manager Environment Variables" on page 5-9. Command line options are described in "Node Manager Properties" on page 5-10.

Sample start scripts for Node Manager are installed in the *WL_HOME*`\server\bin` directory, where *WL_HOME* is the top-level installation directory for WebLogic Server. Use `startNodeManager.cmd` on Windows systems and `startNodeManager.sh` on UNIX systems.

The scripts set the required environment variables and start Node Manager in *WL_HOME*`/common/nodemanager`. Node Manager uses this directory as a working directory for output and log files. To specify a different working directory, edit the start script with a text editor and set the value of the `NODEMGR_HOME` variable to the desired directory.

Edit the sample start script to make sure that the command qualifiers set the correct listen address and port number for your Node Manager process.

Command Syntax for Starting Node Manager

In WebLogic Server 8.1, you can enter Node Manager properties on the command line or define them in the `nodemanager.properties` file, which is installed in the directory where you start Node Manager.

Values supplied on the command line override the values in `nodemanager.properties`.

The syntax for starting Node Manager is:

```
java [java_property=value ...] -D[nodemanager_property=value]
-D[server_property=value] weblogic.NodeManager
```

Note: WebLogic Server 8.1 provides a new wrapper to `weblogic.nodeManager.NodeManager`. The new wrapper is `weblogic.NodeManager`.

In the command line, a *`java_property`* indicates a direct argument to the `java` executable, such as `-ms` or `-mx`. If you did not set the `CLASSPATH` environment variable, use the `-classpath` option to identify required Node Manager classes.

Node Manager communicates with Administration Servers and Managed Servers using two-way SSL. When you start Node Manager, you must supply startup arguments that identify security components related to SSL communication. See "Node Manager Properties" on page 5-10 for information about all Node Manager command-line arguments.

Note: If you run 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 when starting Node Manager. For example, this command fails due to the space character in the name "`big iron`".

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

Node Manager Environment Variables

Before starting Node Manager, you must set several environment variables. You can set the environment variables for a domain in a start script or on the command line. The sample start scripts provided with WebLogic Server—`startNodeManager.cmd` for Windows systems and `startNodeManager.sh` for UNIX systems—set the required variables, which are listed in the following table.

Table 5-1 Node Manager Environment Variables

Environment Variable	Description
JAVA_HOME	Root directory of JDK that you are using for Node Manager. For example: `set JAVA_HOME=c:\bea\jdk131` Node Manager has the same JDK version requirements as WebLogic Server.
WL_HOME	WebLogic Server installation directory. For example: `set WL_HOME=c:\bea\weblogic700`
PATH	Must include the WebLogic Server `bin` directory and path to your Java executable. For example: `set PATH=%WL_HOME%\server\bin;%JAVA_HOME%\bin;%PATH%`

LD_LIBRARY_PATH (UNIX only)	For HP UX and Solaris systems, must include the path to the native Node Manager libraries.
	Solaris example:
	`LD_LIBRARY_PATH:$WL_HOME/server/lib/solaris:$WL_HOME/server/lib/solaris/oci816_8`
	HP UX example:
	`SHLIB_PATH=$SHLIB_PATH:$WL_HOME/server/lib/hpux11:$WL_HOME/server/lib/hpux11/oci816_8`

CLASSPATH	You can set the Node Manager `CLASSPATH` either as an option on the `java` command line used to start Node Manager, or as an environment variable. Windows NT example: `set CLASSPATH=.;%WL_HOME%\server\lib\weblogic_sp.jar;%WL_HOME%\server\lib\weblogic.jar`

Node Manager Properties

Node Manager properties define a variety of configuration settings for a Node Manager process. You can specify Node Manager properties on the command line or define them in the `nodemanager.properties` file, which is created in the directory where you start Node Manager the first time it starts up after installation of WebLogic Server. Values supplied on the command line override the values in `nodemanager.properties`.

Table 5-2, "Node Manager Properties," on page 5-11 describes Node Manager properties.

Table 5-2 Node Manager Properties

Node Manager Property	Description	Default
`CustomIdentityAlias`	Specifies the alias when loading the private key into the keystore. This property is required when the `Keystores` property is set as `CustomIdentityandCustom Trust` or `CustomIdentityAndJava StandardTrust`.	none
`CustomIdentityKey StoreFileName`	Specifies the file name of the Identity keystore (meaning the keystore that contains the private key for the Node Manager). This property is required when the `Keystores` property is set as `CustomIdentityand CustomTrust` or `CustomIdentity AndJavaStandardTrust`.	none
`CustomIdentityKey StorePassPhrase`	Specifies the password defined when creating the Identity keystore. This field is optional or required depending on the type of keystore. All keystores require the passphrase in order to write to the keystore. However, some keystores do not require the passphrase to read from the keystore. WebLogic Server only reads from the keystore, so whether or not you define this property depends on the requirements of the keystore.	none
`CustomIdentityKey StoreType`	Specifies the type of the Identity keystore. Generally, this is JKS. This property is optional	default keystore type from `java. security`
`CustomIdentityPrivate KeyPassPhrase`	Specifies the password used to retrieve the private key for WebLogic Server from the Identity keystore. This property is required when the `Keystores` property is set as `CustomIdentityand CustomTrust` or `CustomIdentityAndJavaStandard Trust`.	none
`CustomTrustKeyPass Phrase`	The password used to access the encrypted private key in the key file.	none

Node Manager Property	Description	Default
`CustomTrustKeyStore FileName`	Specifies the file name of the Trust keystore (meaning the keystore that contains the trusted CA certificates for the Node Manager). This property is required when the `Keystores` property is set as `Custom IdentityandCustomTrust`.	none
`CustomTrustKeyStore PassPhrase`	Specifies the password defined when creating the Trust keystore. This field is optional or required depending on the type of keystore. All keystores require the passphrase in order to write to the keystore. However, some keystores do not require the passphrase to read from the keystore. WebLogic Server only reads from the keystore, so whether or not you define this property depends on the requirements of the keystore.	none
`CustomTrustKeyStore Type`	Specifies the type of the Trust keystore. Generally, this is JKS. This property is optional.	default keystore type from java.security
`JavaHome`	The Java home directory that Node Manager uses to start a Managed Servers on this machine, if the Managed Server does not have a Java home configured in its Remote Start tab. If not specified in either place, Node Manager uses the Java home defined for the Node Manager process.	none

Node Manager Property	Description	Default
JavaStandardTrustKeyStorePassPhrase	Specifies the password defined when creating the Trust keystore. This field is optional or required depending on the type of keystore. All keystores require the passphrase in order to write to the keystore. However, some keystores do not require the passphrase to read from the keystore. WebLogic Server only reads from the keystore, so whether or not you define this property depends on the requirements of the keystore.This property is required when the Keystores property is set as CustomIdentityandJavaStandard Trust or DemoIdentityAndDemoTrust.	none
KeyStores	Indicates the keystore configuration the Node Manager uses to find its identity (private key and digital certificate) and trust (trusted CA certificates). Possible values are: • DemoIdentityAndDemoTrust Use the demonstration Identity and Trust keystores located in the BEA_HOME\server\lib directory that are configured by default. The demonstration Trust keystore trusts all the certificate authorities in the Java Standard Trust keystore (JAVA_HOME\jre\lib\security\cacerts) • CustomIdentityAndJavaStandard Trust Uses a keystore you create, and the trusted CAs defined in the cacerts file in the JAVA_HOME\jre\lib\security\cacerts directory. • CustomIdentityAndCustomTrust Uses Identity and Trust keystores you create.	DemoIdentityAndDemoTrust

Node Manager Property	Description	Default
`ListenAddress`	Any address upon which the machine running Node Manager can listen for connection requests. This argument deprecates `weblogic.nodemanager.listenAddress`.	null With this setting, Node Manager will listen on any IP address on the machine
`weblogic.nodemanager.listenAddress` (Deprecated)	The address on which Node Manager listens for connection requests. *Use `ListenAddress` in place of this deprecated argument.*	null With this setting, Node Manager will listen on any IP address on the machine
`ListenPort`	The TCP port number on which Node Manager listens for connection requests. This argument deprecates `weblogic.nodemanager.listenPort`.	5555
`weblogic.nodemanager.listenPort` (Deprecated)	The TCP port number on which Node Manager listens for connection requests. *Use `ListenPort` in place of this deprecated argument*.	5555
`NativeVersionEnabled`	When set to `true`, native Node Manager libraries are used for the operating system. Native Node Manager libraries are available only for Windows, Solaris, HP UX, Linux on Intel, Linux on Z-Series, and AIX operating systems. For other operating systems only non-native Node Manager libraries are available. For UNIX systems other the above operating systems, set this property to `false` to run Node Manager in non-native mode. This will cause Node Manager to use the start script specified by the `StartTemplate` property to start Managed Servers.	true

Node Manager Property	Description	Default
`ReverseDnsEnabled`	Specifies whether entries in the trusted hosts file can contain DNS names (instead of IP addresses).	false
`SavedLogsDirectory`	The path to directory where Node Manager stores log files. Node Manager creates a subdirectory in the `savedLogsDirectory` named `NodeManagerLogs`.	`./NodeManagerLogs`
`ScavangerDelaySeconds`	This is the period within which Node Manager expects a Managed Server it is starting to connect back. If the Managed Server does not connect back within this period, the state of the Managed Server is declared UNKNOWN and the task fails. When Node Manager is re-started after stopping while monitoring a Managed Server, it waits for this period of time for the Managed Server to connect back, otherwise, it will try to re-start the Managed Server. If Node Manager is running as a service, this feature is useful after a machine crash.	60 seconds
`StartTemplate`	For UNIX systems, specifies the path, relative to `WL_HOME`, of a script file used to start Managed Servers. For example, if you set `StartTemplate=mytemplate.sh`, `mytemplate.sh` should be in the `$WL_HOME` directory **Note:** Node Manager will use the specified start script to start Managed Servers only if the `NativeVersionEnabled` property is set to `false`. `StartTemplate` is used only for Unix systems that do not have native Node Manager libraries. If you create your own start template script, refer to `nodemenager.sh` as an example.	`./nodemanager.sh`

Node Manager Property	Description	Default
`TrustedHosts`	The path to the trusted hosts file that Node Manager uses. Node Manager will accept requests only from Administration Servers running on these hosts. Changes to this file do not require Node Manager restart. This file does not need to contain localhost. See "Set Up the Node Manager Hosts File" on page 5-3.	`./nodemanager.hosts`
`WeblogicHome`	Root directory of the WebLogic Server installation. This is used as the default value of `-Dweblogic.RootDirectory` for a Managed Server that does not have a root directory configured in its Remote Start tab. If not specified in either place, Node Manager starts the Managed Server in the directory where Node Manager runs.	none

Server Properties

Node Manager uses the server properties defined in the following table when starting a Managed Server. These values can be defined on the Remote Start tab for the Managed Server, or supplied on the command line when starting Node Manager. Values specified on the Remote Start tab take precedence over values specified on the command line.

Table 5-3 Server Properties

Server Property	Description	Default
bea.home	Specifies the BEA home directory that Managed Servers use on this machine.	none
java.security.policy	Specifies the path to the security policy file that Managed Servers use.	none

Stopping Node Manager

To stop a Node Manager process, close the command shell in which it is running.

If you stop a Node Manager process that is currently monitoring Managed Servers, do not shut down those Managed Servers while the Node Manager process is shut down. Node Manager will be unaware of shutdowns performed on Managed Servers while it was down. When Node Manager is restarted, if a Managed Server it was previously monitoring is not running, it will automatically restart it.

Troubleshooting Node Manager

The following sections describe how to diagnose and correct Node Manager problems. Use the Node Manager log files to help troubleshoot problems in starting or stopping individual Managed Servers. Use the steps in "Correcting Common Problems" on page 5-19 to solve problems in Node Manager configuration and setup.

Node Manager Log Files

Node Manager generates its own log files, which contain Node Manager startup and status messages. Node Manager log files are written to the `NodeManagerLogs/NodeManagerInternal` subdirectory of the directory where you start Node Manager. By default, Node Manager starts in *`WL_HOME`*`/common/nodemanager`—in which case Node Manager log files would be stored in *`WL_HOME`*`/common/nodemanager/NodeManagerLogs/NodeManagerInternal`

The log files are named `nm_hostname_`*`date-time`*`.log`, where *`date-time`* indicates the time at which Node Manager started.

Because Node Manager creates a new log file each time it starts, you should periodically remove the `NodeManagerLogs` subdirectory to reclaim the space used by old log files.

Managed Server Log Files

When you start a WebLogic Server instance, startup or error messages are printed to `STDOUT` or `STDERROR` and to the server log file. You can view the log file by right clicking on the server in the left pane of the Administration Console and selecting the option **View server log**, or by selecting the **View server log** link on any server tab page.

If you start a server instance with Node Manager, the server instance's startup and error messages are written to log files in the `NodeManagerLogs/`*`domain_serverName directory`*, where

domain_serverName designates the domain name and Managed Server name. NodeManagerLogs is a subdirectory of the directory where you start Node Manager. By default, Node Manager starts in *WL_HOME*/common/nodemanager—in which case Managed Server log files would be stored in *WL_HOME*/common/nodemanager/NodeManagerLogs/*domain_serverName*.

The NodeManagerLogs directory contains one subdirectory for each Managed Server started by the Node Manager process on that machine.

Logs files stored in the server directory include:

- *servername*_pid—Saves the process ID of the Managed Server named *servername*. Node Manager uses this information to kill the Managed Server, if requested by the Administration Server to do so.
- nodemanager.config—Saves startup configuration information passed to Node Manager from the Administration Server when starting a Managed Server.
- *servername*_output.log—Saves Node Manager startup messages generated when Node Manager attempts to start a Managed Server. If a new attempt is made to start the server, this file is renamed by appending _PREV to the file name.
- *servername*_error.log—Saves Node Manager error messages generated when Node Manager attempts to start a Managed Server. If a new attempt is made to start the server, this file is renamed by appending _PREV to the file name.

You can view the standard output and error messages for a server, as well as Node Manager's log messages for a particular Managed Server, on its Server-—>Monitoring-—>Remote Start Output tab.

Node Manager Client Logs

The NodeManagerClientLogs directory, which is created in the directory in which the Administration Server was started, contains log files used by the Node Manager client residing in the Administration Server.

The NodeManagerClientLogs directory contains a subdirectory for each Managed Server you attempted to start with Node Manager. Each log in these subdirectories corresponds to an attempt to carry out some action, such as starting or killing the server process. The name of the log file includes a timestamp that indicates the time at which the action was attempted.

Correcting Common Problems

The table below describes common Node Manager problems and their solutions

Table 5-4 Troubleshooting Node Manager.

Symptom	Explanation
Error message: `Could not start server 'MyServer' via Node Manager - reason: 'Target machine configuration not found'.`	You have not assigned the Managed Server to a machine. Follow the steps in "Configure a Machine to Use Node Manager" on page 5-5.
Error message: `<SecureSocketListener: Could not setup context and create a secure socket on 172.17.13.26:7001>`	The Node Manager process may not be running on the designated machine. See "Starting and Stopping Node Manager" on page 5-8.

Symptom	Explanation
I configured self-health monitoring attributes for a server, but Node Manager doesn't automatically restart the server.	To automatically reboot a server, you must configure the server's automatic restart attributes as well as the health monitoring attributes. See "Configure Monitoring, Shutdown, and Restart for Managed Servers" on page 5-7 and "Starting and Stopping Node Manager" on page 5-8. In addition, you must start Managed Servers using Node Manager. You cannot automatically reboot servers that were started outside of the Node Manager process (for example, servers started directly at the command line).

Symptom	Explanation

Applications on the Managed Server are using the wrong directory for lookups.	Applications deployed to WebLogic Server should never make assumptions about the current working directory. File lookups should generally take place relative to the Root Directory obtained with the `ServerMBean.getRootDirectory()` method (this defaults to the "." directory). For example, to perform a file lookup, use code similar to
	`String rootDir = ServerMBean.getRootDirectory(); //application root directory File f = new File(rootDir + File.separator + "foo.in");`
	rather than simply:
	`File f = new File("foo.in");`
	If an application is deployed to a server that is started using Node Manager, use the following method calls instead:
	`String rootDir //application root directory if ((rootDir = ServerMBean.getRootDirectory()) == null) rootDir = ServerStartMBean.getRootDirectory(); File f = new File(rootDir + File.separator + "foo.in");`
	The `ServerStartMBean.getRootDirectory()` method obtains the Root Directory value that you specified when configuring the server for startup using Node Manager. (This corresponds to the `Root Directory` attribute specified the Configuration—>Remote Start page of the Administration Console.)

Node Manager and Managed Server States

Node Manager defines its own, internal Managed Server states for use when restarting a server. If Node Manager is configured to restart Managed Servers, you may observe these states in the Administration Console during the restart process.

- FAILED_RESTARTING—Indicates that Node Manager is currently restarting a failed Managed Server.
- ACTIVATE_LATER—Indicates that MaxRestart restart attempts have been made in current RestartInterval, and Node Manager will attempt additional restarts in the next RestartInterval.
- FAILED_NOT_RESTARTABLE—Indicates that the Managed Server is Failed or was killed by Node Manager (as a result of the Managed Server's AutoKillIfFailed attribute being set to True), but Node Manager cannot restart the Managed Server because its AutoRestart attribute is set to False.

CHAPTER 6

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 computer, you can set up the server as a Windows service.

For each server instance 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, the Windows Service Control Manager (SCM), which is part of the Windows operating system, uses the information in the Windows Registry key to invoke the `weblogic.Server` main class. The Windows SCM cannot be configured to use a Node Manager to start Managed Servers, and therefore the Node Manager's monitoring and automatic restart features cannot be used for servers that run as a Windows service.

The following tasks set up and manage WebLogic Server instances that run as Windows services:

Setting Up a Windows Service: Main Steps

To set up a Windows service:

1. Do **one** of the following:
   - If the Configuration Wizard prompts you to install a server as a Windows service, choose Yes.

     Some of the domain templates in the Configuration Wizard prompt you to set up a server as a Windows service. If you choose Yes, the wizard installs the server as a Windows service. If you are using the wizard to create an Administration Server and Managed Servers, the wizard creates a Windows service only for the Administration Server. The wizard also creates a server-specific script that you can modify and use to install other servers as Windows services. The script is named *domain-name*`\installService.cmd`, where *domain-name* is the name of the domain that you created.
   - Create a script that sets values for server-specific variables and then calls a WebLogic Server master script. For more information, refer to "Creating a Server-Specific Script" on page 6-3.
2. If you are installing a Managed Server as a Windows service, add a variable to the server specific script that specifies the location of the domain's Administration Server. For more information, refer to "Configuring a Connection to the Administration Server" on page 6-5.
3. If you set up both an Administration Server and a Managed Server to run as Windows services on the same computer, modify the WebLogic Server master script so that the Managed Server starts only after the Administration Server finishes its startup cycle. For more information, refer to "Requiring Managed Servers to Start After Administration Servers" on page 6-6.
4. If you want a server instance to shut down gracefully when you use the Windows Control Panel to stop the Windows service, create a Java class and modify the master script so that the Windows SCM will invoke the class. For more information, refer to "Enabling Graceful Shutdowns from the Windows Control Panel" on page 6-8.
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 6-12.

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.

7. Run the server-specific script. For more information, refer to "Run the Server-Specific Script" on page 6-16.

Creating a Server-Specific Script

If the Configuration Wizard did not already create a server-specific script for your domain, you can create one. The script that you create must set values for variables that identify the name of the server instance and other server-specific information. Then it must call a master script, `WL_HOME\server\bin\installSvc.cmd`, where `WL_HOME` is the directory in which you installed WebLogic Server. The master scripts invokes the `beasvc` utility, which adds a key to the Windows Registry.

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

To see an example of a server-specific script, refer to Listing 6-1, "Example Script for Setting Up a Server as a Windows Service," on page 6-6.

To create a server-specific script:

1. In the root directory for the domain's Administration Server (the directory that contains the domain's `config.xml` file), create a text file.

2. Add the following, **required** `batch` commands to the text file, each command on a separate line:

   - `SETLOCAL`

     This is a `batch` command that begins the localization of environment variables in a batch file.

   - `set DOMAIN_NAME=`*`domain-name`*

     where *`domain-name`* is the name of your WebLogic Server domain.

   - `set USERDOMAIN_HOME=`*`absolute-pathname`*

     where *`absolute-pathname`* is the absolute pathname of the Administration Server's root directory (the directory that contains the domain's configuration file). For more information about the root directories for servers, refer to "A Server's Root Directory" on page 2-12.

- `set SERVER_NAME=`*`server-name`*

  where *`server-name`* is the name of an existing server instance that you want set up as a Windows service.

3. Add the following **optional** `batch` commands to the text file. Place each command on a separate line:

   - `set WLS_USER=`*`username`*
     `set WLS_PW=`*`password`*

     where *`username`* is the name of an existing user with privileges to start a server instance and *`password`* is the user's password. The `beasvc` utility encrypts the login credentials and stores them in the Windows registry.

     This is one of two possible methods for avoiding the username/password prompt when a server instance starts. The disadvantage to this method is that changing the username or password for the server instance requires you to delete the Windows service and set up a new one with the new username and password. Instead of this method, you can use a boot identity file. With a boot identity file, you can change the login credentials without needing to modify the Windows service. For more information, refer to "Boot Identity Files" in the *Administration Console Online Help*.

   - `set PRODUCTION_MODE=[`*`true`*`]`

     When the `PRODUCTION_MODE` variable is set to `true`, the server instance starts in production mode. When not specified, or when set to `false`, the server starts in development mode. For more information about development mode and production mode, refer to Chapter 3, "Creating and Configuring Domains Using the Configuration Wizard."

   - `set JAVA_OPTIONS=`*`java-options`*

     where *`java-options`* is one or more Java arguments that you want to pass to the Java Virtual Machine (JVM). Separate multiple arguments with a space. For a list of Java options that are specific to WebLogic Server, refer to "weblogic.Server Command-Line Reference" in the *WebLogic Server Command Line Reference*. The JVM that you use supports additional options and are documented by the JVM vendor.

   - `set JAVA_VM=-`*`JVM-mode`*

     where *`JVM-mode`* is a text string that indicates the mode in which you want the JVM to run. The values that you supply depend on the JVM that you are using. For example, the Sun JDK can run a `-hotspot`, `-client` or `-server` JVM. If you use the Sun JDK 1.3.1, the default value is `-hotspot`. If you use the Sun JDK 1.4.1, the default value is `-client`. If you use the JRockit JVM, the default value is `-jrockit`. For more

information, refer to "Starting and Configuring the JRockit JVM" in the JRockit *User Guide*.

– `set MEM_ARGS=[-Xms`*`Number`*`m] [-Xmx`*`Number`*`m]`

where *`Number`* is a numerical value in megabytes (MB). The `-Xms`*`Number`*`m` argument establishes a minimum heap size for the JVM and the `-Xmx`*`Number`*`m` sets a maximum heap size. By default, the minimum heap size is 23 MB and the maximum heap size is 200 MB.

Note: To specify a non-default JVM heap size, set the `MEM_ARGS` values in *`WL_HOME`*`\common\bin\commEnv.cmd`; however, this change affects all the domains under the same *`WL_HOME`*.

4. Add the following **required** commands to the end of the script:

   – `call "`*`WL_HOME`*`\server\bin\installSvc.cmd"`

   where *`WL_HOME`* is an absolute pathname for the directory in which you installed WebLogic Server. This command calls the WebLogic Server master script.

   – `ENDLOCAL`

   This is a `batch` command that ends the localization of environment variables in a batch file.

5. Save the text file with a `.cmd` extension. By default, the Windows command prompt associates the `.cmd` extension with `batch` files.

Configuring a Connection to the Administration Server

If you want to install a Managed Server as a Windows service, you must include a variable that specifies the location of the domain's Administration Server. The Managed Server must contact the Administration Server to retrieve its configuration data.

To configure a connection to the Administration Server:

1. In a text editor, open the server-specific script.

   If you are modifying a script that the Configuration Wizard created, BEA recommends that you make a copy of *`domain-name`*`\installService.cmd` (where *`domain-name`* is the name of the domain that you created) and open the copy.

2. In the text file, between the `SETLOCAL` command and the `call` command, create the following command:

   `set ADMIN_URL=`*`protocol`*`://`*`listen-address`*`:`*`listen-port`*

where

- *protocol* is `http` or `https`
- *listen-address* is a listen address of the Administration Server
- *listen-port* is a port of the Administration Server

For more information, refer to "Configuring a Connection to the Administration Server" in the *Administration Console Online Help*.

For an example, refer to the bold text in Listing 6-1.

3. Save your modifications to the server-specific script.

Listing 6-1 Example Script for Setting Up a Server as a Windows Service

```
echo off
SETLOCAL

set DOMAIN_NAME=myWLSdomain
set USERDOMAIN_HOME=d:\bea\user_projects\domains\myWLSdomain
set SERVER_NAME=myWLSserver
set PRODUCTION_MODE=true
set
JAVA_OPTIONS=-Dweblogic.Stdout="d:\bea\user_projects\domains\myWLSdomain\
stdout.txt" -Dweblogic.Stderr="d:\bea\user_projects\domains\myWLSdomain\
stderr.txt"

set ADMIN_URL=http://adminserver:7501

set MEM_ARGS=-Xms40m -Xmx250m

call "d:\bea\weblogic81\server\bin\installSvc.cmd"

ENDLOCAL
```

Requiring Managed Servers to Start After Administration Servers

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. Create a backup copy of the `WL_HOME\server\bin\installSvc.cmd` master script.

2. If you have already installed the Administration Server as a Windows service, remove the service. For more information, refer to "Removing a Server as a Windows Service" on page 6-18.
3. Before you install (or reinstall) the Administration Server as a Windows service, do the following:
   a. In a text editor, open the `WL_HOME\server\bin\installSvc.cmd` master script.

      The last command in this script invokes `beasvc`, which is the WebLogic Server utility that modifies the Windows Registry.

   b. In `installSvc.cmd`, 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 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:

      ```
      "%WL_HOME%\server\bin\beasvc" -install
      -svcname:"%DOMAIN_NAME%_%SERVER_NAME%"
      -delay:120000
      -javahome:"%JAVA_HOME%" -execdir:"%USERDOMAIN_HOME%"
      -extrapath:"%WL_HOME%\server\bin" -password:"%WLS_PW%"
      -cmdline:%CMDLINE%
      ```

      For more information about `beasvc`, enter the following command at a command prompt: *`WL_HOME`*`\server\bin\beasvc -help`, where *`WL_HOME`* 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 `WL_HOME\server\bin\installSvc.cmd` master script.
   b. In `installSvc.cmd`, 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:

```
"%WL_HOME%\server\bin\beasvc" -install
-svcname:"%DOMAIN_NAME%_%SERVER_NAME%"
-depend:"myDomain_myAdminServer"
-javahome:"%JAVA_HOME%" -execdir:"%USERDOMAIN_HOME%"
-extrapath:"%WL_HOME%\server\bin" -password:"%WLS_PW%"
-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. Create a Java class that invokes the `weblogic.management.runtime.ServerRuntime.shutdown()` method.

   This method gracefully shuts down a server after the server has completed all inflight work. For an example of such a class, refer to "Java Class that Shuts Down a Server Instance" on page 6-10.

2. Create a backup copy of the `WL_HOME\server\bin\installSvc.cmd` master script.

3. In a text editor, open the `WL_HOME\server\bin\installSvc.cmd` master script and do the following:

   a. Add the class that you created to the `set CLASSPATH` statement.

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

```
set
CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\server\lib\weblogic_sp
.jar;%WL_HOME%\server\lib\weblogic.jar;c:\myJar;%CLASSPATH%
```

b. Add the following argument to the last line of the script, which calls the `beasvc` utility:
`-stopclass:javaclass`
where *`javaclass`* is the full classpath name of the class that you created. This argument loads *`javaclass`* and then invokes its `public void static stop()` method.

For example, if you packaged the class in Listing 6-2 in `com.myClasses`, the modified `beasvc` command will be as follows:
the modified `beasvc` invocation will resemble the following:

```
"%WL_HOME%\server\bin\beasvc" -install
-svcname:"%DOMAIN_NAME%_%SERVER_NAME%"
-stopclass:com.myClasses.ServerStopper
-javahome:"%JAVA_HOME%" -execdir:"%USERDOMAIN_HOME%"
-extrapath:"%WL_HOME%\server\bin" -password:"%WLS_PW%"
-cmdline:%CMDLINE%
```

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

4. In the Administration Console, on the server's Control →Start/Stop tab, configure the Managed Server's graceful shutdown behavior.

   You can determine whether a graceful shutdown operation drops all HTTP sessions immediately and you can configure the amount of time that a graceful shutdown operation waits before forcing a shut down. For more information, refer to "Controlling Graceful Shutdowns" in the *Administration Console Online Help*.

5. 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.

6. Save your changes to the WebLogic Server master script.

Java Class that Shuts Down a Server Instance

The following Java class uses Java Management Extensions (JMX) to shut down a server instance. Each server uses JMX Managed Beans (MBeans) to expose its management attributes and operations. One such MBean, `ServerRuntime`, exposes a `shutdown()` method that gracefully shuts down a server.

The class in Listing 6-2 uses the Administration `MBeanHome` interface, which can retrieve and call `ServerRuntime` MBean operations for all server instances in a domain.

For more information about JMX programming, refer to the Programming WebLogic Management Services with JMX guide. For more information about the `ServerRuntime` MBean, refer to the WebLogic Server Javadoc.

Listing 6-2 Java Class that Shuts Down a Server Instance

```
import java.util.Set;
import java.util.Iterator;
import java.rmi.RemoteException;
import javax.naming.Context;
import javax.management.ObjectName;

import weblogic.jndi.Environment;
import weblogic.management.MBeanHome;
import weblogic.management.WebLogicMBean;
import weblogic.management.configuration.ServerMBean;
import weblogic.management.runtime.ServerRuntimeMBean;
import weblogic.management.runtime.ServerStates;
import weblogic.management.WebLogicObjectName;

public class ServerStopper {
  public static void stop() throws Exception {
    MBeanHome home = null;
```

```
    //url of the Admin server
    String url = "t3://qa113:7001";
    String username = "system";
    String password = "gumby1234";
    ServerRuntimeMBean serverRuntime = null;
    Set mbeanSet = null;
    Iterator mbeanIterator = null;

    try {
      // Set ContextClassloader to prevent assertions
      URL[] urls = { new File("/").toURL() };
      Thread.currentThread().setContextClassLoader(new
         URLClassLoader(urls));

      Environment env = new Environment();
      env.setProviderUrl(url);
      env.setSecurityPrincipal(username);
      env.setSecurityCredentials(password);
      Context ctx = env.getInitialContext();
      home = (MBeanHome)
           ctx.lookup("weblogic.management.adminhome");
      mbeanSet = home.getMBeansByType("ServerRuntime");
      mbeanIterator = mbeanSet.iterator();

      while(mbeanIterator.hasNext()) {
        serverRuntime = (ServerRuntimeMBean)mbeanIterator.next();
        if(serverRuntime.getState().equals(ServerStates.RUNNING)){
            serverRuntime.shutdown();
        }
      }

    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

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 its JVM print to standard out and standard error.

To view these messages, you must redirect standard out and standard error to a file:

1. Create a backup copy of the `WL_HOME\server\bin\installSvc.cmd` master script.
2. In a text editor, open the `WL_HOME\server\bin\installSvc.cmd` master script.
3. In `installSvc.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:

   ```
   "%WL_HOME%\server\bin\beasvc" -install
   -svcname:"%DOMAIN_NAME%_%SERVER_NAME%"
   -javahome:"%JAVA_HOME%" -execdir:"%USERDOMAIN_HOME%"
   -extrapath:"%WL_HOME%\server\bin" -password:"%WLS_PW%"
   -cmdline:%CMDLINE%
   -log:"d:\bea\user_projects\domains\myWLSdomain\myWLSserver-stdout.txt"
   ```

4. 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 6-13.

After you install the service and restart the Windows host, to view the messages that the server writes 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, using a DOS utility that supports the `tail` command, enter `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 3. 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.

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" in the *WebLogic Server Command Reference*.
- Open a command prompt and enter the following command:

`WL_HOME\bin\beasvc -dump -svcname:`*`service-name`*
where `WL_HOME` 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\weblogic81\server\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 use the WebLogic Server master script to install a server instance as a Windows service, the master 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. Create a backup copy of the `WL_HOME\server\bin\installSvc.cmd` master script.
2. In a text editor, open the `WL_HOME\server\bin\installSvc.cmd` master script.
3. Add your class to the `set CLASSPATH` statement.

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

   ```
   set
   CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\server\lib\weblogic_sp.
   jar;%WL_HOME%\server\lib\weblogic.jar;c:\myJar;%CLASSPATH%
   ```

   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.

   b. In the `WL_HOME\server\bin\installSvc.cmd` master script, find the `set CMDLINE` command.

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

      `-classpath @`*`pathname`*`\`*`filename`*
      where *`pathname`*`\`*`filename`* is the absolute path and name of the file that contains the classpath values.

      For example:

```
set CMDLINE="%JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-classpath @c:\myClasspath.txt -Dweblogic.Name=%SERVER_NAME%
-Dbea.home=\"D:\bea_70sp2\"
-Dweblogic.management.username=%WLS_USER%
-Dweblogic.management.server=\"%ADMIN_URL%\"
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Djava.security.policy=\"%WL_HOME%\server\lib\weblogic.policy\"
weblogic.Server"
```

4. Save your changes to the WebLogic Server master script.

Run the Server-Specific Script

Note: To run the server-specific script, you must log in to the Windows computer with a user account that has privileges to modify the Windows registry.

If you install the Windows service in a production environment, BEA recommends that you do **not** run the service under an operating-system user account that has administrator-level priveleges. For more information, see "Verifying the User Account Under Which the Service Runs" on page 6-17.

To run the server-specific script:

1. Open a command prompt and change to Administration Server's root directory, which is the directory that contains the server-specific script.
2. Enter the name of the server-specific script.

   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. If you modified the `WL_HOME\server\bin\installSvc.cmd` master script, consider undoing your modifications so the script can be used to set up other server instances.

Verifying the Setup

To verify that you successfully set up a WebLogic Server as a Windows service, do the following:

1. Open a command window and enter the following command:
   `set PATH=`*`WL_HOME`*`\server\bin;%PATH%`
2. Navigate to the directory immediately above your domain directory. For example, to verify the setup for *`BEA_HOME`*`\user_domains\mydomain`, navigate to *`BEA_HOME`*`\user_domains`.
3. Enter:
   `beasvc -debug "`*`yourServiceName`*`"`

   For example, `beasvc -debug "mydomain_myserver"`.

If your setup was successful, the `beasvc -debug` command starts your server. If the script returns an error similar to the following, make sure that you specified the correct service name:

```
Unable to open Registry Key .......
System\CurrentControlSet\Services\beasvc example_examplesServer\Parameters
```

Verifying the User Account Under Which the Service Runs

In a production environment, WebLogic Server Windows services should run under a special operating-system user account that has limited access privileges. For example, the OS user should have access privileges only to BEA files and to your domain files. This should be the only user account that has access to these files.

To ensure that the WebLogic Server instance runs under the special OS user account:

1. Open the Services control panel.

   For example, from the Windows 2000 desktop:

   a. Select the Start menu.

   b. On the Start menu, select Settings → Control Panel

   c. In the Control Panel window, open the Administrative Tools folder

   d. In the Administrative Tools window, open the Services control panel.
2. On the Services control panel, right click the WebLogic Server Windows service and click Properties.
3. In the Properties window, click the Log On tab.
4. Under Log on as, select This account. Then enter the user name and password of the special OS user account.

5. Click OK.

Note: When accessing network drives, the Windows service must run under the same username as the one who shared the network drive.

Using the Control Panel to Stop or Restart a Server Instance

After you set up a server instance to run as a Windows service, you can use the Service Control Panel to stop and restart the server.

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. For information on enabling graceful shutdowns from the Windows Control Panel, refer to "Enabling Graceful Shutdowns from the Windows Control Panel" on page 6-8.

To stop or restart a WebLogic Server instance that is installed as a Windows service:

1. Select Start→Settings→Control Panel.
2. On Windows 2000, open the Administrative Tools Control Panel. Then open the Services Control Panel.

   On Windows NT, open the Services Control Panel directly from the Control Panel window.
3. In the Services Control Panel, find the service that you created. By default, the service name starts with `beasvc`.
4. Right-click the service name and select commands from the shortcut menu.

Removing a Server as a Windows Service

To remove a Windows service that runs a WebLogic Server instance, you can use a script that causes the `beasvc` utility to remove the associated key from the Windows Registry. Removing the Windows service has no effect on the server instance's configuration that is saved in the domain's configuration file. After you remove the Windows service, you can start the WebLogic Server instance with start scripts or, for Managed Servers, the Node Manager.

If the Configuration Wizard did not already create a script for your domain, you can create one. The script sets values for variables that identify the name of the server instance and other server-specific information. Then the script calls a master uninstall script, `WL_HOME\server\bin\uninstallSvc.cmd`, where `WL_HOME` is the directory in which you

installed WebLogic Server. The master scripts invokes the `beasvc` utility, which removes a key from the Windows Registry.

To see an example of a server-specific uninstaller script, refer to Listing 6-3, "Script to Remove a Windows Service," on page 6-20.

To create a script for removing a Windows service that runs a WebLogic Server instance:

1. In the root directory for the domain's Administration Server (the directory that contains the domain's `config.xml` file), create a text file.
2. Add the following, **required** `batch` commands to the text file, each command on a separate line:
   - `SETLOCAL`

     This is a `batch` command that begins the localization of environment variables in a batch file.
   - `set DOMAIN_NAME=`*`domain-name`*

     where *`domain-name`* is the name of your WebLogic Server domain.
   - `set SERVER_NAME=`*`server-name`*

     where *`server-name`* is the name of an existing server instance that you want set up as a Windows service.
   - `call "`*`WL_HOME`*`\server\bin\uninstallSvc.cmd"`

     where *`WL_HOME`* is an absolute pathname for the directory in which you installed WebLogic Server. This command calls the WebLogic Server master uninstall script.
   - `ENDLOCAL`

     This is a `batch` command that ends the localization of environment variables in a batch file.
3. Save the text file with a `.cmd` extension. By default, the Windows command prompt associates the `.cmd` extension with `batch` files.
4. Enter the name of the server-specific script.

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

If the removal script runs successfully, it prints a line similar to the following to standard out: `mydomain_myserver removed.`

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

Listing 6-3 Script to Remove a Windows Service

```
echo off
SETLOCAL

set DOMAIN_NAME=myWLSdomain
set SERVER_NAME=myWLSserver
call "D:\bea\weblogic81\server\bin\uninstallSvc.cmd"

ENDLOCAL
```

Changing Startup Credentials for a Server Set Up as a Windows Service

To change a Windows service so that a WebLogic Server instance runs under different user credentials, do **one** of the following:

- If you set up the Windows service to retrieve usernames and passwords from a boot identity file, you can overwrite the existing file with a new one that contains the new username and password. You must specify the name of an existing user in the WebLogic Server default security realm. For information, refer to "Boot Identity Files" in the *Administration Console Online Help*.
- If you set up the Windows service to retrieve usernames and passwords from the Windows registry, then you must remove the Windows service and create a new one that uses your new username or password:

1. Uninstall the Windows service that runs the WebLogic Server instance. For more information, refer to "Removing a Server as a Windows Service" on page 6-18.
2. In a text editor, open the script that you used to install the service and enter the new username and password as the value for the `set WLS_USER` and `set WLS_PW` commands. WebLogic encrypts these values in the Windows Registry.
3. Save your modifications to the script.
4. Enter the name of the server-specific script.

   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.

5. (Optional) Remove the username and password from the script file.

CHAPTER 7

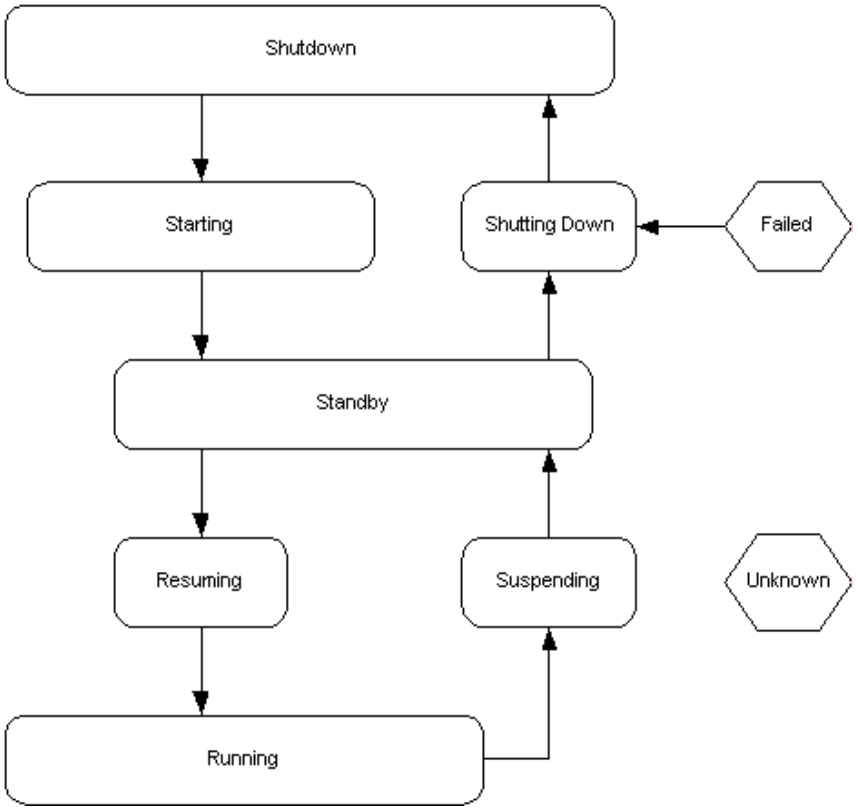
Server Life Cycle

At any time, a WebLogic Server instance is in a particular operating state. Commands—such as start, stop, and suspend—cause specific changes to the state of a server instance. The following sections describe WebLogic Server states, state transitions, life cycle commands.

Life Cycle Overview

The series of states through which a WebLogic Server instance can transition is called the *server life cycle*. Figure 7-1 illustrates the server life cycle and the relationships between WebLogic Server operating states.

Figure 7-1 The Server Life Cycle



To understand each state, and the relationships among states, see "Understanding WebLogic Server States" on page 7-2.

Understanding WebLogic Server States

WebLogic Server displays and stores information about the current state of a server instance, and state transitions that have occurred since the server instance started up. This information is useful to administrators who:

- Monitor the availability of server instances and the applications they host.
- Perform day-to-day operations tasks, including startup and shutdown procedures.

- Diagnose problems with application services.
- Plan correction actions, such as migration of services, when a server instance fails or crashes.

Getting Server State

There are multiple methods of accessing the state of a server instance:

- Administration Console—Multiple pages display state information:
  - The Servers table on the Servers page displays the current state of each server instance in the current domain.
  - The Server→Monitoring tab displays the state of the current server instance, and the date and time it entered the state.
  - The log file for a server, available from the **Server Log** command on all Server tabs, includes timestamped messages for state transitions that have occurred since the server instance was last started.
- Programmatically—You can obtain the state of a server instance programmatically, using the `getState()` method on the server's `weblogic.management.runtime.ServerRuntimeMBean`. For example, to monitor the progress of a long-running graceful shut down process, issue a `getstate` inquiry on a separate thread. For more information see "Accessing Runtime Information" in *Programming WebLogic Management Services with JMX*.
- Command Line Interface—For information about obtaining state information from a command line interface, see "GETSTATE" in *WebLogic Server Command-Line Interface Reference*.

Understanding Server State

The following sections describe the key states that a server instance can have, the processing associated with the state, and how the state fits into a sequence of state transitions.

SHUTDOWN

In the `SHUTDOWN` state, a server instance is configured but inactive. A server instance reaches the `SHUTDOWN` state as a result of a graceful shutdown or forced shutdown.

A graceful shutdown can be initiated when the server instance is in the `RUNNING` or the `STANDBY` state. The graceful shutdown process consists of the following state transitions:

RUNNING→SUSPENDING→STANDBY→SHUTTING DOWN→SHUTDOWN

See "Graceful Shutdown" on page 7-8.

A forced shutdown can be initiated from any server state. The forced shutdown process consists of the following state transitions:

any state→STANDBY→SHUTTING DOWN→SHUTDOWN

See "Forced Shutdown" on page 7-12.

See "Starting and Stopping Servers" in *Administration Console Online Help.*

STARTING

In the STARTING state, a server instance prepares itself to accept requests and perform application processing. A server instance cannot accept requests while in the STARTING state.

When a server instance starts itself, it retrieves its configuration data, starts its kernel-level services, initializes subsystem-level services, deploys applications, and loads and runs startup classes. For more information about the startup process, see "Start" on page 7-7.

A server instance can enter the STARTING state only from the SHUTDOWN state as a result of a **Start** command.

SHUTDOWN→STARTING→RUNNING

See "Starting and Stopping Servers" in *Administration Console Online Help.*

STANDBY

In the STANDBY state, a server has initialized all of its services and internal applications (such as the Administration Console), can accept administration commands, and can participate in cluster communication. Non-internal applications are partially deployed; when you move the server to the RUNNING state it completes the remainder of the deployment process. A server in the STANDBY state does not accept requests from external clients.

A server instance enters STANDBY when:

- It is started in standby mode with **Start** command. You can start a server instance in either Running or Standby mode. Configure Start Mode for a server instance on the advanced options portion of its Server→Configuration→General tab. The default Start Mode is Running. Starting a server instance in standby mode requires a domain-wide administrative port. For more information, see the following:
  - "Administration Port Requires SSL" on page 11-7.

– "Starting a Managed Server in the STANDBY State" in the Administration Console Help.

Starting a server instance in standby is a method of keeping it available as a "hot" backup, especially in a high-availability environment. A server instance started in standby can be quickly brought to the running state to replace a failed server instance.

A server instance started in standby mode goes through the following state transitions:

```
SHUTDOWN→STARTING→STANDBY
```

- Shut down, either gracefully or forcefully. A server instance transitions through the `STANDBY` state during any shutdown process.

  During the graceful shutdown process, a server instance goes through the following state transitions:

  ```
  RUNNING→SUSPENDING→STANDBY→SHUTTING DOWN→SHUTDOWN
  ```

  During the forceful shutdown process, a server instance goes through the following state transitions:

  ```
  any state→STANDBY→SHUTDOWN
  ```

- Running, during the regular startup process. During the regular startup process, a server instance goes through the following state transitions:

  ```
  SHUTDOWN→STARTING→STANDBY→RUNNING
  ```

RESUMING

A server instance enters the `RESUMING` state as a result of the **Resume** command. A server instance that is resumed from the `STANDBY` goes through the following state transitions:

```
STANDBY→RESUMING→RUNNING
```

RUNNING

When a server instance is in the `RUNNING` state, it offers its services to clients and can operate as a full member of a cluster. A server instance can enter the `RUNNING` state if:

- It is started using the **Start** command. During the regular startup process, a server instance goes through the following state transitions:

  ```
  SHUTDOWN→STARTING→STANDBY→RUNNING
  ```

- It is started with the **Resume** command. During the resume process, a server instance goes through the following state transitions:

  STANDBY→RESUMING→RUNNING

SUSPENDING

A server instance enters this state during the graceful shutdown process. While in the SUSPENDING state, the server handles a predefined portion of the work that is currently in process—referred to as "in-flight" work. The processing a server instance perform for in-flight work while in the SUSPENDING state is described in "In-Flight Work Processing" on page 7-10. Upon completion of in-flight work, the server progresses from the SUSPENDING state to the SHUTTING_DOWN state.

During the graceful shutdown process, a server instance goes through the following state transitions:

RUNNING→SUSPENDING→STANDBY→SHUTTING DOWN→SHUTDOWN

SHUTDOWN

A server instance enters the shutdown state as a result of a graceful shutdown or forced shutdown process.

During the graceful shutdown process, a server instance goes through the following state transitions:

RUNNING→SUSPENDING→STANDBY→SHUTTING DOWN→SHUTDOWN

See "Graceful Shutdown" on page 7-8.

During the forced shutdown process, a server instance goes through the following state transitions:

any state→STANDBY→SHUTDOWN

See "Forced Shutdown" on page 7-12.

FAILED

A server instance enters the FAILED state when one or more critical services become dysfunctional. When a server instance finds one or more critical subsystems have failed, the server instance sets its state to FAILED to indicate that the it cannot reliably host an application.

To recover from the FAILED state, a server instance must be shutdown and restarted, either manually, or automatically with Node Manager, if it is configured on the host machine. For information about automatic restarts, see "Shut Down Failed Managed Servers" on page 4-5.

A server instance can only enter the FAILED state from the RUNNING state:

RUNNING→FAILED

UNKNOWN

If a server instance cannot be contacted, it is in the UNKNOWN state.

States Defined by Node Manager

When Node Manager restarts a failed or killed Managed Server, it defines additional server states, which are described in "Node Manager and Managed Server States" on page 5-21. These states are displayed in the Administration Console, and provide visibility into the status of the restart process.

For information about Node Manager, see "Overview of Node Manager" on page 4-1.

Life Cycle Commands

This section describes key commands that affect the state of a server instance, and the processing a server instance performs as a result of the command. For information about issuing life cycle commands, see "Starting and Stopping Servers" in *Administration Console Online Help.*

For information about Node Manager processing related to key lifecycle events in environments that use Node Manager, see "Node Manager Communications for Lifecycle Operations" on page 4-6.

Start

When a server instance starts, it:

1. Retrieves its configuration data.

   An Administration Server retrieves domain configuration data, including security configuration data, from the `config.xml` for the domain.

   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.

   Initializes subsystem-level services with the configuration data that it retrieved in step 1. These services include the following:

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

3. If the server instance is in a domain for which a domain-wide administration port is configured, the server instance enables remote configuration and monitoring. For information about administration ports, see "Administration Port Requires SSL" on page 11-7.
4. Deploys modules in the appropriate container and in the order that you specify in the WebLogic Server Administration Console.

   Startup classes that are configured to load before application deployments are loaded and run after the server instance deploys JDBC connection pools and before it deploys Web applications and EJBs.
5. Startup classes that are configured to load after application deployments are loaded and run.

Graceful Shutdown

A graceful shutdown gives WebLogic Server subsystems time to complete certain application processing currently in progress. The work that it completes during graceful shutdown is referred to as *in-flight work*. During a graceful shutdown, subsystems complete in-flight work and suspend themselves in a specific sequence and in a synchronized fashion, so that back-end subsystems like JDBC connection pools are available when front-end subsystems are suspending themselves.

Note: The graceful shutdown process synchronizes shutdown operations and in-flight work for WebLogic execute queue threads. Applications that create threads must control the

in-flight processing for pending work in application threads; this can be accomplished in a shutdown class.

Graceful Shutdown Sequence

The following list shows the order in which subsystems suspend themselves. Each subsystem completes its in-flight work before the next one commences its preparation to suspend.

1. Non-Transaction RMI Service
2. Web Container
3. Client Initiated Transaction Service
4. Remote RMI Service
5. Timer Service
6. Application Service
7. EJB Container
8. JMS Provider
9. JDBC Container
10. Transaction Service

Controlling Graceful Shutdown

`ServerMBean` has two new attributes for controlling the length of the graceful shutdown process. Their values are displayed and configurable on the Server→Control→Start/Stop tab:

- Ignore Sessions During Shutdown— If you enable this option WebLogic Server will drop all HTTP sessions immediately, rather than waiting for them to complete or timeout. Waiting for abandoned sessions to timeout can significantly lengthen the graceful shutdown process, because the default session timeout is one hour.
- Graceful Shutdown Timeout—Specifies a time limit for a server instance to complete a graceful shutdown. If you supply a timeout value, and the server instance does not complete a graceful shutdown within that period, WebLogic Server performs a forced shutdown on the server instance.

In-Flight Work Processing

The following sections describe how each subsystem handles work in process when it suspends itself during a graceful shutdown.

RMI Subsystem

The Remote Method Invocation (RMI) subsystem suspends in three steps. Each step in this process completes before the following step commences.

1. Non-transaction remote requests are rejected by the Non-Transaction RMI Service.
2. The Client Initiated Transaction Service waits for pending client transactions to complete.
3. The Remote RMI Service rejects all remote requests with or without transactions.

After these steps are completed, no remote client requests are allowed. Requests with administrative privileges and internal system calls are accepted.

When a clustered server instance is instructed to prepare to suspend, the RMI system refuses any in-memory replication calls, to allow other cluster members to choose new hosts for replicated sessions.

Web Container

After the Web Container subsystem is instructed to prepare to suspend, it rejects new sessions requests. Existing sessions are handled according to the persistence method:

- No persistence—Pending sessions with no persistence are allowed to complete.
- In-memory replication in a cluster—Sessions with secondary sessions are immediately suspended. If a primary session does not have a secondary session, the Web Contains waits until a secondary session is created, or until the session times out, whichever occurs first.
- JDBC persistence and file persistence—The Web Container immediately suspends sessions that are replicated in a database or file store.

The completion of pending sessions is optional. To drop all sessions immediately, use the Ignore Sessions During Shutdown option on the Servers→Control→Start/Stop tab in the Administration Console, or the `-ignoreSessions` option with `weblogic.Admin`.

Timer Service

The Timer Service cancels all triggers running on application execute queues. Application execute queues include the default queue and queues configured through the `ExecuteQueueMBean`.

Application Service

The Application Service completes pending work in the application queues before suspending. Application execute queues include the default queue and queues configured through the `ExecuteQueueMBean`.

EJB Container

The EJB Container suspends Message Drive Beans (MDBs.)

JMS Service

The Java Messaging Service (JMS) marks itself as suspending, which causes new requests to be rejected. The JMS system suspends gracefully in this fashion:

If the server instance being shutdown has a JMS server:

- Any send requests that are waiting because of message quotas are returned immediately.
- All consumers on destinations belonging the JMS Server are closed.
- The paging store is closed.

If the server instance being shutdown has a JMS connection factory:

- Client connections are closed.

Generally each step in the graceful suspend of the JMS subsystem occurs quickly—in less than a second. Potentially, completion of a client request could take longer, if the request requires higher than normal disk I/O, for example, a request for a persistent "send" of a 100-megabyte message.

You can monitor the number of connections to a JMS server, the number of consumers to a JMS connection factory, and related run-time information using JMS run-time Mbeans, including JMSRuntimeMbean, JMSConnectionRuntimeMBean, JMSConsumerRuntimeMBean.

JDBC Service

The JDBC Service closes idle connections in the connection pools.

Note: If connections are still in use, the shutdown of the JDBC service will fail, and the graceful shutdown will not complete. To shut down a server instance while applications still hold connections, use a forced shutdown command, described in "Forced Shutdown" on page 7-12.

Transaction Service

The Transaction Service waits for the pending transaction count in the Transaction Manager to drop to zero before suspending. Completing all pending transactions can be a lengthy process, depending on the configured transaction timeout.

If a graceful shutdown takes too long because of pending transactions, you can halt it with a forced shutdown command. A forced shutdown suspends all pending work in all subsystems.

Shutdown Operations and Application Undeployment

During both graceful and forced shutdown, subsystems undeploy applications as appropriate. This processing can result in invocation of application code, such as servlet `destroy()` or `ejbRemove()` during shutdown. During the shutdown sequence, JMS, JDBC, and transactions are shutdown *after* applications are shutdown, allowing application code to access JMS, JDBC, and transaction services.

Forced Shutdown

A forced shutdown is immediate—WebLogic Server subsystems suspend all application processing currently in progress. A forced shutdown can be performed on a server instance in any state.

If a fatal exception causes the forced shutdown to fail, the server will exit after the number of seconds specified by the `ServerLifecycleTimeoutVal` attribute in `ServerMBean`.

For information about undeployment processes during a forced shutdown, and related programming considerations, see "Shutdown Operations and Application Undeployment".

CHAPTER 8

Configuring Web Server Functionality for WebLogic Server

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

Overview of Configuring Web Server Components

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

Table 8-1 HTTP Operating Parameters

Attribute	Description	Range of Values	Default Value
Default Server Name	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
Post Timeout	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 Size	Max Post Size (in bytes) for reading HTTP POST data in a servlet request. MaxPostSize < 0 means unlimited	Bytes	-1

Table 8-1 HTTP Operating Parameters

Attribute	Description	Range of Values	Default Value
Enable Keepalives	Whether or not HTTP keep-alive is enabled.	Boolean True = enabled False = not enabled	True
Duration (labeled Keep Alive Secs on the Virtual Host panel)	Number of seconds to maintain HTTP Keep Alive before timing out the session.	Integer	30
HTTPS Duration (labeled Https Keep Alive Secs on the Virtual Host panel)	Number of seconds to maintain HTTPS Keep Alive before timing out the session.	Integer	60

Table 8-2 Advanced Attributes

Attribute	Description	Range of Values	Default Value
Frontend Host	Set when the Host 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.	String	Null
Frontend HTTP Port	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.	Integer	0

Table 8-2 Advanced Attributes

Attribute	Description	Range of Values	Default Value
Frontend HTTPS Port	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.	Integer	0
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
Accept Context Path In Get Real Path	Beginning with the WebLogic Sever 8.1 release inclusion of the contextPath in the virtualPath to the context.getRealPath() will not be allowed as it breaks the case when the subdirectories have the same name as contextPath. In order to support applications which might have been developed according to the old behaviour BEA is providing a compatibility switch. This switch will be deprecated in future releases.	Boolean True = enabled False = not enabled	False

Table 8-2 Advanced Attributes

Attribute	Description	Range of Values	Default Value
HTTP Max Message Size	Maximum HTTP message size allowable in a message header. This attribute attempts to prevent a denial of service attack whereby a caller attempts to force the server to allocate more memory than is available, thereby keeping the server from responding quickly to other requests. This setting only applies to connections that are initiated using one of the default ports (ServerMBean setListenPort and setAdministrationPort or SSLMBean setListenPort). Connections on additional ports are tuned via the NetworkChannelMBean.	Minimum: 4096 Maximum: 2000000000	-1

Table 8-2 Advanced Attributes

Attribute	Description	Range of Values	Default Value
HTTP Message Timeout	Maximum number of seconds spent waiting for a complete HTTP message to be received. This attribute helps guard against denial of service attacks in which a caller indicates that they will be sending a message of a certain size which they never finish sending. This setting only applies to connections that are initiated using one of the default ports (ServerMBean setListenPort and setAdministrationPort or SSLMBean setListenPort). Connections on additional ports are tuned via the NetworkChannelMBean.	Minimum: 0 Maximum: 480	-1

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`.

Note: Setting up WebLogic Server to listen on port 80

- weblogic.system.listenPort=integer
- weblogic.system.enableSetUID=boolean
- weblogic.system.nonPrivUser=UNIXusername
- weblogic.system.enableSetGID=boolean

- weblogic.system.nonPrivGroup=UNIXgroupname

The latter four properties apply only to UNIX users.

On UNIX systems, binding a process to a port less than 1025 must be done from the account of a privileged user, usually root. Consequently, if you want WebLogic Server to listen on port 80, you must start WebLogic Server as a privileged user; yet it is generally considered undesirable from a security standpoint to allow long-running processes like WebLogic Server to run with more privileges than necessary. WebLogic needs root privileges only until the port is bound.

By setting the weblogic.system.enableSetUID property (and, if desired, the weblogic.system.enableSetGID property) to true, you enable an internal process by which WebLogic Server switches its UNIX user ID (UID) after it binds to port 80. The companion properties, weblogic.system.nonPrivUser and weblogic.system.nonPrivGroup, identifies a non-privileged UNIX user account (and optionally a groupname) under which WebLogic will run after startup.

You may choose to switch to the UNIX account "nobody," which is the least privileged user on most UNIX systems. If desired, you may create a UNIX user account expressly for running WebLogic Server. Make sure that files needed by WebLogic Server, such as log files and the WebLogic classes, are accessible by the non-privileged user. Once ownership of the WebLogic process has switched to the non-privileged user, WebLogic will have the same read, write, and execute permissions as the non-privileged user.

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 Connections/SSL tab.

Configuring the Listen Ports from the Administration Console

1. In the left pane, expand the Servers folder.
2. Click the server whose Listen Port you want to configure.
3. In the right pane, click the Configuration tab and the General subtab.
4. If you want to disable the non-SSL listen port so that the server listens only on the SSL listen port, remove the checkmark from the Listen Port Enabled box.
5. If you are using the non-SSL listen port and you want to modify the default port number, change the default number in the Listen Port box.
6. If you want to disable the SSL listen port so that the server listens only on the non-SSL listen port, remove the checkmark from the Enable SSL Listen Port box.

Note: You cannot disable both the non-SSL listen port and the SSL listen port. At least one port must be active.

7. If you want to modify the default SSL listen port number change the value in the SSL Listen Port box.
8. Click Apply.
9. Restart the server.

Web Applications

HTTP and Web Applications 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.

By default JSPs are compiled into the servers' temporary directory the location for which is (for a server: "myserver" and for a webapp: "mywebapp"):
<domain_dir>\myserver\.wlnotdelete\appname_mywebapp_4344862

For more information, see Assembling and Configuring Web Applications at `http://e-docs.bea.com/wls/docs81/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/docs81/cluster/index.html.

Designating a Default Web Application

Every server instance 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 8-10).

The examples domain that is shipped with Weblogic Server has a default Web Application already configured. The default Web Application in this domain is named `DefaultWebApp` and is located in the `applications` directory of the 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 set the context root in the weblogic.xml file to "/".

Virtual Directory Mapping

Using the virtual directory mapping feature, you can create one directory to serve static files such as images for multiple Web Applications. For example, you would create a mapping similar to the folowing:

```
<virtual-directory-mapping>
  <local-path>c:/usr/gifs</local-path>
  <url-pattern>/images/*</url-pattern>
</virtual-directory-mapping>
```

A request to `HTTP:// localhost:7001/mywebapp/images/test.gif` will cause your WebLogic Server implementation to look for the requested image at: `c:/usr/gifs/images/*`.

This directory must be located in the relative uri, such as "`/images/test.gif`".

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 instance 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 the 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 8-12.

When using multiple Virtual HOsts with diferent default web applications, you can not use single sign-on, as each web application will overwrite the JSESSIONID cookies set by the previous web application. This will occur even if the CookieName, CookiePath, and CookieDomain are identical in each of the default web applications.

Virtual host resolution for a given HTTP request is performed base on the incoming HOST header. If the HOST header is incorrect or absent, the Web application resolves to the default virtual host (default Web server).

Setting Up a Virtual Host

Use the Administration Console to define a virtual host.

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 the virtual hosts node. If any virtual hosts are defined, the node expands and displays a list of virtual hosts.
   c. Click 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 instance or cluster targeted by this virtual host.
   f. 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 8-15.)
   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 --> Servers tab. You will see a list of available servers.
   b. Select a server.
   c. Click Apply.

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/docs81/cluster/index.html`.
   a. Select the Targets tab.
   b. Select the Clusters tab. You will see a list of available clusters.
   c. Select a cluster.
   d. Click Apply.
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.
   f. Click Apply.

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/docs81/webapp/deployment.html#war-ear`.

The table below provides some sample URLs and the file that is served by WebLogic Server.

Table 8-3 Examples of How WebLogic Server Resolves URLs

URL	This file is served in response
`http://host:port/apples`	Welcome file* defined in the `apples` Web Application.
`http://host:port/apples`	Directory listing of the top level directory of the `apples` Web Application.
`http://host:port/oranges/naval`	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/docs81/webapp/components.html#configuring-servlets`.
`http://host:port/naval`	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/docs81/webapp/components.html#configuring-servlets`.

Table 8-3 Examples of How WebLogic Server Resolves URLs

URL	This file is served in response
`http://host:port/apples/pie.jsp`	`pie.jsp`, from the top-level directory of the `apples` Web Application.
`http://host:port`	Directory listing of the top level directory of the *default* Web Application
`http://host:port`	Welcome file* from the *default* Web Application.
`http://host:port/apples/myfile.html`	`myfile.html`, from the top level directory of the `apples` Web Application.
`http://host:port/myfile.html`	`myfile.html`, from the top level directory of the *default* Web Application.
`http://host:port/apples/images/red.gif`	`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.	Error 404 For more information, see Customizing HTTP Error Responses at `http://e-docs.bea.com/wls/docs81/webapp/components.html#error-page`.
`http://www.fruit.com/`	Welcome file* from the default Web Application for a virtual host with a host name of `www.fruit.com`.

Table 8-3 Examples of How WebLogic Server Resolves URLs

URL	This file is served in response
`http://www.fruit.com/`	Directory listing of the top level directory of the `default` Web Application for a virtual host with a host name of `www.fruit.com`.
`http://www.fruit.com/oranges/myfile.html`	`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/docs81/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.

For information on setting up HTTP logging for a server or a virtual host, refer to the following topics in the Administration Console online help:

- Enabling and Configuring an HTTP Log
- Specifying HTTP Log File Settings for a Virtual Host

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. You can configure the name of the access log file to include a

time and date stamp that indicates when the file was rotated. If you do not configure a time stamp, each rotated file name inlcudes a numeric portion that is incremented upon each rotation. Separate HTTP Access logs are kept for each Web Server you have defined.

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 "Creating Custom Field Identifiers" on page 8-19).

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 8-23.

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 8-17.

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 8-20.

– 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/docs81/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 8-16.

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 8-4 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 8-4 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.HttpServletRequest
`String getContextPath();`	javax.servlet.http.HttpServletRequest
`Cookie[] getCookies();`	javax.servlet.http.HttpServletRequest
`long getDateHeader(String name);`	javax.servlet.http.HttpServletRequest
`String getHeader(String name);`	javax.servlet.http.HttpServletRequest
`Enumeration getHeaderNames();`	javax.servlet.http.HttpServletRequest

Table 8-4 Getter Methods of HttpAccountingInfo

HttpAccountingInfo Methods	Where to find information on the methods
`Enumeration getHeaders(String name);`	javax.servlet.http.HttpServletRequest
`int getIntHeader(String name);`	javax.servlet.http.HttpServletRequest
`String getMethod();`	javax.servlet.http.HttpServletRequest
`String getPathInfo();`	javax.servlet.http.HttpServletRequest
`String getPathTranslated();`	javax.servlet.http.HttpServletRequest
`String getQueryString();`	javax.servlet.http.HttpServletRequest
`String getRemoteUser();`	javax.servlet.http.HttpServletRequest
`String getRequestURI();`	javax.servlet.http.HttpServletRequest
`String getRequestedSessionId();`	javax.servlet.http.HttpServletRequest
`String getServletPath();`	javax.servlet.http.HttpServletRequest
`Principal getUserPrincipal();`	javax.servlet.http.HttpServletRequest
`boolean isRequestedSessionIdFromCookie();`	javax.servlet.http.HttpServletRequest
`boolean isRequestedSessionIdFromURL();`	javax.servlet.http.HttpServletRequest
`boolean isRequestedSessionIdFromUrl();`	javax.servlet.http.HttpServletRequest
`boolean isRequestedSessionIdValid();`	javax.servlet.http.HttpServletRequest
`byte[] getURIAsBytes();`	Returns the URI of the HTTP request as byte array, for example: If `GET /index.html HTTP/1.0` is the first line of an HTTP Request, `/index.html` is returned as an array of bytes.

Table 8-4 Getter Methods of HttpAccountingInfo

HttpAccountingInfo Methods	Where to find information on the methods

`long getInvokeTime();`	Returns the starting time of `currentTimeMillis()`.
	To get the length of time taken by the servlet to send the response to the client, use the following code:
	`long milsec = System.currentTimeMillis() - metrics.getInvokeTime();`
	`Float sec = new Float(milsec / 1000.0);`

`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 Connections and Protocols tabs. 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.

  Note that the server must also support both the HTTP and T3 protocols in order to use HTTP tunneling.

`Tunneling Client 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 Client 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 4K 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>
```

`weblogic.http.nativeIOEnabled` can also be set as a context parameter in the FileServlet.

CHAPTER 9

Monitoring a WebLogic Server Domain

These sections describe WebLogic Server monitoring capabilities that help you manage and optimize application availability, performance, and security:

Facilities for Monitoring WebLogic Server

These sections describe WebLogic Server facilities for monitoring the health and performance of a WebLogic Server domain:

Administration Console

The WebLogic Server Administration Console provides visibility into a broad array of configuration and status information.

The Administration Console obtains information about domain resources from the domain's Administration Server. The Administration Server 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:

- Configuration MBeans, which control the domain's configuration
- Run-time MBeans, which provide a snapshot of information about domain resources, such as JVM memory usage. When a resource in the domain—for instance, a Web application—is instantiated, a run-time MBean instance is created which collects information about that resource.

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

For details on what data is available on specific console pages, see "Monitoring WebLogic Server Using the Administration Console" on page 9-4.

Server Self-Health Monitoring

WebLogic Server provides a self-health monitoring feature to improve the reliability and availability of server instances in a domain. Selected subsystems within each server instance monitor their health status based on criteria specific to the subsystem. If an individual subsystem determines that it can no longer operate in a consistent and reliable manner, it registers its health state as "failed" with the host server instance.

Each server instance checks the health state of all its registered subsystems to determine the overall viability of the server. If one or more critical subsystems have reached the "failed" state, the server instance marks its own health state as "failed" to indicate that it cannot reliably host an application.

When used in combination with Node Manager, server self-health monitoring enables you to automatically reboot servers that have failed. This improves the overall reliability of a domain, and requires no direct intervention from an administrator. See "Node Manager Capabilities" on page 4-3 for more information.

Obtaining Server Health Programmatically

You can check the self-reported health state of a server instance programmatically by calling the `getHealthState()` method on the `ServerRuntimeMBean`. Similarly, you can obtain the health state of a registered WebLogic Server subsystem by calling the `getHealthState()` method on its MBean. The following MBeans automatically register their health states with the host server:

- `JMSRuntimeMBean`
- `JMSServerRuntimeMBean`

- `JTARuntimeMBean`
- `TransactionResourceRuntimeMBean`

See the Javadocs for WebLogic Classes for more information on individual MBeans.

Messages and Log Files

WebLogic Server records information about events such as configuration changes, deployment of applications, and subsystem failures in log files. The information in log files is useful for detecting and troubleshooting problems, and monitoring performance and availability.

For detailed information about log files and the WebLogic Server logging subsystem, see "Logging" in *Administration Console Online Help*.

WebLogic Server outputs status and error messages to:

- **Standard Out**—By default, a WebLogic Server instance prints all messages of WARNING severity or higher to standard out—typically the command shell window in which you are running the server instance. You can control what messages a server instance writes to standard out using the Server→Logging tab.

  If you start a Managed Server with Node Manager, Node Manager redirects the server instance's standard out to a file. In this case, you can view the Managed Server's output using Domain→Server→Remote Start Output→View Server output.

- **Standard Error**—A WebLogic Server instance writes errors to standard error—typically the command shell window in which you are running the server instance.

  If you start a Managed Server with Node Manager, Node Manager redirects the server instance's standard error to a file. In this case, you can view the Managed Server's output using Domain→Server→Remote Start Output→View Server error output.

- **Node Manager Output**—If you start a Managed Server with Node Manager, you can access Node Manager output using Domain→Server→Remote Start Output→View Node Manager output.

- **Server Logs**—Each WebLogic Server instance writes all messages from its subsystems and applications to a log file on its host machine. You can configure logging behavior using the Server→Logging→Server tab. You can view a server instance's log file using the **View server log** link on any server tabs page.

- **Domain Log**—By default, each server instance in a domain forwards all messages from its subsystems and applications to the Administration Server for the domain. The Administration Server writes a subset of the messages to the Domain Log. You can control

whether or not a server instance sends its messages to the Administration Server, and configure filters that control which messages it sends using the Server→Logging→Domain tab. You can view the Domain Log using the **View domain log** link on any domain tab page.

An incorrect timestamp might be displayed for the domain log when multiple managed servers create the log files simultaneously. A difference in timestamp might also be observed when multiple components create logs even though the same managed server creates them.

- **Node Manager Logs**—Node Manager writes startup and status messages to a log file in the *WL_HOME*\common\nodemanager\NodeManagerLogs\NodeManagerInternal directory. Node Manager log files are named NodeManagerInternal_*timestamp*, where *timestamp* indicates the time at which Node Manager started.
- **HTTP Logs**—By default, each server instance maintains a log of HTTP requests. Disable HTTP logging, or configure logging behavior using the Server→Logging→HTTP tab.
- **JTA Logs**—Configure a server instance to maintain a JTA transaction log using the Server→Logging→JTA tab.
- **JDBC Logs**—Configure a server instance to maintain a JDBC log using the Server→Logging→JDBC tab.

Monitoring WebLogic Server Using the Administration Console

The left pane of the Administration Console is a tree control, with a node for key entities you have configured. The following sections list the attributes displays on monitoring pages in each node:

- "Domain Monitoring Pages" on page 9-5
- "Server Monitoring Pages" on page 9-5
- "Clusters Monitoring Pages" on page 9-11
- "Machine Monitoring Pages" on page 9-11
- "Deployments Monitoring Pages" on page 9-12
- "Services Monitoring Pages" on page 9-18

Domain Monitoring Pages

You can access one WebLogic domain at a time using the Administration Console. The Domain->Monitoring tab provides access to key configuration attributes and the current state for the servers and clusters in the current domain. The following table lists the monitoring pages available for a domain, and the attributes displayed on each page.

Table 9-1 Domain Monitoring Pages

Console Page	Attributes Displayed
Domain->Monitoring->Server	• Name • Machine • Listen Address • Listen Port • State • SSL Listen Port
Domain->Monitoring->Cluster	• Name • Cluster Status • Cluster Address • Multicast Address • Multicast Send Delay • Configure Server Count • Good Server Count • Bad Server Count

Other Domain Monitoring Links

Each domain-level monitoring page has links to display:

- Domain Log—The Domain Log contains messages forwarded by all server instances in the domain.

Server Monitoring Pages

When expanded, the Servers node lists each server instance in the current domain. To monitor key run-time attributes for a server instance, click on its name, and choose one of Monitoring tabs. The monitoring pages available depend on the application objects deployed to the server

instance. The following table lists the monitoring pages available for a server instance, and the attributes displayed on each page.

Table 9-2 Server Monitoring Pages

Console Page	Attributes Displayed
Server->Monitoring->General	• State Activation Time • WebLogic Version • JDK Vendor • JDK Version • Operating System • OS Version
Server->Monitoring->General Monitor All Active Queues...	• Name • Threads • Idle Threads • Oldest Pending Request • Queue Length • Throughput
Server->Monitoring->General Monitor all Connections...	• Remote Address • Remote JVMID • Protocol • Connect Time • Bytes Sent • Messages Received • Messages Sent
Server->Monitoring->General Monitor all Active Sockets...	• Remote Address • Protocol
Server->Monitoring->Performance	• Idle Threads • Oldest Pending Request • Throughput • Queue Length • Memory Usage

Console Page	Attributes Displayed
Server->Monitoring-> Security	• Invalid Login Attempts Total Count • User Lockout Total Count • Login Attempts while Locked Total Count • Invalid Login Users High Count • Locked Users Current Count • Unlocked Users Total Count
Server->JRockit	• Used Heap • Used Physical Memory • Total Nursery Size • Max Heap Size • Gc Algorithm • Total Garbage Collection Count • Last GcEnd • Last GcStart • Total GcTime • GCHandles Compaction • Concurrent • Generational • Incremental • Parallel • Number Of Processors • All Processors Average Load • JVM Processor Load • Total Number Of Threads • Number Of Daemon Threads • Uptime
Server->Monitoring->JMS	• Current Connections • Connections High • Total Connections • Current JMS Servers • Servers High • Servers Total

Console Page	Attributes Displayed
Server->Monitoring->JMS Monitor all Active JMS Connections...	• Client ID • Active sessions • Most sessions • Total sessions
Server->Monitoring->JMS Monitor all Active JMS Servers...	• Server • Connections • Active JMS Destinations • Session Pools • Destinations • Messages • Bytes Current • Bytes Pending • Bytes Received • Bytes Threshold Time • Destinations High • Destinations Total • Messages High • Messages Pending • Messages Received • Messages Threshold Time • Name • Session Pools High • Session Pools Total

Console Page	Attributes Displayed
Server->Monitoring->JMS Monitor all Pooled JMS Connections...	• Sessions Available • Sessions Reserved • Sessions Waiting • Current Pool Capacity • Maximum Pool Capacity • Open Connections • Average Sessions Reserved • Creation Delay Time • Highest Sessions Available • Highest Sessions Not Available • Highest Sessions Reserved • Highest Sessions Waiting • Highest Wait Seconds • Leaked Sessions • Name • Refresh Failures • Sessions Not Available • Total Sessions Allocated • Total Sessions Destroyed
Server->Monitoring->JTA	• Total Transactions • Total Committed • Total Rolled Back • Timeout Rollbacks • Resource Rollbacks • Application Rollbacks • System Rollbacks • Total Heuristics • Total Transactions Abandoned • Average Transactions Count • Average Commit Time

Console Page	Attributes Displayed
Server->Monitoring->JTA Monitor all Transactions by Name	• Name • Total Transactions • Total Committed • Total Rolled Back • Timeout Rollbacks • Resource Rollbacks • Application Rollbacks • System Rollbacks • Total Heuristics • Total Transactions Abandoned • Average Transactions Count • Average Commit Time
Server->Monitoring->JTA Monitor all Transactions by Resource	• Name • Transactions • Commits • Rollbacks • Heuristics • Heuristic Commits • Heuristic Rollbacks • Mixed Heuristics • Heuristic Hazards
Server->Monitoring->JTA Monitor all In-Flight Transactions	• Transaction Id • Name • Status • Seconds Active • Servers • Resources

Other Server Monitoring Links

Each top level tab page for a server instance—Performance, Security, JMS, JTA— has links to display:

- Server Log—The Server Log contains all messages generated by its subsystems and applications.
- JNDI Tree—The JNDI tree shows the objects deployed to the current server instance.

Clusters Monitoring Pages

The following table lists the monitoring pages available for a cluster, and the attributes displayed on each page

Table 9-3 Cluster Monitoring Pages

Console Page	Attributes Displayed
Domain->Cluster->Monitoring	• Number of Servers configured for this cluster • Number of Servers currently participating in this cluster • Name • State • Servers • Resend Requests • Fragments Received • Lost Multicast Messages

Machine Monitoring Pages

The following table lists the monitoring pages available for a machine, and the attributes displayed on each page.

Table 9-4 Machine Monitoring Pages

Console Page	Attributes Displayed
Machine-->Monitoring Node Manager Status	• State • `bea.home` • `weblogic.nodemanager.javaHome` • `weblogic.nodemanager.listenAddress` • `weblogic.nodemanager.listenPort` • `CLASSPATH`
Machine-->Monitoring Node Manager Log	See "Node Manager Logs" on page 4.

Deployments Monitoring Pages

The following table lists the monitoring pages available in the Deployments Node, and the attributes displayed on each page.

Connector Deployments

The following table lists the monitoring pages available for deployed Connectors and the attributes displayed on each page.

Table 9-5 Connector Deployments

Console Page	Attributes Displayed
Connector Modules	• Connections • Active Connections High Count • Free Connections High Count • Average Active Usage • Connections Created Total Count • Connections Matched Total Count • Connections Destroyed Total Count • Connections Rejected Total Count • Connection Idle Profile Count • Connection Idle Profiles • Connection Leak Profile Count • Connection Leak Profiles • Number Detected Idle • Number Detected Leaks • Recycled Total

EJB Deployments

The following table lists the monitoring pages available for deployed EJBs and the attributes displayed on each page.

Table 9-6 EJB Deployments

Console Page	Attributes Displayed
EJB Modules->Message Driven EJBs	• Pool Miss Ratio • Destroyed Bean Ratio • JMS Connection Alive • Transaction Rollback Ratio • Transaction Timeout Ratio • Access Total Count • Beans In Use Current Count • Destroyed Total Count • EJB Name • Miss Total Count • Pool Timeout Total Count • Pool Waiter Total Count • Pooled Beans Current Count • Server • Transactions Committed Total Count • Transactions Rolled Back Total Count • Transactions Timed out Total Count • Waiter Current Count
EJB Modules->Stateless EJBs	• Pool Timeout Total Count • Pool Waiter Total Count • Pooled Beans Current Count • Server • Transactions Committed Total Count • Transactions Rolled Back Total Count • Transactions Timed out Total Count • Waiter Current Count

Console Page	Attributes Displayed
EJB Modules->Entity EJBs	• Cache Miss Ratio • Lock Manager Waiter Ratio • Lock Manager Timeout Ratio • Pool Miss Ratio • Destroyed Bean Ratio • Pool Timeout Ratio • JMS Connection Alive • Transaction Rollback Ratio • Transaction Timeout Ratio • Access Total Count • Activation Count • Beans In Use Current Count • Cache Access Count • Cache Miss Count • Cached Beans Current Count • Destroyed Total Count • EJB Name • Lock Mgr Access Count • Lock Mgr Entries Current Count • Lock Mgr Timeout Total Count • Lock Mgr Waiter Total Count • Miss Total Count • Passivation Count • Pool Timeout Total Count • Pool Waiter Total Count • Pooled Beans Current Count • Server • Transactions Committed Total Count • Transactions Rolled Back Total Count • Transactions Timed out Total Count • Waiter Current Count

Console Page	Attributes Displayed
EJB Modules->Stateful EJBs	• Cache Miss Ratio • Lock Manager Waiter Ratio • Lock Manager Timeout Ratio • Transaction Rollback Ratio • Transaction Timeout Ratio • Activation Count • Beans In Use Current Count • Cache Access Count • Cache Miss Count • Cached Beans Current Count • Destroyed Total Count • EJB Name • Lock Mgr Access Count • Lock Mgr Entries Current Count • Lock Mgr Timeout Total Count • Lock Mgr Waiter Total Count • Passivation Count • Server • Transactions Committed Total Count • Transactions Rolled Back Total Count • Transactions Timed out Total Count

Web Services Deployments

The following table lists the monitoring pages available for deployed Web Services and the attributes displayed on each page.

Table 9-7 Web Services Deployments

Console Page	Attributes Displayed
Web Service	• Server • Context Root • Servlets • Sessions • Sessions High • Total Sessions • Machine • Source Info
Sessions	•
Servlets	• Servlet Name • Servlet Path • Machine • Pool Max Capacity • Reload Total Count • Server • Invocation Total Count • Execution Time Average • Execution Time High • Execution Time Low • Execution Time Total

Web Application Deployments

The following table lists the monitoring pages available for deployed Web Applications and the attributes displayed on each page.

Table 9-8 Web Application Deployments

Console Page	Attributes Displayed
Web Applications	• Server • Context Root • Servlets • Sessions • Sessions High • Total Sessions • Machine • Source Info
Servlets	• Servlet Name • URLPatterns • Server • Invocation Total Count • Execution Time Average • Execution Time High • Execution Time Low • Execution Time Total • Machine • Name • Pool Max Capacity • Reload Total Count • Servlet Path
Sessions	•

Services Monitoring Pages

The following table lists the monitoring pages available in the Services node, and the attributes displayed on each page.

Table 9-9 JDBC Monitoring Pages

Console Page	Attributes Displayed
JDBC->Connection Pools	• Server • State • Connections • Waiters • Num Unavailable • Active Connections Average • Connections Delay Time • Connections High • Connections Total • Curr Capacity • Driver Version • Failures to Reconnect Count • Highest Num Unavailable • Max Capacity • Wait Seconds High • Waiters High

Table 9-10 JMS Deployments

Console Page	Attributes Displayed
JMS Connection Factory	• Name • JNDIName • Client Id • Default Priority • Default Time To Live • Default Redelivery Delay
JMS Server Monitor Active JMS Servers	• Name • Server • Connections • Active JMS Destination • Session Pools • Destinations • Messages • Bytes Current • Bytes Pending • Bytes Received • Bytes Threshold Time • Destinations High • Destinations Total • Messages High • Messages Pending • Messages Received • Messages Threshold Time • Messages Maximum • Session Pools High • Session Pools Total

Console Page	Attributes Displayed
JMS Server Monitor Active JMS Destinations	• Name • Consumers • Consumers High • Consumers Total • Messages • Messages High • Messages Pending • Messages Received • Messages Threshold Time • Messages Maximum • Bytes • Bytes Pending • Bytes Received • Bytes Threshold Time
JMS Server Monitor JMS Session Pools	• Name • Store • Temporary Template • Bytes Maximum • Messages Maximum
Monitor Durable Subscribers	• Client ID • Subscription Name • No Local • Active • Selector • Messages Pending Count • Messages Current Count • Bytes Pending Count — • Bytes Current Count

Console Page	Attributes Displayed
Active JMS Producers	• Bytes Pending • Bytes Sent • Messages Pending • Messages Sent
Active JMS Consumers	• Name • Active • Selected • Durable • Messages Pending • Messages Received • Bytes Pending • Bytes Received

CHAPTER 10

Recovering Failed Servers

A variety of events can lead to the failure of a server instance. Often one failure condition leads to another. Loss of power, hardware malfunction, operating system crashes, network partitions, and unexpected application behavior can all contribute to the failure of a server instance.

Depending on availability requirements, you may implement a clustered architecture to minimize the impact of failure events. (For information about failover in a WebLogic Server cluster, see "Failover and Replication in a Cluster" in *Using WebLogic Server Clusters*.) However, even in a clustered environment, server instances may fail periodically, and it is important to be prepared for the recovery process.

These following sections provide information about and procedures for recovering failed server instances:

WebLogic Server Failure Recovery Features

This section describes WebLogic features that support recovery from failure.

Automatic Restart for Managed Servers

WebLogic Server self-health monitoring improves the reliability and availability of server instances in a domain. Selected subsystems within each WebLogic Server instance monitor their

health status based on criteria specific to the subsystem. (For example, the JMS subsystem monitors the condition of the JMS thread pool while the core server subsystem monitors default and user-defined execute queue statistics.) If an individual subsystem determines that it can no longer operate in a consistent and reliable manner, it registers its health state as "failed" with the host server.

Each WebLogic Server instance, in turn, checks the health state of its registered subsystems to determine its overall viability. If one or more of its critical subsystems have reached the `FAILED` state, the server instance marks its own health state `FAILED` to indicate that it cannot reliably host an application.

When used in combination with Node Manager, server self-health monitoring enables you to automatically reboot servers that have failed. This improves the overall reliability of a domain, and requires no direct intervention from an administrator. See "Node Manager Capabilities" on page 4-3.

To configure Node Manager and automatic restart behaviors, see "Configuring Node Manager" on page 5-1.

Managed Server Independence Mode 

When a Managed Server starts, it tries to contact the Administration Server to retrieve its configuration information. If a Managed Server cannot connect to the Administration Server during startup, it can retrieve its configuration by reading configuration and security files directly. A Managed Server that starts in this way is running in *Managed Server Independence (MSI)* mode. By default, MSI mode is enabled. For information about disabling MSI mode, see "Disabling Managed Server Independence" in *Administration Console Online Help*.

In Managed Server Independence mode, a Managed Server looks in its root directory for the following files:

- `msi-config.xml`—a replica of the domain's `config.xml`. (Even if the domain's configuration file is named something other than `config.xml`, a Managed Server in MSI mode always looks for a configuration file named `msi-config.xml`.)
- `SerializedSystemIni.dat`
- `boot.properties`—an optional file that contains an encrypted version of your username and password. For more information, see "Boot Identity Files" in *Administration Console Online Help*.

MSI Mode and the Managed Servers Root Directory

By default, a server instance assumes that its root directory is the directory from which it was started. For more information about a server's root directory, refer to "A Server's Root Directory" on page 2-12.

If you enable replication of configuration data, as described in "Backing Up Security Data" on page 10-8, and if you have started the Managed Server at least once while the Administration Server was running, `msi-config.xml` and `SerializedSystemIni.dat` will already be in the server's root directory. The `boot.properties` file is not replicated. If it is not already in the Managed Server's root directory, you must create one. For more information, see "Boot IdentityFiles" in *Administration Console Online Help*.

If `msi-config.xml` and `SerializedSystemIni.dat` are not in the root directory, you can either:

- Copy `config.xml` and `SerializedSystemIni.dat` from the Administration Server's root directory (or from a backup) to the Managed Server's root directory. Then, rename the configuration file to `msi-config.xml`, or
- Use the `-Dweblogic.RootDirectory=`*`path`* startup option to specify a directory that already contains these files.

MSI Mode and the Domain Log File

Each WebLogic Server instance writes log messages to its local log file and a domain-wide log file. The domain log file provides a central location from which to view messages from all servers in a domain.

Usually, a Managed Server forwards messages to the Administration Server, and the Administration Server writes the messages to the domain log file. However, when a Managed Server runs in MSI mode, it assumes the role of writing to the domain log file.

By default, the pathnames for local log files and domain log files are relative to the Manged Server's root directory. With these default settings, if a Managed Server is located in its own root directory (and it does not share its root directory with the Administration Server), when it runs in MSI mode the Managed Server will create its own domain log file in its root directory.

If a Managed Server shares its root directory with the Administration Server, or if you specified an absolute pathname to the domain log, the Managed Server in MSI mode will write to the domain log file that the Administration Server created.

Note: The Managed Server must have permission to write to the existing file. If you run the Administration Server and Managed Servers under different operating system accounts,

you must modify the file permissions of the domain log file so that both user accounts have write permission.

MSI Mode and the Security Realm

A Managed Server must have access to a security realm to complete its startup process.

If you use the security realm that WebLogic Server installs, then the Administration Server maintains an LDAP server to store the domain's security data. All Managed Servers replicate this LDAP server. If the Administration Server fails, Managed Servers running in MSI mode use the replicated LDAP server for security services.

If you use a third party security provider, then the Managed Server must be able to access the security data before it can complete its startup process.

MSI Mode and SSL

If you set up SSL for your servers, each server requires its own set of certificate files, key files, and other SSL-related files. Managed Servers do not retrieve SSL-related files from the Administration Server (though the domain's configuration file does store the pathnames to those files for each server). Starting in MSI Mode does not require you to copy or move the SSL-related files unless they are located on a machine that is inaccessible.

MSI Mode and Deployment

A Managed Server that starts in MSI mode deploys its applications from its staging directory: *serverroot*/stage/*appName*.

MSI Mode and Managed Server Configuration Changes

If you start a Managed Server in MSI mode, you cannot change its configuration until it restores communication with the Administration Server.

MSI Mode and Node Manager

You cannot use Node Manager to start a server instance in MSI mode, only to restart it. For a routine startup, Node Manager requires access to the Administration Server. If the Administration Server is unavailable, you must log onto Managed Server's host machine to start the Managed Server.

MSI Mode and Configuration File Replication

Managed Server Independence mode includes an option that copies the required configuration files into the Managed Server's root directory every 5 minutes. This option does not replicate a boot identity file. (For more information about boot identity files, see "Boot Identity Files" in *Administration Console Online Help*.)

By default, a Managed Server does not replicate these files. Depending on your backup schemes and the frequency with which you update your domain's configuration, this option might not be worth the performance cost of copying potentially large files across a network.

To enable a Managed Server to replicate the domain's configuration files, see "Replicating a Domain's Configuration Files for Managed Server Independence" in *Administration Console Online Help*.

MSI Mode and Restored Communication with an Administration Server

When the Administration Server starts, it can detect the presence of running Managed Servers (if `-Dweblogic.management.discover=true`, which is the default setting for this property).

Upon startup, the Administration Server looks at a persisted copy of the file `running-managed-servers.xml` and notifies all the Managed Servers listed in the file of its presence.

Managed Servers that were started in Managed Server Independence Mode while the Administration Server was unavailable will not appear in `running-managed-servers.xml`. To re-establish a connection between the Administration Server and such Managed Servers, use the `weblogic.Admin DISCOVERMANAGEDSERVER` command. See "DISCOVERMANAGEDSERVER" in *WebLogic Server Command Reference*.

When an Administration Server starts up and contacts a Managed Server running in MSI mode, the Managed Server deactivates MSI mode and registers itself to the Administration Server for future configuration change notifications.

Backing Up Configuration and Security Data

Recovery from the failure of a server instance requires access to the domain's configuration and security data. This section describes file backups that WebLogic Server performs automatically, and recommended backup procedures that an administrator should perform.

Backing up config.xml

By default, an Administration Server stores a domain's configuration data in a file called *domain_name*\config.xml, where *domain_name* is the root directory of the domain.

Back up config.xml to a secure location in case a failure of the Administration Server renders the original copy unavailable. If an Administration Server fails, you can copy the backup version to a different machine and restart the Administration Server on the new machine.

WebLogic Server Archives Previous Versions of config.xml

By default, the Administration Server archives up to 5 previous versions of config.xml in the *domain-name*/configArchive directory.

When you save a change to a domain's configuration, the Administration Server saves the previous configuration in *domain-name*\configArchive\config.xml#*n*. Each time the Administration Server saves a file in the configArchive directory, it increments the value of the #*n* suffix, up to a configurable number of copies—5 by default. Thereafter, each time you change the domain configuration:

- The archived files are rotated so that the newest file has a suffix with the highest number,
- The previous archived files are renamed with a lower number, and
- The oldest file is deleted.

Example of Archived config.xml Naming and Rotation

In the MedRec domain, the current configuration file used by the MedRecServer is *WL_HOME*\samples\domains\medrec\config.xml. If you add a server instance using the Administration Console, when you click the Create button, MedRecServer saves the old config.xml file as *WL_HOME*\samples\domains\medrec\configArchive\config.xml#2.

The new file, *WL_HOME*\samples\domains\medrec\config.xml, represents the MedRec domain with the new server instance. The previous file, *WL_HOME*\samples\domains\medrec\configArchive\config.xml#2, contains the MedRec domain configuration as it was prior to creation of the new server instance.

The next time you change the configuration, MedRecServer saves the current config.xml file as config.xml#3. After four changes to the domain, the configArchive directory contains four files: config.xml#2, config.xml#3, config.xml#4, config.xml#5. The next time you change the configuration, MedRecServer saves the old config.xml as config.xml#5. The

previous config.xml#5 is renamed as config.xml#4, and so on. The old config.xml#2 is deleted.

Configuring the Number of Archived config.xml Versions

To configure how many previous versions of the domain configuration are archived:

1. In the left pane of the Administration Console, click on the name of the domain.
2. In the right pane, click the Configuration->General tab.
3. In the Advanced Options bar, click Show.
4. In the Archive Configuration Count box, enter the number of versions to save.
5. Click Apply.

WebLogic Server Archives config.xml during Server Startup

In addition to the files in *domain-name*\configArchive, the Administration Server creates two other files that back up the domain's configuration at key points during the startup process:

- *domain-name**config-file*.xml.original—The configuration file just before the Administration Server parses it and adds subsystem data.
- *domain-name**config-file*.xml.booted—The configuration file just after the Administration Server successfully boots. If the config.xml becomes corrupted, you can boot the Administration Server with this file.

Example of Archives of config.xml During Startup

If your domain configuration is stored in config.xml, when you start the domain's Administration Server, the Administration Server:

1. Copies config.xml to config.xml.original.
2. Parses config.xml. Depending on the domain configuration, some WebLogic subsystems add configuration information to config.xml. For example, the Security service adds MBeans and encrypted data for SSL communication.
3. Copies the parsed and modified config.xml to MyConfig.xml.booted.

The Administration Server uses the parsed and modified config.xml. When you update the domain's configuration, it copies the old config.xml to *domain-name*\configArchive\MyConfig.xml#2.

Backing Up Security Data

The WebLogic Security service stores its configuration data `config.xml` file, and also in an LDAP repository and other files.

Backing Up the WebLogic LDAP Repository

The default Authentication, Authorization, Role Mapper, and Credential Mapper providers that are installed with WebLogic Server store their data in an LDAP server. Each WebLogic Server contains an embedded LDAP server. The Administration Server contains the master LDAP server which is replicated on all Managed Servers. If any of your security realms use these installed providers, you should maintain an up-to-date backup of the following directory tree:

`domain_name``\`*`adminServer`*`\ldap`

where *`domain_name`* is the domain's root directory and *`adminServer`* is the directory in which the Administration Server stores runtime and security data.

Each WebLogic Serve has an LDAP directory, but you only need to back up the LDAP data on the Administration Server—the master LDAP server replicates the LDAP data from each Managed Server when updates to security data are made. WebLogic security providers cannot modify security data while the domain's Administration Server is unavailable. The LDAP repositories on Managed Servers are replicas and cannot be modified.

The `ldap/ldapfiles` subdirectory contains the data files for the LDAP server. The files in this directory contain user, group, group membership, policies, and role information. Other subdirectories under the `ldap` directory contain LDAP server message logs and data about replicated LDAP servers.

Do not update the configuration of a security provider while a backup of LDAP data is in progress. If a change is made—for instance, if an administrator adds a user—while you are backing up the `ldap` directory tree, the backups in the `ldapfiles` subdirectory could become inconsistent. If this does occur, consistent, but potentially out-of-date, LDAP backups are available, as described in "WebLogic Server Backs Up LDAP Files" on page 10-8.

WebLogic Server Backs Up LDAP Files

Once a day, a server suspends write operations and creates its own backup of the LDAP data. It archives this backup in a `ZIP` file below the `ldap\backup` directory and then resumes write operations. This backup is guaranteed to be consistent, but it might not contain the latest security data.

For information about configuring the LDAP backup, see "Configuring Backups for the Embedded LDAP Server" in *Administration Console Online Hel*p.

Backing Up SerializedSystemIni.dat and Security Certificates

All servers create a file named `SerializedSystemIni.dat` and locate it in the server's root directory. This file contains encrypted security data that must be present to boot the server. You must back up this file.

If you configured a server to use SSL, you must also back up the security certificates and keys. The location of these files is user-configurable.

Restarting Failed Server Instances

The nature of your applications and user demand determine the steps you take to restore application service. In particular, these factors influence the recovery process:

- Was the failed server instance an Administration Server or a Managed Server?
- Can you restart the failed server instance on same machine upon which it was running when it failed?
- What are the network conditions when you restart the server instance? Can the service instance you are restarting establish communications with its Administration Server?
- Was the server instance that failed the active host for a migratable service in a WebLogic Server cluster?
- Were any changes made to the domain configuration made while the failed server instance was down?
- Was the domain configuration corrupted?

Restarting an Administration Server

The following sections describe how to start an Administration Server after a failure.

Restarting an Administration Server When Managed Servers Not Running

If no Managed Servers in the domain are running when you restart a failed Administration Server, no special steps are required. Start the Administration Server as you normally do. See "Starting and Stopping Servers" in *Administration Console Online Help*.

Restarting an Administration Server When Managed Servers Are Running

If the Administration Server shuts 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 an Administration Server on the Same Machine

If you restart the WebLogic Administration Server while Managed Servers continue to run, by default the Administration Server can discover the presence of the running Managed Servers.

Note: Make sure that the startup command or startup script does not include `-Dweblogic.management.discover=false`, which disables an Administration Server from discovering its running Managed Servers. For more information about `-Dweblogic.management.discover`, see "Server Communication" in *weblogic.Server Command-Line Reference*.

The root directory for the domain contains a file `running-managed-servers.xml` which contains a list of the Managed Servers in the domain and whether they are running or not. When the Administration Server restarts, it checks this file to determine which Managed Servers were under its control before it stopped running.

When a Managed Server is gracefully or forcefully shut down, its status in `running-managed-servers.xml` is updated to "not-running". When an Administration Server restarts, it does not try to discover Managed Servers with the "not-running" status. A Managed Servers that stops running because a system crash, or that was stopped by killing the JVM or the command prompt (shell) in which it was running, will still have the status "running' in `running-managed-servers.xml`. The Administration Server will attempt to discover them, and will throw an exception when it determines that the Managed Server is no longer running.

Restarting the Administration Server does not cause Managed Servers to update the configuration of static attributes. *Static attributes* are those that a server refers to only during its startup process. Servers instances must be restarted to take account of changes to static configuration attributes. Discovery of the Managed Servers only enables the Administration Server to monitor the Managed Servers or make runtime changes in attributes that can be configured while a server is running (dynamic attributes).

Restarting an 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).
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 data available to the new administration machine by copying them from backups or by using a shared disk. For more information, refer to "Backing Up Configuration and Security Data" on page 10-5.
4. Restart the Administration Server on the new machine.

   Make sure that the startup command or startup script does not include `-Dweblogic.management.discover=false`, which disables an Administration Server from discovering its running Managed Servers. For more information about `-Dweblogic.management.discover`, see "Server Communication" in *weblogic.Server Command-Line Reference*.

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.

Restarting Managed Servers

The following sections describe how to start Managed Servers after failure. For recovery considerations related to transactions and JMS, see "Additional Failure Topics" on page 10-12.

Starting a Managed Server When the Administration Server Is Accessible

If the Administration Server is reachable by Managed Server that failed, you can:

- Restart it manually or automatically using Node Manager—You must configure Node Manager and the Managed Server to support this behavior. For details, see "Configure Monitoring, Shutdown, and Restart for Managed Servers" on page 5-7.
- Start it manually with a command or script—For instructions, see "Starting and Stopping Servers" in *Administration Console Online Help*.

Starting a Managed Server When the Administration Server Is Not Accessible

If a Managed Server cannot connect to the Administration Server during startup, it can retrieve its configuration by reading locally cached configuration data. A Managed Server that starts in this way is running in Managed Server Independence (MSI) mode. For a description of MSI mode, and the files that a Managed Server must access to start up in MSI mode, see "Managed Server Independence Mode" on page 10-2.

Note: If the Managed Server that failed was a clustered Managed Server that was the active server for a migratable service at the time of failure, perform the steps described in "Migrating When the Currently Active Host is Unavailable" in *Using WebLogic Server Clusters*. Do not start the Managed Server in MSI mode.

To start up a Managed Server in MSI mode:

1. Ensure that the following files are available in the Managed Server's root directory:
   - `msi-config.xml`.
   - `SerializedSystemIni.dat`
   - `boot.properties`

   If these files are not in the Managed Server's root directory:

   a. Copy the config.xml and `SerializedSystemIni.dat` file from the Administration Server's root directory (or from a backup) to the Managed Server's root directory.

   b. Rename the configuration file to `msi-config.xml`. When you start the server, it will use the copied configuration files.

   Note: Alternatively, you can use the `-Dweblogic.RootDirectory=`*`path`* startup option to specify a root directory that already contains these files.

2. Start the Managed Server at the command line or using a script.

   The Managed Server will run in MSI mode until it is contacted by its Administration Server. For information about restarting the Administration Server in this scenario, see "Restarting an Administration Server When Managed Servers Are Running" on page 10-10.

Additional Failure Topics

For information related to recovering JMS data from a failed server instance, see "Configuring JMS Migratable Targets" in *Programming WebLogic JMS*.

For information about transaction recovery after failure, see "Moving a Server to Another Machine" and "Transaction Recovery After a Server Fails" in *Administration Console Online Help*.

CHAPTER 11

Configuring Network Resources

Configurable WebLogic Server resources, including network channels and domain-wide administration ports, help you effectively utilize the network features of the machines that host your applications and manage quality of service.

The following sections describe configurable WebLogic Server network resources, examples of their use, and the configuration process:

Overview of Network Configuration

For many development environments, configuring WebLogic Server network resources is simply a matter of identifying a Managed Server's listen address and listen port. However, in most production environments, administrators must balance finite network resources against the demands placed upon the network. The task of keeping applications available and responsive can be complicated by specific application requirements, security considerations, and maintenance tasks, both planned and unplanned.

WebLogic Server allows you to control the network traffic associated with your applications in a variety of ways, and configure your environment to meet the varied requirements of your applications and end users. You can:

- Designate the Network Interface Cards (NICs) and ports used by Managed Servers for different types of network traffic.
- Support multiple protocols and security requirements.
- Specify connection and message timeout periods.
- Impose message size limits.

You specify these and other connection characteristics by defining a network channel—the primary configurable WebLogic Server resource for managing network connections. You configure a network channel with the Servers-->Protocols-->Channels tab of the Administration Console or by using `NetworkAccessPointMBean`.

New Network Configuration Features in WebLogic Server

In this version of WebLogic Server, the functionality of network channels is enhanced to simplify the configuration process. Network channels now encompass the features that, in WebLogic Server 7.x, required both network channels and network access points. In this version of WebLogic Server, network access points are deprecated. The use of `NetworkChannelMbean` is deprecated in favor of `NetworkAccessPointMBean`.

Understanding Network Channels

The sections that follow describe network channels and the standard channels that WebLogic Server pre-configures, and discusses common applications for channels.

What Is a Channel?

A network channel is a configurable resource that defines the attributes of a network connection to WebLogic Server. For instance, a network channel can define:

- The protocol the connection supports.
- The listen address.
- The listen ports for secure and non-secure communication.
- Connection properties such as the login timeout value and maximum message sizes.
- Whether or not the connection supports tunneling.

- Whether the connection can be used to communicate with other WebLogic Server instances in the domain, or used only for communication with clients.

Rules for Configuring Channels

Follow these guidelines when configuring a channel.

- You can assign a particular channel to only one server instance.
- You can assign multiple channels to a server instance.
- Each channel assigned to a particular server instance must have a unique combination of listen address, listen port, and protocol.
- If you assign non-SSL and SSL channels to the same server instance, make sure that they do not use the same port number.

Custom Channels Can Inherit Default Channel Attributes

If you do not assign a channel to a server instance, it uses WebLogic Server's default channel, which is automatically configured by WebLogic Server, based on the attributes in `ServerMBean` or `SSLMBean`. The default channel is described in "The Default Network Channel" on page 11-5.

`ServerMBean` and `SSLMBean` represent a server instance and its SSL configuration. When you configure a server instance's Listen Address, Listen Port, and SSL Listen port, using the Server-->Configuration-->General tab, those values are stored in the `ServerMBean` and `SSLMBean` for the server instance.

If you do not specify a particular connection attribute in a custom channel definition, the channel inherits the value specified for the attribute in `ServerMBean`. For example, if you create a channel, and do not define its Listen Address, the channel uses the Listen Address defined in `ServerMBean`. Similarly, if a Managed Server cannot bind to the Listen Address or Listen Port configured in a channel, the Managed Server uses the defaults from `ServerMBean` or `SSLMBean`.

Why Use Network Channels?

You can use network channels to manage quality of service, meet varying connection requirements, and improve utilization of your systems and network resources. For example, network channels allow you to:

- **Segregate different types of network traffic**—You can configure whether or not a channel supports outgoing connections. By assigning two channels to a server instance—one that supports outgoing connections and one that does not—you can independently

configure network traffic for client connections and server connections, and physically separate client and server network traffic onto different listen addresses or listen ports.

You can also segregate instance administration and application traffic by configuring a domain-wide administration port or administration channel. For more information, see "Administration Port and Administrative Channel" on page 11-5.

- **Support varied application or user requirements on the same Managed Server—**You can configure multiple channels on a Managed Server to support different protocols, or to tailor properties for secure vs. non-secure traffic.

If you use a network channel with a server instance on a multi-homed machine, you must enter a valid Listen Address either in `ServerMBean` or in the channel. If the channel and `ServerMBean` Listen Address are blank or specify the localhost address (IP address 0.0.0.0 or 127.*.*.*), the server binds the network channel listen port and SSL listen ports to all available IP addresses on the multi-homed machine. See "The Default Network Channel" on page 11-5 for information on setting the listen address in `ServerMBean`.

Handling Channel Failures

When initiating a connection to a remote server, and multiple channels with the same required destination, protocol and quality of service exist, WebLogic Server will try each in turn until it successfully establishes a connection or runs out of channels to try.

Upgrading Quality of Service Levels for RMI

For RMI lookups only, WebLogic Server may upgrade the service level of an outgoing connection. For example, if a T3 connection is required to perform an RMI lookup, but an existing channel supports only T3S, the lookup is performed using the T3S channel.

This upgrade behavior does not apply to server requests that use URLs, since URLs embed the protocol itself. For example, the server cannot send a URL request beginning with `http://` over a channel that supports only `https://`.

Standard WebLogic Server Channels

WebLogic Server provides pre-configured channels that you do not have to explicitly define.

- Default channel—Every Managed Server has a default channel.
- Administrative channel—If you configure a domain-wide Administration Port, WebLogic Server configures an Administrative Channel for each Managed Server in the domain.

The Default Network Channel

Every WebLogic Server domain has a default channel that is generated automatically by WebLogic Server. The default channel is based on the Listen Address and Listen Port defined in the `ServerMBean` and `SSLMBean`. It provides a single Listen Address, one port for HTTP communication (7001 by default), and one port for HTTPS communication (7002 by default). You can configure the Listen Address and Listen Port using the Configuration-->General tab in the Administration Console; the values you assign are stored in attributes of the `ServerMBean` and `SSLMBean`.

The default configuration may meet your needs if:

- You are installing in a test environment that has simple network requirements.
- Your server uses a single NIC, and the default port numbers provide enough flexibility for segmenting network traffic in your domain.

Using the default configuration ensures that third-party administration tools remain compatible with the new installation, because network configuration attributes remain stored in `ServerMBean` and `SSLMBean`.

Even if you define and use custom network channels for your domain, the default channel settings remain stored in `ServerMBean` and `SSLMBean`, and are used if necessary to provide connections to a server instance.

Note: Messages sent via the default channel can contain DNS information about the hosts they originate on or are destined to. If a T3 connection is established across a firewall that has network address translation (NAT) enabled, it is possible that some information about the network configuration behind the firewall will be revealed. Using the firewall to prevent T3 connections through the firewall will prevent this problem.

Administration Port and Administrative Channel

You can define an optional administration port for your domain. When configured, the administration port is used by each Managed Server in the domain exclusively for communication with the domain's Administration Server.

Administration Port Capabilities

An administration port enables you to:

- Start a server in standby state. This allows you to administer a Managed Server, while its other network connections are unavailable to accept client connections. For more information on the standby state, see "STANDBY" on page 7-4.

- Separate administration traffic from application traffic in your domain. In production environments, separating the two forms of traffic ensures that critical administration operations (starting and stopping servers, changing a server's configuration, and deploying applications) do not compete with high-volume application traffic on the same network connection.
- Administer a deadlocked server instance using the `weblogic.Admin` command line utility. If you do not configure an administration port, administrative commands such as `THREAD_DUMP` and `SHUTDOWN` will not work on deadlocked server instances.

If a administration port is enabled, WebLogic Server automatically generates an administration channel based on the port settings upon server instance startup.

Administration Port Restrictions

The administration port accepts only secure, SSL traffic, and all connections via the port require authentication. Enabling the administration port imposes the following restrictions on your domain:

- The Administration Server and all Managed Servers in your domain must be configured with support for the SSL protocol. Managed Servers that do not support SSL cannot connect with the Administration Server during startup—you will have to disable the administration port in order to configure them.
- Because all server instances in the domain must enable or disable the administration port at the same time, you configure the administration port at the domain level. You can change an individual Managed Server's administration port number, but you cannot enable or disable the administration port for an individual Managed Server. The ability to change the port number is useful if you have multiple server instances with the same Listen Address.
- After you enable the administration port, you must establish an SSL connection to the Administration Server in order to start any Managed Server in the domain. This applies whether you start Managed Servers manually, at the command line, or using Node Manager. For instructions to establish the SSL connection, see "Booting Managed Servers to Use Administration Port" on page 11-7.
- After enabling the administration port, all Administration Console traffic *must* connect via the administration port.
- If multiple server instances run on the same computer in a domain that uses a domain-wide administration port, you must either:
  - Host the server instances on a multi-homed machine and assign each server instance a unique listen address, or

– Override the domain-wide port on all but one of one of the servers instances on the machine. Override the port using the Local Administration Port Override option on the Advanced Attributes portion of the Server->Connections->SSL Ports page in the Administration Console.

Administration Port Requires SSL

The administration port requires SSL, which is enabled by default when you install WebLogic Server. If SSL has been disabled for any server instance in your domain, including the Administration Server and all Managed Servers, re-enable it using the Server--> Configuration-->General tab in the Administration Console.

Ensure that each server instance in the domain has a configured default listen port or default SSL listen port. The default ports are those you assign on the Server-->Configuration-->General tab in the Administration Console. A default port is required in the event that the server cannot bind to its configured administration port. If an additional default port is available, the server will continue to boot and you can change the administration port to an acceptable value.

By default WebLogic Server is configured to use demonstration certificate files. To configure production security components, follow the steps in "Configuring the SSL Protocol" in *Managing WebLogic Security*.

Configure Administration Port

Enable the administration port as described in "Enabling the Domain-Wide Administration Port" in *Administration Console Online Help*.

After configuring the administration port, you must restart the Administration Server and all Managed Servers to use the new administration port.

Booting Managed Servers to Use Administration Port

If you reboot Managed Servers at the command line or using a start script, specify the Administration Port in the port portion of the URL. The URL must specify the `https://` prefix, rather than `http://`, as shown below.

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

Note: If you use Node Manager for restarting the Managed Servers, it is not necessary to modify startup settings or arguments for the Managed Servers. Node Manager automatically obtains and uses the correct URL to start a Managed Server.

If the hostname in the URL is not identical to the hostname in the Administration Server's certificate, disable hostname verification in the command line or start script, as shown below:

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

Custom Administrative Channels

If the standard WebLogic Server administrative channel does not satisfy your requirements, you can configure a custom channel for administrative traffic. For example, a custom administrative channel allows you to segregate administrative traffic on a separate NIC.

To configure a custom channel for administrative traffic, configure the channel as described in "Configuring a Channel" on page 11-8, and select "admin" as the channel protocol. Note the configuration and usage guidelines described in:

- "Administration Port Requires SSL" on page 11-7
- "Booting Managed Servers to Use Administration Port" on page 11-7

Configuring a Channel

You can configure a network channel using Servers-->Protocols-->Channels tab in the Administration Console or using the `NetworkAccessPointMBean`.

For instructions to configure a channel for a non-clustered Managed Server, see "Configuring a Network Channel" in *Administration Console Online Help*. To configure a channel for clustered Managed Servers see, "Configuring Network Channels with a Cluster" on page 11-10.

For a summary of key facts about network channels, and guidelines related to their configuration, see "Configuring Channels: Facts and Rules" on page 11-8.

Configuring Channels: Facts and Rules

Follow these guidelines when configuring a channel.

Channels and Server Instances

- Each channel you configure for a particular server instance must have a unique combination of listen address, listen port, and protocol.
- A channel can be assigned to a single server instance.
- You can assign multiple channels to a server instance.
- If you assign non-SSL and SSL channels to the same server instance, make sure that they do not use the same combination of address and port number.

Configuration Changes are Not Dynamic

- After creating a new channel, restart the server instance for the channel settings to take effect. Similarly, you must restart the server instance for most channel configuration changes to take effect.

Channels and Protocols

- Some protocols do not support particular features of channels. In particular the COM protocol does not support SSL or tunneling.
- You must define a separate channel for each protocol you wish the server instance to support, with the exception of HTTP.

  HTTP is enabled by default when you create a channel, because RMI protocols typically require HTTP support for downloading stubs and classes. You can disable HTTP support on the Advanced Options portion of Servers-->Protocols-->Channels tab in the Administration Console.

Reserved Names

- WebLogic Server uses the internal channel names `.WLDefaultChannel` and `.WLDefaultAdminChannel` and reserves the `.WL` prefix for channel names. do not begin the name of a custom channel with the string `.WL`.

Channels, Proxy Servers, and Firewalls

If your configuration includes a a firewall between a proxy web server and a cluster (as described in "Firewall Between Proxy Layer and Cluster", in *Using WebLogicServer Clusters*, and the clustered servers are configured with two custom channels for segregating https and http traffic, those channels must share the same listen address. Furthermore if both http and https traffic needs to be supported there must be a custom channel for each—it is not possible to use the default configuration for one or the other.

If either of those channels has a `PublicAddress` defined, as is likely given existence of firewall both channels must define `PublicAddress`, and they both must define the same `PublicAddress`.

Configuring Network Channels with a Cluster

To configure a channel for clustered Managed Servers, note the information in "Configuring Channels: Facts and Rules" on page 11-8, and follow the guidelines described in the following sections.

Create the Cluster

If you have not already configured a cluster you can:

- Use the Configuration Wizard to create a new, clustered domain, following the instructions in "Create a Clustered Domain" in *Using WebLogic Clusters*, or
- Use the Administration Console to create a cluster in an existing domain, following the instructions "Configuring a Cluster" in *Administration Console Online Help*.

For information and guidelines about configuring a WebLogic Server cluster, see "Before You Start" in *Using WebLogic Clusters*.

Create and Assign the Network Channel

Use the instructions in "Configuring a Network Channel" in *Administration Console Online Help* to create a new network channel for each Managed Server in the cluster. When creating the new channels:

- For each channel you want to use in the cluster, configure the channel identically, including its name, on each Managed Server in the cluster.
- Make sure that the listen port and SSL listen port you define for each Managed Server's channel are different than the Managed Server's default listen ports. If the custom channel specifies the same port as a Managed Server's default port, the custom channel and the Managed Server's default channel will each try to bind to the same port, and you will be unable to start the Managed Server.
- If a cluster address has been configured for the cluster, it will be appear in the Cluster Address field on the Server-->Protocols-->Channels-->Configuration tab. If a cluster address has not been configured, supply it when configuring the channel. The network channel requires a cluster address to generate EJB handles and failover addresses for use with the cluster. for information on cluster addressing, see "Cluster Address" in *Using WebLogic Clusters*.

Increase Packet Size When Using Many Channels

Use of more than about twenty channels in a cluster can result in the formation of multicast header transmissions that exceed the default maximum packet size. The `MTUSize` attribute in the `Server` element of `config.xml` sets the maximum size for packets sent using the associated network card to 1500. Sending packets that exceed the value of `MTUSize` can result in a `java.lang.NegativeArraySizeException`. You can avoid exceptions that result from packet sizes in excess of `MTUSize` by increasing the value of `MTUSize` from its default value of 1500.

APPENDIX A

Starting and Stopping Servers: Quick Reference

The following sections describe simple, frequently used ways to start and shut down instances of WebLogic Server:

- “Starting Instances of WebLogic Server” on page A-2
- “Shutting Down Instances of WebLogic Server” on page A-5

For a comprehensive discussion of starting and shutting down WebLogic Server instances, refer to "Starting and Stopping Servers" in the *Administration Console Online Help*.

Starting Instances of WebLogic Server

In the following table, *WL_HOME* refers to the top-level installation directory for WebLogic Platform.

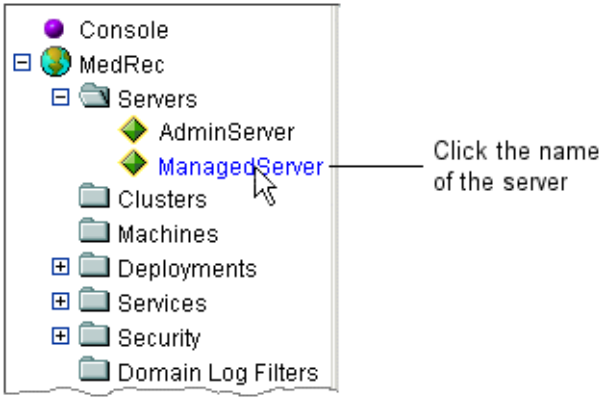
Table 11-1 Starting Server Instances

To Start	Do The Following
The MedRecServer sample server	Invoke: *WL_HOME*\samples\domains\medrec\startMedRecServer.cmd (Windows) *WL_HOME*/samples/domains/medrec/startMedRecServer.sh (UNIX) The server starts as an Administration Server in the MedRec domain. On Windows, you can also start MedRecServer from the Start menu.
The Examples server	Invoke: *WL_HOME*\samples\domains\examples\startExamplesServer.cmd (Windows) *WL_HOME*/samples/domains/examples/startExamplesServer.sh (UNIX) The server starts as an Administration Server in the Examples domain. On Windows, you can also start ExamplesServer from the Start menu.

Table 11-1 Starting Server Instances

To Start	Do The Following
An Administration Server that you have created	Invoke: *domain_directory*\startWebLogic.cmd (Windows) *domain_directory*/startWebLogic.sh (UNIX) where *domain_directory* is the directory in which you created the domain. If the server prompts you to enter a username and password, enter the name of a WebLogic Server user who has permission to start servers. For more information, refer to "Providing Usernames and Passwords to Start a Server" in the *Administration Console Online Help*. **Note:** In a development environment, it is usually sufficient to start an Administration Server and deploy your applications directly onto the Administration Server. In a production environment, you typically create Managed Servers to run applications. On Windows, you can instruct the Domain Configuration Wizard to create a shortcut on the Start menu to start the server that you create. You can also instruct the wizard to install the server as a Windows service. If you do so, the server starts automatically when you boot the host machine.

Table 11-1 Starting Server Instances

To Start	Do The Following
A Managed Server that you have created	1. Start the domain's Administration Server. (If you have configured the server as a Windows service, it starts automatically when you boot the Windows host computer.) 2. Start the Node Manager on the computer that will host the Managed Server you want to start. (If you have configured the Node Manager as a Windows service, it starts automatically when you boot the Windows host computer.) For more information, refer to "Starting and Stopping Node Manager" on page 5-8. 3. Start the domain's Administration Console. For more information, refer to "Starting the Administration Console." 4. In left pane of the Administration Console, expand the Servers folder. 5. Click on the name of the server. (See Figure 11-1.) **Figure 11-1 Click on the Name of the Server**  6. In the right pane, select the Control →Start/Stop tab. 7. Select Start this server and click Yes to confirm. For information additional ways to start Managed Servers, refer to "Starting and Stopping Servers."

Shutting Down Instances of WebLogic Server

The recommended procedure for shutting down a server is as follows:

1. Start the domain's Administration Console. See "Starting the Administration Console" in the *Administration Console Online Help*.
2. In left pane of the Administration Console, expand the Servers folder.
3. Click on the name of a server. (See Figure 11-1.)
4. In the right pane, select the Control →Start/Stop tab.
5. Select Shutdown this server and click Yes to confirm.

This initiates a graceful shutdown, in which the server notifies subsystems to complete all in-work requests. After the subsystems complete their work, the server stops.

Index