



BEA WebLogic Server®

Managing Server Startup and Shutdown

Version 10.0
Revised: March 30, 2007

Contents

1. Introduction and Roadmap

Document Scope and Audience	1-1
Guide to This Document	1-1
Related Documentation	1-2
New and Changed Features for Managing Server Life Cycle	1-2

2. Starting and Stopping Servers

Starting Servers: Before You Begin	2-2
Version Requirements for a Domain	2-2
Starting an Administration Server with a Startup Script	2-3
Starting an Administration Server from the Windows Start Menu	2-4
Starting an Administration Server with the java weblogic.Server Command	2-4
Starting an Administration Server Using WLST and Node Manager	2-5
Starting an Administration Server Using WLST Without Node Manager	2-5
Starting Managed Servers with a Startup Script	2-5
Starting Managed Servers from the Administration Console	2-7
Starting Managed Servers and Clusters with WLST and Node Manager	2-7
Starting Managed Servers with the java weblogic.Server Command	2-8
Starting a Managed Server When the Administration Server Is Unavailable	2-8
Provide User Credentials to Start and Stop Servers	2-8
Specifying an Initial Administrative User for a Domain	2-9
Boot Identity Files	2-10

Creating a Boot Identity File for an Administration Server	2-11
Using java weblogic.Server to Create a Boot Identity File for an Administration Server	2-11
Creating Boot Identity Files for Managed Servers	2-12
How a Server Uses a Boot Identity File at Startup	2-13
Removing Boot Identity Files After Startup	2-14
Specifying User Credentials for Starting a Server with Node Manager	2-15
Other Startup Tasks	2-15
Making Java Classfiles Globally Available	2-15
Configuring Managed Server Connections to the Administration Server	2-16
Specifying Java Options for a WebLogic Server Instance	2-18
Changing the JVM That Runs Servers	2-19
Shutting Down Instances of WebLogic Server	2-20
Shutting Down Servers with a Stop Script	2-20
Killing the JVM	2-21

3. Using Node Manager to Control Servers

Overview of Node Manager	3-1
Java-based and Script-based Node Manager	3-2
Java-based Node Manager	3-2
Script-based Node Manager	3-2
Determining Which Node Manager Version to Use	3-3
Accessing Node Manager	3-3
What You Can Do with Node Manager	3-4
Start, Shut Down, and Restart an Administration Server	3-4
Start, Shut Down, Suspend, and Restart Managed Servers	3-4
Restart Administration and Managed Servers	3-5
Monitor Servers and View Log Data	3-5

How Node Manager Works in the WebLogic Server Environment	3-5
Diagram of Node Manager and Servers	3-6
How Node Manager Starts an Administration Server	3-6
How Node Manager Starts a Managed Server	3-7
How Node Manager Restarts an Administration Server	3-9
How Node Manager Restarts a Managed Server	3-9
How Node Manager Shuts Down a Server Instance	3-11
Node Manager and System Crash Recovery	3-12
Node Manager Configuration and Log Files	3-13
Configuration Files	3-13
nodemanager.properties	3-14
nodemanager.domains	3-14
nm_data.properties	3-14
nm_password.properties	3-14
boot.properties	3-14
startup.properties	3-14
server_name.addr	3-15
server_name.lck	3-15
server_name.pid	3-15
server_name.state	3-15
Log Files	3-16
nodemanager.log	3-16
server_name.out	3-16
WebLogic Server Log Files	3-17
General Node Manager Configuration	3-17
Configuring Your Computer to Run Node Manager	3-17
Controlling and Configuring Node Manager Using WLST	3-18
Using nmConnect() in a Production Environment	3-18

Specifying Node Manager Username and Password	3-19
Configuring a Machine to Use Node Manager	3-19
Configuring nodemanager.domains File	3-19
Configuring Remote Startup Arguments	3-20
Setting Server Startup Properties	3-21
startup.properties	3-21
Setting Startup Properties Using Administration Utilities	3-21
Server Startup Properties	3-21
Ensuring Administration Server Address Is Defined	3-22
Setting Node Manager Environment Variables	3-22
Configuring Java-based Node Manager	3-23
Reconfigure Startup Service for Windows Installations	3-24
Configuring Java-based Node Manager Security	3-24
Remote Server Start Security for Java-based Node Manager	3-25
Reviewing nodemanager.properties	3-25
Deprecated Node Manager Properties	3-31
Configuring Node Manager to Use Start and Stop Scripts	3-33
Using Start Scripts	3-33
Using Stop Scripts	3-34
Using SSL With Java-based Node Manager	3-34
Configuring Node Manager on Multiple Machines	3-35
Configuring Node Manager as an xinetd Service	3-35
Configuring Script-based Node Manager	3-36
Creating a Node Manager User	3-36
Overriding the Default SSH Port	3-36
Configuring Script-based Node Manager Security	3-37
Security for WebLogic Server Scripts	3-37
Remote Server Start Security for Script-based Node Manager	3-37

Generating and Distributing Key Value Pairs	3-38
Starting Node Manager and Servers	3-39
Running Node Manager as a Startup Service.	3-39
Starting Java-based Node Manager Using Scripts	3-40
Command Syntax for Starting Java-based Node Manager	3-40
Running Script-based Node Manager	3-41
Stopping Node Manager.	3-43

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

Setting Up a Windows Service: Main Steps	4-2
Creating a Server-Specific Script	4-2
Configuring a Connection to the Administration Server	4-5
Requiring Managed Servers to Start After Administration Servers	4-6
Enabling Graceful Shutdowns from the Windows Control Panel.	4-8
Redirecting Standard Out and Standard Error to a File.	4-11
Adding Classes to the Classpath	4-14
Run the Server-Specific Script	4-15
Verifying the Setup.	4-16
Verifying the User Account Under Which the Service Runs	4-16
Using the Control Panel to Stop or Restart a Server Instance	4-17
Removing a Server as a Windows Service.	4-18
Changing Startup Credentials for a Server Set Up as a Windows Service.	4-19

5. Avoiding and Recovering From Server Failure

Failure Prevention and Recovery Features	5-2
Overload Protection	5-2
Failover for Clustered Services	5-2
Automatic Restart for Failed Server Instances.	5-2

Server-Level Migration	5-3
Service-Level Migration	5-3
Managed Server Independence Mode	5-3
Directory and File Backups for Failure Recovery	5-4
Back Up Domain Configuration Directory	5-4
Back Up LDAP Repository	5-4
Back Up SerializedSystemIni.dat and Security Certificates	5-5
WebLogic Server Exit Codes and Restarting After Failure	5-6
Restarting a Failed Administration Server	5-6
Restarting an Administration Server	5-6
Restarting an Administration Server on the Same Machine.	5-7
Restarting an Administration Server on Another Machine	5-8
Managed Servers and Re-started Administration Server	5-9
Restarting a Failed Managed Server	5-9
Starting a Managed Server When the Administration Server Is Accessible	5-9
Starting a Managed Server When the Administration Server Is Not Accessible.	5-10
Understanding Managed Server Independence Mode	5-10
MSI Mode and Node Manager	5-10
MSI Mode and the Security Realm.	5-10
MSI Mode and SSL	5-11
MSI Mode and Deployment	5-11
MSI Mode and the Domain Log File	5-11
MSI Mode and Managed Server Configuration Changes.	5-11
Starting a Managed Server in MSI Mode	5-11
Additional Failure Topics	5-12

6. Understanding Server Life Cycle

Diagram of the Server Life Cycle	6-1
--	-----

Getting and Using Server State	6-2
Understanding Server States in the Server Life Cycle	6-3
SHUTDOWN State	6-3
STARTING State	6-4
STANDBY State	6-9
ADMIN State	6-10
RESUMING State	6-10
RUNNING State	6-11
SUSPENDING State	6-11
FORCE_SUSPENDING State	6-11
SHUTTING_DOWN State	6-12
FAILED State	6-12
Using Server Life Cycle Commands	6-12
Start	6-13
Start in Standby	6-13
Start in Admin	6-14
Resume	6-14
Graceful Suspend	6-14
Force Suspend	6-15
Graceful Shutdown	6-15
Controlling Graceful Shutdown	6-15
Shutdown Operations and Application Undeployment	6-16
Force Shutdown	6-16
Processing In-Flight Work During Suspend and Shutdown	6-17
RMI Subsystem	6-17
Web Container	6-17
Timer Service	6-18
Application Service	6-18

EJB Container	6-18
JMS Service	6-18
JDBC Service	6-19
Transaction Service	6-19

A. Starting and Stopping Servers: Quick Reference

Starting Instances of WebLogic Server	A-1
Shutting Down Instances of WebLogic Server	A-3

Introduction and Roadmap

The following sections describe the contents and organization of this guide—*Managing Server Startup and Shutdown*.

- [“Document Scope and Audience”](#) on page 1-1
- [“Guide to This Document”](#) on page 1-1
- [“Related Documentation”](#) on page 1-2
- [“New and Changed Features for Managing Server Life Cycle”](#) on page 1-2

Document Scope and Audience

This document describes how you manage BEA WebLogic Server[®] startup, shutdown, and server life cycle. It also describes WebLogic features that you help prevent and recover from server failure.

This document is a resource for system administrators and operators responsible for monitoring and managing a WebLogic Server installation. It is relevant to all phases of a software project, from development through test and production phases.

It is assumed that the reader is familiar with Java Platform, Enterprise Edition (Java EE) and Web technologies, object-oriented programming techniques, and the Java programming language.

Guide to This Document

The document is organized as follows:

- This chapter, [“Introduction and Roadmap,”](#) describes the scope of the guide and lists related documentation.
- [Chapter 2, “Starting and Stopping Servers,”](#) describes several ways to start and stop server instances.
- [Chapter 3, “Using Node Manager to Control Servers,”](#) describes using Node Manager for the remote control of Administration and Managed Server instances.
- [Chapter 4, “Setting Up a WebLogic Server Instance as a Windows Service,”](#) describes procedures to set up a WebLogic Server instance as a Windows service in a Windows host computer.
- [Chapter 5, “Avoiding and Recovering From Server Failure,”](#) describes failover procedures for WebLogic Server instances.
- [Chapter 6, “Understanding Server Life Cycle,”](#) describes the operational phases of a WebLogic Server instance, from start up to shut down.
- [Appendix A, “Starting and Stopping Servers: Quick Reference,”](#) provides simple procedures for starting and stopping WebLogic Server instances.

Related Documentation

- [Creating WebLogic Domains Using the Configuration Wizard](#)
- [Understanding Domain Configuration](#)
- [Administration Console Online Help](#)

New and Changed Features for Managing Server Life Cycle

For a comprehensive listing of the new WebLogic Server features introduced in this release, see [“What's New in WebLogic Server 10”](#) in *Release Notes*.

Starting and Stopping Servers

WebLogic Server provides several ways to start and stop server instances. The method that you choose depends on whether you prefer using the Administration Console or a command-line interface, and on whether you are using Node Manager to manage a server's life cycle.

No matter how you start a server, the end result passes a set of configuration options to initialize a Java Virtual Machine (JVM). The server instance runs within the JVM, and the JVM can host only one server instance.

Note: For procedures that require the Administration Console, see [“Start and Stop Servers”](#) and various startup and shutdown procedures in the Cluster section of the *Administration Console Online Help*. For information on restarting failed server instances and clusters, see [Chapter 5, “Avoiding and Recovering From Server Failure.”](#)

The following sections describe other methods of starting and stopping server instances:

- [“Starting Servers: Before You Begin”](#) on page 2-2
- [“Version Requirements for a Domain”](#) on page 2-2
- [“Starting an Administration Server with a Startup Script”](#) on page 2-3
- [“Starting an Administration Server from the Windows Start Menu”](#) on page 2-4
- [“Starting an Administration Server with the java weblogic.Server Command”](#) on page 2-4
- [“Starting an Administration Server Using WLST and Node Manager”](#) on page 2-5
- [“Starting an Administration Server Using WLST Without Node Manager”](#) on page 2-5

- [“Starting Managed Servers with a Startup Script” on page 2-5](#)
- [“Starting Managed Servers from the Administration Console” on page 2-7](#)
- [“Starting Managed Servers and Clusters with WLST and Node Manager” on page 2-7](#)
- [“Starting Managed Servers with the java weblogic.Server Command” on page 2-8](#)
- [“Starting a Managed Server When the Administration Server Is Unavailable” on page 2-8](#)
- [“Provide User Credentials to Start and Stop Servers” on page 2-8](#)
- [“Other Startup Tasks” on page 2-15](#)
- [“Shutting Down Instances of WebLogic Server” on page 2-20](#)

For a concise overview of starting and stopping servers, see [“Starting and Stopping Servers: Quick Reference” on page A-1](#).

Starting Servers: Before You Begin

Depending on the method you choose to manage server startup and what setup tasks you have already performed, you might need to complete the following procedures before you can start server instances:

- Meet version requirements—[“Version Requirements for a Domain” on page 2-2](#)
- Create a domain—[“Choosing the Appropriate Technology for Your Administrative Tasks” in *Introduction to WebLogic Server*](#)
- Provide user credentials—[“Provide User Credentials to Start and Stop Servers” on page 2-8](#)
- Set up Node Manager—[“General Node Manager Configuration” on page 3-17](#)
- Configure Managed Server connections to the Administration Server—[“Configuring Managed Server Connections to the Administration Server” on page 2-16](#)
- Specify Java startup options—[“Specifying Java Options for a WebLogic Server Instance” on page 2-18](#)

Version Requirements for a Domain

The Administration Server and all Managed Servers in a domain must be the same WebLogic Server version. The Administration Server must be either at the same service-pack level or at a later service-pack level than the Managed Servers. For example, if the Managed Servers are at

version 8.1, then the Administration Server can be either version 8.1, 8.1 SP1 or higher. However, if the Managed Servers are at SP1, then the Administration Server must be at SP1 or higher.

Starting an Administration Server with a Startup Script

An Administration Server is a WebLogic Server instance that maintains configuration data for a domain. 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 create Managed Servers to run applications. For more information about Administration Servers and Managed Servers, see “[Understanding WebLogic Server Domains](#)” in *Understanding Domain Configuration*.

You can start an Administration Server with a default startup script or create your own. To start an Administration Server with the WebLogic Server-included startup script:

1. If you have not already done so, use the Configuration Wizard or WebLogic Scripting Tool (WLST) to create a domain.

See [Creating WebLogic Domains Using the Configuration Wizard](#) or “[Creating and Configuring WebLogic Domains Using WLST Offline](#)” in *WebLogic Scripting Tool*.

2. Open a shell (command prompt) on the computer on which you created the domain.
3. Change to the directory in which you located the domain.

By default, this directory is `BEA_HOME\user_projects\domains\DOMAIN_NAME`, where `DOMAIN_NAME` is the root directory of the domain. (The name of this directory is the name of the domain.)

4. Run one of the following scripts:

- `bin/startWebLogic.cmd` (Windows)
- `bin\startWebLogic.sh` (UNIX and Windows. On Windows, this script supports the MKS and Cygnus BASH UNIX shell emulators.)

Note: If you use a Configuration Wizard template that is provided by WebLogic Server, your domain directory includes a start script named `startWebLogic`. If you use a domain template from another source, the wizard might not create a start script, or it might create a script with a different name. The template designer determines whether the wizard creates a start script and the name of the script.

The `startWebLogic` script does the following:

Starting and Stopping Servers

1. Sets environment variables by invoking `DOMAIN_NAME\bin\setDomainEnv.cmd` (`setDomainEnv.sh` on UNIX), where `DOMAIN_NAME` is the directory in which you located the domain; for example, `WL_HOME\user_projects\domains\DOMAIN_NAME`, and where `WL_HOME` is the location in which you installed WebLogic Server.
2. Invokes the `java weblogic.Server` command, which starts a JVM that is configured to run a WebLogic Server instance.

When the server successfully completes its startup process, it writes the following message to standard out (which, by default, is the command window):

```
<Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>
```

Starting an Administration Server from the Windows Start Menu

When you create an Administration Server on a Windows computer, the Configuration Wizard creates a shortcut on the Start Menu for starting the server (User Projects→`DOMAIN_NAME`→Start Admin Server for WebLogic Domain).

The command that the Configuration Wizard adds to the Start menu opens a command window and calls the startup script that is described in [“Starting an Administration Server with a Startup Script” on page 2-3](#). When the server has successfully completed its startup process, it writes the following message to standard out (which, by default, is the command window):

```
<Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>
```

Starting an Administration Server with the `java weblogic.Server` Command

The `weblogic.Server` class is the main class for a WebLogic Server instance. You start a server instance by directly invoking `weblogic.Server` in a Java command. See [“`weblogic.Server` Command-Line Reference”](#) and [“Using the `weblogic.Server` Command Line to Start a Server Instance”](#) in *WebLogic Server Command Reference*.

Starting an Administration Server Using WLST and Node Manager

Node Manager is a utility for remote control of WebLogic Server instances. In previous versions, Node Manager required access to a running Administration Server, and could control and monitor only Managed Servers. In this release of WebLogic Server, Node Manager can also start, stop, and restart Administration Servers.

You can access these Node Manager features using the WebLogic Scripting Tool commands and scripts. If you use the `nmStart` command with WLST connected to a Node Manager, Node Manager supports monitoring, stopping, and restarting the Administration Server.

“[Using WLST and Node Manager to Manage Servers](#)” in *WebLogic Scripting Tool* describes how to start the Administration Server with WLST and Node Manager. “[How Node Manager Starts an Administration Server](#)” on page 3-6 describes how Node Manager accomplishes this process.

The WebLogic Server custom installation process optionally installs and starts Node Manager as a Windows service on Windows systems. BEA Systems recommends running Node Manager as an operating system service so that it automatically restarts in the event of system failure or reboot, and using Node Manager to start and restart both Administration and Managed Servers.

For more information, see “[About Node Manager Installation as a Windows Service](#)” in the *Installation Guide* and “[Restart Administration and Managed Servers](#)” on page 3-5.

Starting an Administration Server Using WLST Without Node Manager

The WLST `startServer` command starts the Administration Server without using Node Manager. The server runs in a separate process from WLST; exiting WLST does not shut down the server. See “[Starting an Administration Server Without Node Manager](#)” in *WebLogic Scripting Tool*.

Starting Managed Servers with a Startup Script

A Managed Server is a WebLogic Server instance that runs deployed applications. It refers to the Administration Server for all of its configuration and deployment information. Usually, you use Managed Servers to run applications in a production environment.

For more information about Managed Servers and Administration Servers, see “[Understanding WebLogic Server Domains](#)” in *Understanding Domain Configuration*.

Starting and Stopping Servers

If you use one of the Configuration Wizard templates that WebLogic Server provides, your domain directory includes a start script named `startManagedWebLogic` that you can use to start Managed Servers. You can use this script to start all the Managed Servers in a cluster.

For more information on domain directory files, see “[Domain Configuration Files](#)” in *Understanding Domain Configuration*.

This script does not use the Node Manager to start and manage the server. Instead, it uses a Java command to invoke the `weblogic.Server` class, which is the main class for a WebLogic Server instance. For information about invoking `weblogic.Server` in a Java command, see “[weblogic.Server Command-Line Reference](#)” in *WebLogic Server Command Reference*.

To use the WebLogic Server scripts to start Managed Servers:

1. Refer to “[Starting Servers: Before You Begin](#)” on page 2-2 for prerequisite tasks.
2. If you have not already done so, create one or more Managed Servers.
See *Creating WebLogic Domains Using the Configuration Wizard* or “[Create Managed Servers](#)” in the *Administration Console Online Help*.
3. Start the domain’s Administration Server.
4. In a shell (command prompt) on the computer that hosts the Managed Server, change to the directory that contains the `startManagedWebLogic` script:
`DOMAIN_NAME\bin\startManagedWebLogic.cmd` (Windows)
`DOMAIN_NAME/bin/startManagedWebLogic.sh` (UNIX)

where `DOMAIN_NAME` is the directory in which you located the domain. By default, this directory is `BEA_HOME\user_projects\domains\DOMAIN_NAME`.

5. Enter one of the following commands:
 - `startManagedWebLogic.cmd managed_server_name admin_url` (Windows)
 - `startManagedWebLogic.sh managed_server_name admin_url` (UNIX)

where `managed_server_name` specifies the name of the Managed Server and `admin_url` specifies the listen address (host name or IP address) and port number of the domain’s Administration Server.

For example, the following command uses `startManagedWebLogic.cmd` to start a Managed Server named `myManagedServer`. The listen address for the domain’s Administration Server is `AdminHost:7001`:

```
c:\bea\user_projects\domains\mydomain\bin\startManagedWebLogic.cmd  
myManagedServer http://AdminHost:7001
```

6. For each Managed Server that you want to start, open a separate command shell and follow steps 4 and 5. If you are starting Managed Servers on another machine, log in to that machine (remotely or locally) and then follow steps 4 and 5.

For information on running Managed Servers on a remote WebLogic Server host, see [“How Do I: Create and Start Managed Servers on a Remote Machine”](#) in *Create Templates and Domains Using the Pack and Unpack Commands*.

For information on configuring a connection to the Administration Server, see [“Configuring Managed Server Connections to the Administration Server”](#) on page 2-16.

The `startManagedWebLogic` script does the following:

1. Calls the `startWebLogic` script, which sets the environment variables by invoking `WL_HOME\user_projects\domains\DOMAIN_NAME\bin\setDomainEnv.cmd` (`setDomainEnv.sh` on UNIX), where `WL_HOME` is the location in which you installed WebLogic Server.
2. Invokes the `java weblogic.Server` command, which starts a JVM that is configured to run a WebLogic Server instance.

When the server successfully completes its startup process, it writes the following message to standard out (which, by default, is the command window):

```
<Notice> <WebLogicServer> <000360> <Server started in RUNNING mode>
```

Starting Managed Servers from the Administration Console

To use the Administration Console to start a Managed Server, see [“Start Managed Servers from the Administration Console”](#) in the *Administration Console Online Help*.

Starting Managed Servers and Clusters with WLST and Node Manager

To start Managed Servers and clusters using WLST and Node Manager, see [“Starting Managed Servers and Clusters With Node Manager”](#) in *WebLogic Scripting Tool*. For detailed information about WebLogic Server clusters, see [“Setting up WebLogic Clusters”](#) in *Using WebLogic Server Clusters*.

Starting Managed Servers with the `java weblogic.Server` Command

The `weblogic.Server` class is the main class for a WebLogic Server instance. You start a server instance by directly invoking `weblogic.Server` in a Java command. See “[weblogic.Server Command-Line Reference](#)” and “[Using the weblogic.Server Command Line to Start a Server Instance](#)” in *WebLogic Server Command Reference*.

Starting a Managed Server When the Administration Server Is Unavailable

Usually, a Managed Server contacts the Administration Server during its startup sequence to retrieve its configuration information. If a Managed Server cannot connect to the Administration Server during startup, it can retrieve its configuration by reading its locally cached configuration data from the `config` directory.

Note: The first time you start a Managed Server instance, it must be able to contact the Administration Server. Thereafter, the Managed Server instance can start even if the Administration Server is unavailable.

For more information on starting Managed Servers when the Administration Server is unavailable, see “[Starting a Managed Server When the Administration Server Is Not Accessible](#)” on page 5-10.

Provide User Credentials to Start and Stop Servers

To start and stop a WebLogic Server instance, you must provide the credentials of a user who is permitted to start and stop servers for the domain. For information on user credentials, roles, and permissions, see “[Users, Groups, And Security Roles](#)” in *Securing WebLogic Resources*.

[Table 2-1](#) describes providing user credentials when starting a WebLogic Server instance.

Table 2-1 Providing User Credentials

If you specify this...	The server instance does this...
Username and password on the command line.	Uses them and does not prompt you for either credential.
Username and password in <code>boot.properties</code> .	Uses them and does not prompt you for either credential.

Table 2-1 Providing User Credentials

Neither username nor password on the command line.	<ul style="list-style-type: none"> • Prompts you for the username. • Prompts you for the password twice.
Username but no password on the command line.	<ul style="list-style-type: none"> • Uses the username from the command line. • Prompts you for the password twice.
Password but no username on the command line.	<ul style="list-style-type: none"> • Prompts you for the username. • Ignores the password from the command line and prompts you for the password twice.

For more information on providing user credentials, see [“Specifying User Credentials”](#) in *WebLogic Server Command Reference*.

This section describes the following tasks:

- [“Specifying an Initial Administrative User for a Domain”](#) on page 2-9
- [“Boot Identity Files”](#) on page 2-10
- [“Specifying User Credentials for Starting a Server with Node Manager”](#) on page 2-15

Specifying an Initial Administrative User for a Domain

When you create a domain, the Configuration Wizard prompts you to provide the username and password for an initial administrative user. The Configuration Wizard does the following with this information:

1. Assigns the user to the Administrators security group.

The Administrators group grants the highest level of privileges for starting and managing WebLogic Server. For information on administrative privileges, see [“Users, Groups, And Security Roles”](#) in *Securing WebLogic Resources*.

2. Adds the user to the `myrealm` security realm.

A **security realm** is a collection of components (providers) that authenticate usernames, determine the type of resources that the user can access, and provide other security-related services for WebLogic resources. WebLogic Server installs the `myrealm` security realm and uses it by default.

You can use the Administration Console to add users to security realms. If you use an Authentication provider other than the one that WebLogic Server installs, you must use the provider's administration tools to create at least one user with administrative privileges.

3. If you are creating a domain in development mode, the wizard creates a boot identity file in the `security` directory of the Administration Server's root directory. The boot identity file contains an encrypted version of the username and password which lets you bypass the login prompt during subsequent instantiations of the server. See [“Boot Identity Files” on page 2-10](#).

In production domains, you are prompted to enter user credentials on the command line when booting the server.

Boot Identity Files

A boot identity file is a text file that contains user credentials for starting and stopping an instance of WebLogic Server. An Administration Server can refer to this file for user credentials instead of prompting you to provide them. Because the credentials are encrypted, using a boot identity file is much more secure than storing unencrypted credentials in a startup or shutdown script. If there is no boot identity file when starting a server, the server instance prompts you to enter a username and password.

If you start a Managed Server from a script that invokes the `java weblogic.Server` command (or if you invoke the `java weblogic.Server` command directly), a Managed Server can also refer to a boot identity file. If the Managed Server and Administration Server use the same root directory, the Managed Server can refer to the Administration Server's `boot.properties` file. If a Managed Server's `security` directory contains a valid `boot.properties` file, it uses this file during its startup process by default. The `boot.properties` file can be different for each server instance in the domain.

If you use the Node Manager to start a Managed Server, the Node Manager encrypts and saves the credentials with which it started the server in a server-specific `boot.properties` file for use in automatic restarts. This file is located in

`DOMAIN_NAME/servers/SERVER_NAME/data/nodemanager`, where `DOMAIN_NAME` is the name of the directory in which you located the domain and `SERVER_NAME` is the name of the server. For more information, see [“Node Manager Configuration and Log Files” on page 3-13](#).

The following sections describe working with boot identity files:

- [“Creating a Boot Identity File for an Administration Server” on page 2-11](#)
- [“Creating Boot Identity Files for Managed Servers” on page 2-12](#)
- [“How a Server Uses a Boot Identity File at Startup” on page 2-13](#)

- [“Removing Boot Identity Files After Startup” on page 2-14](#)

Creating a Boot Identity File for an Administration Server

If you use the Configuration Wizard to create a domain in development mode, the Configuration Wizard creates an encrypted boot identity file in the `security` directory of the Administration Server’s root directory. For more information on domain directory files, see [“Domain Directory Contents”](#) in *Understanding Domain Configuration*.

If a boot identity file for an Administration Server does not already exist, and if you want to bypass the prompt for username and password, create one as follows.

1. Start the Administration Server at least once and provide the user credentials on the command line.

During the Administration Server’s initial startup process, it generates security files that must be in place before a server can use a boot identity file.

2. Place the following two lines in a text file:

```
username=username
password=password
```

The username and password values must match an existing user account in the Authentication provider for the default security realm and must belong to a role that has permission to start and stop a server. For information on roles and permissions, see [“Users, Groups, And Security Roles”](#) in *Securing WebLogic Resources*.

3. Save the file.

If you save the file as `boot.properties` and locate it in the `security` directory of the server’s root directory, the server automatically uses this file during its subsequent startup cycles. For more information, see [“How a Server Uses a Boot Identity File at Startup” on page 2-13](#).

The first time you use this file to start a server, the server reads the file and then overwrites it with an encrypted version of the username and password.

Using `java weblogic.Server` to Create a Boot Identity File for an Administration Server

Note: Use this technique only if you invoke the `java weblogic.Server` command from the command line. If you use a script to start an Administration Server, BEA Systems recommends that you do **not** use the technique described in this section for the following reasons:

- It requires you to store an unencrypted password in the startup script.
- Each time you run the script, the server boots with the supplied user credentials and then creates a new boot identity file.

Instead of following the steps in the previous section, [“Creating a Boot Identity File for an Administration Server” on page 2-11](#), you can create a boot identity file by invoking the `weblogic.Server` class directly on the command line and including the following options in the Java command:

```
-Dweblogic.management.username=username
-Dweblogic.management.password=password
-Dweblogic.system.StoreBootIdentity=true
```

These options cause the server instance to boot with the supplied user credentials and then store them in a file named `boot.properties`.

For example, the following command starts an Administration Server named `myAdminServer` and creates a boot identity file:

```
java -Dweblogic.management.username=weblogic
-Dweblogic.management.password=weblogic
-Dweblogic.system.StoreBootIdentity=true
-Dweblogic.Name=myAdminServer weblogic.Server
```

For more information about invoking the `weblogic.Server` class directly from a command line, see [“weblogic.Server Command-Line Reference”](#) in *WebLogic Server Command Reference*.

Creating Boot Identity Files for Managed Servers

If a Managed Server uses the same root directory as the Administration Server, it can use the same boot properties file as the Administration Server. If you use a Node Manager to start a Managed Server, you do not need to create a boot identity file. For more information, see [“Node Manager Configuration and Log Files” on page 3-13](#).

To create a boot identity file for a Managed Server instance:

1. Start the domain’s Administration Server to make sure that the required security files are in the `security` directory of the Administration Server’s domain and root directories. If the files are not present, the Administration Server generates them.

For more information on domain directory files, see [“Domain Configuration Files”](#) in *Understanding Domain Configuration*.

2. Place the following two lines in a text file:

```
username=username
password=password
```

The username and password values must match an existing user account in the Authentication provider for the default security realm and must belong to a role that has permission to start a server. For information on roles and permissions, see “[Users, Groups, And Security Roles](#)” in *Securing WebLogic Resources*.

3. Save the file.

If you save the file as `boot.properties` and locate it in the `security` directory of the server’s root directory, the server automatically uses this file during its subsequent startup cycles. For more information, see “[How a Server Uses a Boot Identity File at Startup](#)” on [page 2-13](#).

4. Repeat steps 2 and 3 for each Managed Server in the domain for which you want to create a boot identity file.

The first time you use this file to start a server, the server reads the file and then overwrites it with an encrypted version of the username and password.

How a Server Uses a Boot Identity File at Startup

A server instance uses a boot identity file during its startup process as follows:

- If a server’s `security` directory contains a valid `boot.properties` file, it uses this file during its startup process by default. For information about a server’s root directory, see “[A Server’s Root Directory](#)” in *Understanding Domain Configuration*.
- If you want to specify a different file (or if you do not want to store boot identity files in a server’s `security` directory), you can include the following argument in the server’s `weblogic.Server` startup command:

```
-Dweblogic.system.BootIdentityFile=filename
```

where *filename* is the fully qualified pathname of a valid boot identity file.

To specify this argument in the `startWebLogic` script, add

```
-Dweblogic.system.BootIdentityFile as a value of the JAVA_OPTIONS variable. For example:
```

```
set
JAVA_OPTIONS=-Dweblogic.system.BootIdentityFile=C:\BEA\user_domains\mydomain\myidentity.prop
```

- If you do *not* want a server instance to use a boot identity file during its startup cycle, include the following options in the server’s `weblogic.Server` startup command:

```
-Dweblogic.management.username=username  
-Dweblogic.management.password=password
```

These options cause a server instance to ignore any boot identity files and override other startup options that cause a server to use boot identity files during its startup cycle.

Note: If you use a script to start a server instance, BEA Systems recommends that you do *not* use this technique because it requires you to store an unencrypted password in the startup script. Use this technique only if you invoke the `weblogic.Server` class directly from the command line. For more information, see “[weblogic.Server Command-Line Reference](#)” in *WebLogic Server Command Reference*.

- If a server is unable to access its boot identity file during its startup cycle, it displays the username and password prompt in its command shell and writes a message to the log file.

For a given server instance, use only the boot identity file that the instance has created. WebLogic Server does not support copying a boot identity file from one server root directory to another.

For example, if you use ServerA to generate a boot identity file, use only that boot identity file with ServerA. Do not copy ServerA’s boot identity file into the `security` directory of ServerB. Instead, create a boot identity file for ServerB as described in “[Creating a Boot Identity File for an Administration Server](#)” on page 2-11 or “[Creating Boot Identity Files for Managed Servers](#)” on page 2-12.

Removing Boot Identity Files After Startup

If you want to remove the boot identity file after a server starts, you can include the following argument in the server’s `weblogic.Server` startup command:

```
-Dweblogic.system.RemoveBootIdentity=true
```

This argument removes only the file that the server used to start. For example, if you specify

```
-Dweblogic.system.BootIdentityFile=c:\secure\boot.MyServer,
```

only `boot.MyServer` is removed, even if the server’s root directory contains a file named `boot.properties`. Open a separate command shell and include the

```
-Dweblogic.system.RemoveBootIdentity=true
```

argument in each Managed Server’s `weblogic.Server` startup command to remove its boot identity file.

To specify this argument in the `startWebLogic` script, add

```
-Dweblogic.system.RemoveBootIdentity=true
```

as a value of the `JAVA_OPTIONS` variable.

For example:

```
set JAVA_OPTIONS=-Dweblogic.system.RemoveBootIdentity=true
```

Specifying User Credentials for Starting a Server with Node Manager

If you use the Node Manager to start a Managed Server, you must provide user credentials on the server's Configuration→Server Start page of the Administration Console. If you do not provide these credentials, the Node Manager throws an exception when it tries to start the server.

When you use the Administration Console or the Configuration Wizard to create a Managed Server, WebLogic Server adds the user credentials to the server's Configuration→Server Start page. If you want the server instance to run under a different WebLogic Server user account, see [“Configure Startup Arguments for Managed Servers”](#) in the *Administration Console Online Help*.

Other Startup Tasks

The following sections describe miscellaneous startup tasks:

- [“Making Java Classfiles Globally Available.”](#)
- [Configuring Managed Server Connections to the Administration Server](#)
- [Specifying Java Options for a WebLogic Server Instance](#)
- [Changing the JVM That Runs Servers](#)

Making Java Classfiles Globally Available

There are two methods for making java classes globally available to WebLogic Server:

- Setting the `$DOMAIN_DIR/lib` environment variable.
- Specifying the `-Dweblogic.ext.dirs` startup option.

You can specify either or both of these methods. When specifying both, classes defined via the startup option take precedence.

Both are available in development and production modes. However, dynamic class loading is generally used in development environments, while the classpath method is preferable in production environments.

In both cases, you must ensure that your classes are packaged into `.jar` files.

Configuring Managed Server Connections to the Administration Server

If you will be starting a Managed Server from a script that invokes the `java weblogic.Server` command, or if you invoke the `java weblogic.Server` command directly, you must make sure that the Managed Server specifies the correct listen address of the Administration Server. A Managed Server uses this address to retrieve its configuration from the Administration Server.

Use the following format to specify the listen address:

```
[protocol://]Admin-host:port
```

1. For *protocol*, specify any of the following:

- t3
- t3s
- http
- https

If you will be using the domain-wide administration port, you must specify either T3S or HTTPS. If you do not specify a value, the servers use T3.

Note: Regardless of which protocol you use, the initial download of a Managed Server's configuration is over HTTP or HTTPS. After the RMI subsystem initializes, the server instance can use the T3 or T3S protocol.

2. For *Admin-host*, specify any of the following:

- localhost.

Valid only if you are starting the Managed Server on the same computer as the Administration Server.

- The DNS name of the computer that is hosting the Administration Server.
- The IP address of the computer that is hosting the Administration Server.

Because of the following security issue, BEA Systems recommends that you do not use IP addresses for *Admin-host* in a production environment:

To connect to the Administration Server through an SSL port, the Managed Server verifies that the Administration Server's host name matches the host name that is specified in the URL. If you specify an IP address, and if host name verification is enabled, the connection fails because the IP address, which is a series of numbers, does not match the name of the host, which is a string of characters.

In a development environment, where security is less of a concern, you can disable host name verification on the Managed Server so SSL connections that specify an IP address will succeed. See [“Using Hostname Verification”](#) in *Securing WebLogic Server*.

If the Administration Server has been configured to use some other listen address, you must specify the configured listen address.

3. For *port*, specify any of the following:

- The domain-wide administration port.

When configured, the administration port is used by each Managed Server in the domain exclusively for communication with the domain's Administration Server. See [“Configure the Domain-Wide Administration Port”](#) in the *Administration Console Online Help*.

If you have enabled the domain-wide administration port, you must specify this port. You must specify either the T3S or HTTPS protocol to use this port.

- The non-SSL listen port for the Administration Server's default network configuration (7001 by default).

If this listen port has been disabled for the Administration Server, you must use one of the other listen ports described in this list. You must specify either the T3 or HTTP protocol to use this port.

- The SSL listen port for the Administration Server's default network configuration (7002 by default).

If this listen port has been disabled for the Administration Server, you must use one of the other listen ports described in this list. You must specify either the T3S or HTTPS protocol to use this port.

- The port number that is associated with an optional, custom network channel.

If the port is secured with SSL, you must specify either the T3S or HTTPS protocol.

4. To verify the host IP address, name, and default listen port of the Administration Server, start the Administration Server in a shell (command prompt). When the server successfully finishes its startup cycle, it prints to standard out messages that are similar to the following (among other messages):

```
<Nov 5, 2004 12:16:04 PM EST> <Notice> <Server> <BEA-002613> <Channel
"DefaultSecure[2]" is now listening on 127.0.0.1:7012 for protocols
iiops, t3s, ldaps, https.>
```

...

```
<Nov 5, 2004 12:16:04 PM EST> <Notice> <WebLogicServer> <BEA-000331>  
<Started WebLogic Admin Server "MedRecServer" for domain "medrec"  
running in Development Mode>
```

For information on enabling SSL, see “[Set Up SSL](#)” in the *Administration Console Online Help*. For more information on network channels, see “[Understanding Network Channels](#)” in *Configuring WebLogic Server Environments*.

Specifying Java Options for a WebLogic Server Instance

You use Java options to configure operating parameters for the JVM that runs a WebLogic Server instance. For example, you use Java options to tune the performance and monitoring capabilities of the JRockit JVM.

You can also use Java options to override a server’s configuration temporarily. The Java options apply only to the current instance of the server. They are not saved in the domain’s `config.xml` file and they are not visible from the Administration Console. For example, if a server is configured to listen on port 7201, you can use a Java option to start the server so that it listens on port 7555. The Administration Console will still indicate that the server is configured to listen on port 7201. If you do not use the Java option the next time you start the server, it will listen on port 7201.

If you use a WebLogic Server script to start servers, do the following. If you use the Node Manager to start servers, see “[Set Java options for servers started by Node Manager](#)” in the *Administration Console Online Help*.

1. Create a backup copy of the WebLogic Server start scripts:
 - For scripts that start an Administration Server, back up
`DOMAIN_NAME\bin\startWebLogic.cmd` (`startWebLogic.sh` on UNIX)
 - For scripts that start a Managed Server, back up
`DOMAIN_NAME\bin\startManagedWebLogic.cmd` (`startManagedWebLogic.sh` on UNIX)

where `DOMAIN_NAME` is the name of the directory in which you located the domain. By default, this directory is `BEA_HOME\user_projects\domains\DOMAIN_NAME`.
2. Open the start script in a text editor.
3. Edit the `set JAVA_OPTIONS` command to specify the Java options. If you specify multiple options, separate each option by a space, and place quotes around the entire set of options. For example:

```
set JAVA_OPTIONS="-Xgc:gencopy -Xns:30"
```

For more information, see:

- “[weblogic.Server Command-Line Reference](#)” for information on the Java options that set runtime behavior of a WebLogic Server instance.
 - “[Using BEA JRockit JDK](#)” for information on the Java options that the JRockit Virtual Machine supports.
 - The documentation that the JVM vendor provides for information on the Java options that other JVMs support.
4. Save the start script.
 5. Start the server.

Changing the JVM That Runs Servers

When you create a domain, if you choose to customize the configuration, the Configuration Wizard presents a list of SDKs that WebLogic Server installed. From this list, you choose the JVM that you want to run your domain, and the wizard configures the BEA start scripts based on your choice.

After you create a domain, if you want to use a different JVM, you can modify the scripts as follows:

1. Change the value for the `JAVA_HOME` variable.

Specify an absolute pathname to the top directory of the SDK that you want to use. For example, `c:\bea\jrockit90`.

On a Windows or Linux platform, BEA Systems recommends the following JVMs:

- For development mode, the Sun SDK with the HotSpot Client JVM.
 - For production mode, the BEA JRockit[®] SDK. This SDK provides optimal running performance but initial startup cycles can require more time than other SDKs.
2. Change the value for the `JAVA_VENDOR` variable.

Specify the vendor of the SDK. Valid values depend on the platform on which you are running. For more information, see the WebLogic Platform Supported Configurations page at the following URL: <http://e-docs.bea.com/platform/supconfigs/index.html>.

For example:

- `BEA` indicates that you are using the JRockit SDK. It is valid only on platforms that support JRockit.

- Sun indicates that you are using the Sun SDK.
 - HP and IBM indicate that you are using SDKs that Hewlett Packard or IBM have provided. These values are valid only on platforms that support HP or IBM SDKs.
3. Restart any servers that are currently running.

Shutting Down Instances of WebLogic Server

It is recommended that you shutdown WebLogic Server instances through the Administration Console. See [“Shut Down a Server Instance”](#), [“Control Graceful Shutdowns”](#), and [“Shutdown servers in a cluster”](#) in the *Administration Console Online Help*.

On Windows, you can stop Administration Servers that you have created using the Configuration Wizard from the Start menu.

Shutting Down Servers with a Stop Script

If you use a Configuration Wizard template that is provided by WebLogic Server, the `bin` directory under your domain directory includes a stop script named `stopWebLogic` that you can use to stop an Administration Server and one named `stopManagedWebLogic` for stopping Managed Servers. To use the scripts, you must set `SERVER_NAME`, `ADMIN_URL`, `USERID`, and `PASSWORD` as environment variables or specify them on the command line. When using the `stopWebLogic` script, if you do not specify `SERVER_NAME`, the Administration Server name is used by default.

- For an Administration Server, invoke:

```
DOMAIN_NAME\bin\stopWeblogic.cmd username password admin_url (Windows)
DOMAIN_NAME/bin/stopWeblogic.sh username password admin_url (UNIX)
```

- For Managed Servers, invoke:

```
DOMAIN_NAME\bin\stopManagedWeblogic.cmd managed_server_name admin_url
username password (Windows)
DOMAIN_NAME/bin/stopManagedWeblogic.sh managed_server_name admin_url
username password (UNIX)
```

Note: On the command line, specify parameters in the order shown. User credentials come before the `ADMIN_URL` with `stopWebLogic.cmd` and after the `ADMIN_URL` with `stopManagedWebLogic.cmd`.

Killing the JVM

Each WebLogic Server instance runs in its own JVM. If you are unable to shut down a server instance using the methods described in the previous sections, you can use an operating system command to kill the JVM.

Caution: 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.

Some common ways to kill the JVM are as follows:

- If the shell (command prompt) in which you start the server is still open, you can type `Ctrl-C`.
- On a Windows computer, you can use the Task Manager to kill a JVM.
- On a UNIX computer, you can use the `ps` command to list all running processes. Then you can use the `kill` command to kill the JVM.

Starting and Stopping Servers

Using Node Manager to Control Servers

The following sections describe Node Manager functionality, architecture, and configuration procedures.

- [“Overview of Node Manager” on page 3-1](#)
- [“How Node Manager Works in the WebLogic Server Environment” on page 3-5](#)
- [“General Node Manager Configuration” on page 3-17](#)
- [“Configuring Java-based Node Manager” on page 3-23](#)
- [“Configuring Script-based Node Manager” on page 3-36](#)
- [“Starting Node Manager and Servers” on page 3-39](#)

Overview of Node Manager

Server instances in a WebLogic Server production environment are often distributed across multiple domains, machines, and geographic locations. Node Manager is a WebLogic Server utility that enables you to start, shut down, and restart Administration Server and Managed Server instances from a remote location. Although Node Manager is optional, it is recommended if your WebLogic Server environment hosts applications with high availability requirements.

A Node Manager process is not associated with a specific WebLogic domain but with a machine. You can use the same Node Manager process to control server instances in any WebLogic Server domain, as long as the server instances reside on the same machine as the Node Manager process.

Node Manager must run on each computer that hosts WebLogic Server instances -- whether Administration Server or Managed Server -- that you want to control with Node Manager.

Java-based and Script-based Node Manager

WebLogic Server provides two versions of Node Manager, Java-based and script-based, with similar functionality. However, each version has different configuration and security considerations.

Java-based Node Manager

Java-based Node Manager runs within a Java Virtual Machine (JVM) process. It is recommended that you run it as a Windows service on Windows platforms and as an operating service on UNIX platforms, allowing it to restart automatically when the system is rebooted.

BEA provides native Node Manager libraries for Windows, Solaris, HP UX, Linux on Intel, Linux on Z-Series, and AIX operating systems.

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

This version of Node Manager determines its configuration from the `nodemanager.properties` file. See [“Configuring Java-based Node Manager” on page 3-23](#).

Java-based Node Manager provides more security than the script-based version. See [“Configuring Java-based Node Manager Security” on page 3-24](#).

Script-based Node Manager

For UNIX and Linux systems, WebLogic Server provides a script-based version of Node Manager. This script is based on UNIX shell scripts, but uses SSH for increased security. SSH uses user-id based security.

For information on configuring the script version of Node Manager, see [“Configuring Script-based Node Manager” on page 3-36](#). For information on using this version of Node Manager, see [“Running Script-based Node Manager” on page 3-41](#).

This version does not provide as much security as the Java-based version. However, the advantage of the script-based Node Manager is that it can remotely manage servers over a network that has been configured to use SSH. No additional server installation is required. The scripts merely have to be copied to the remote machine.

Note: It is recommended that you run script-based Node Manager as an operating system service, which allows it to restart automatically when the system is rebooted.

Determining Which Node Manager Version to Use

Which version of Node Manager to use depends on the requirements of your WebLogic Server environment. The following considerations can help you decide which version is ideal for your environment:

- If you are installing WebLogic Server on a Windows system, you must use the Java version of Node Manager. The scripted version of Node Manager is not supported on Windows.
- In order to use db-less leasing (consensus leasing) you may see faster performance when using the Java version of Node Manager.
- The script based Node Manager requires a much simpler security configuration than the Java version. RSH and SSH are general easier to configure than SSL which is the method of security used by the Java version of Node Manager. The script version of Node Manager also required a smaller footprint than the Java version
- The Java version of Node Manager can be used in conjunction with `inetd` on supported UNIX systems. `inetd` allows Node Manager to be automatically restarted on upon receiving a request on the configured port.

Accessing Node Manager

A Node Manager client can be local or remote to the Node Managers with which it communicates. You access either version of Node Manager—the Java version or the script-based (SSH) version—from the following clients. (In addition, an SSH client in the form of a shell command template is provided for use with the script-based Node Manager.)

- Administration Server
 - Administration Console, from the Environments>Machines>Configuration>Node Manager page.
 - JMX utilities you write yourself.

For more information about JMX, see [Developing Custom Management Utilities with JMX](#).
- WLST commands and scripts—WLST offline serves as a Node Manager command-line interface that can run in the absence of a running Administration Server. You can use WLST commands to start, stop, and monitor a server instance without connecting to an Administration Server. Starting the Administration Server is the main purpose of the stand-alone client. However, you can also use it to:

- Stop a server instance that was started by Node Manager.
- Start a Managed Server.
- Access the contents of a Node Manager log file.
- Obtain server status.
- Retrieve the contents of server output log.

What You Can Do with Node Manager

The following sections describe basic Node Manager functionality.

Start, Shut Down, and Restart an Administration Server

Using the WebLogic Scripting Tool (or SSH client for Script-based Node Manager only), you connect to the Node Manager process on the machine that hosts the Administration Server and issue commands to start, shut down, or restart an Administrative Server. The relationship of an Administration Server to Node Manager varies for different scenarios.

- An Administration Server can be under Node Manager control—You can start it, monitor it, and restart it using Node Manager.
- An Administration Server can be a Node Manager client—When you start or stop Managed Servers from the Administration Console, you are accessing Node Manager via the Administration Server.
- An Administration Server supports the process of starting up a Managed Server with Node Manager—When you start a Managed Server with Node Manager, the Managed Server contacts the Administration Server to obtain outstanding configuration updates.

Start, Shut Down, Suspend, and Restart Managed Servers

From the WebLogic Server Scripting Tool (WLST) command line or scripts, you can issue commands to Node Manager to start, shut down, suspend, and restart Managed Server instances and clusters.

Node Manager can restart a Managed Server after failure even when the Administration Server is unavailable if Managed Server Independence (MSI) mode is enabled for that Managed Server instance. This is enabled by default.

Note: Node Manager cannot start a Managed Server for the first time in MSI mode, because at the Administration Server for the domain must be available so the Managed Server can obtain its configuration settings.

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

Restart Administration and Managed Servers

If a server instance that was started using Node Manager fails, Node Manager automatically restarts it.

Note: Node Manager can only restart a server that was started via Node Manager.

The restart feature is configurable. Node Manager’s default behavior is to:

- Automatically restart server instances under its control that fail. You can disable this feature.
- Restart failed server instances no more than a specific number of times. You define the number of restarts by setting the `RestartMax` property in the Node Manager `startup.properties` file.

If Node Manager fails or is explicitly shut down, upon restart, it determines the server instances that were under its control when it exited. Node Manager can restart any failed server instances as necessary.

Note: It is advisable to run Node Manager as an operating system service, so that it restarts automatically if its host machine is restarted.

Monitor Servers and View Log Data

Node Manager creates a log file for the Node Manager process and a log file of server output for each server instance it controls. You can view these log files, as well as log files for a server instance using the Administration Console or WLST commands.

How Node Manager Works in the WebLogic Server Environment

The following sections provide a “big picture” diagram of Node Manager’s role in the WebLogic Server environment, as well as illustrations and descriptions of the processes Node Manager uses to communicate with servers:

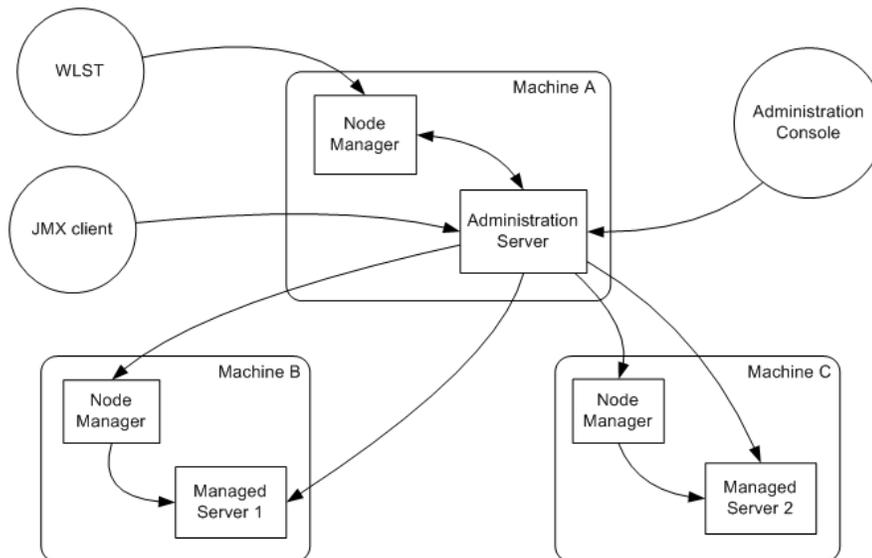
- “[Diagram of Node Manager and Servers](#)” on page 3-6

- [“How Node Manager Starts an Administration Server”](#) on page 3-6
- [“How Node Manager Starts a Managed Server”](#) on page 3-7
- [“How Node Manager Restarts an Administration Server”](#) on page 3-9
- [“How Node Manager Restarts a Managed Server”](#) on page 3-9
- [“How Node Manager Shuts Down a Server Instance”](#) on page 3-11

Diagram of Node Manager and Servers

Figure 3-1 illustrates the relationship between Node Manager, its clients, and the server instances it controls.

Figure 3-1 Node Manager in the WebLogic Server Environment



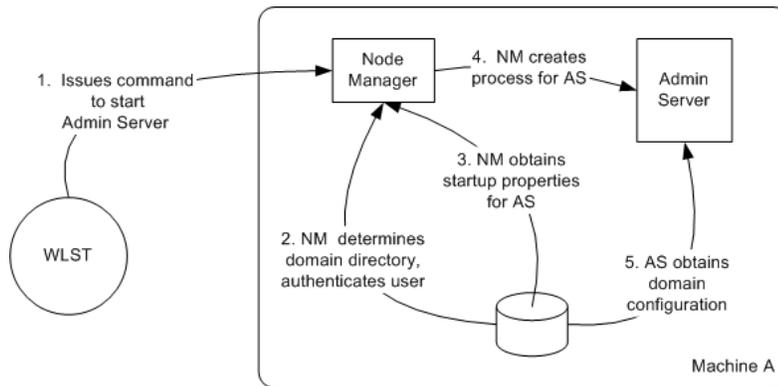
How Node Manager Starts an Administration Server

Figure 3-2 illustrates the process of starting an Administration Server with Node Manager.

This section assumes that you have installed the Administration Server and created its domain directory using the Configuration Wizard.

Node Manager is running on Machine A, which hosts the Administration Server. The stand-alone Node Manager client is remote.

Figure 3-2 Starting an Administration Server



1. An authorized user issues the WLST offline command, `nmConnect` to connect to the Node Manager process on the machine that hosts the Administration Server, and issues a command to start the Administration Server. (If the Node Manager instance is the SSH version, the user can connect using the SSH client).

The start command identifies the domain and server instance to start, and in the case of the Java Node Manager, provides the Node Manager username and password.

Note: If the user has previously connected to the Node Manager, a `boot.properties` file exists, and the user does not have to supply username and password.

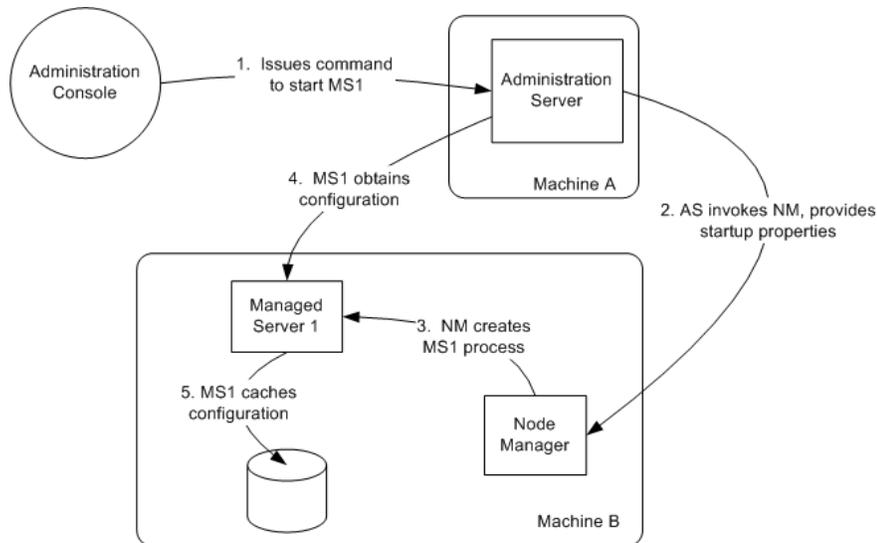
2. Node Manager looks up the domain directory in `nodemanager.domains`, and authenticates the user credentials using a local file that contains the encrypted username and password.
3. Node Manager creates the Administration Server process.
4. The Administration Server obtains the domain configuration from its `config` directory.

How Node Manager Starts a Managed Server

Figure 3-3 illustrates the process of starting a Managed Server with Node Manager.

Node Manager is running on Machine B, which hosts Managed Server 1. The Administration Server for the domain is running on Machine A.

Figure 3-3 Starting a Managed Server



1. From the Administration Console, the user issues a start command for Managed Server 1.
Note: A stand-alone client can also issue a start command for a Managed Server.
2. The Administration Server issues a start command for Managed Server 1 to the Node Manager on the Machine B, providing the remote start properties configured for Managed Server 1. For information about the arguments and how to specify them, see [“Configuring Remote Startup Arguments”](#) on page 3-20.
3. Node Manager starts Managed Server 1.
Node Manager starts the Managed Server using the same root 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—>Server Start console page.
4. Managed Server 1 contacts the Administration Server to check for updates to its configuration information.
5. If there are outstanding changes to the domain configuration, Managed Server 1 updates its local cache of configuration data.

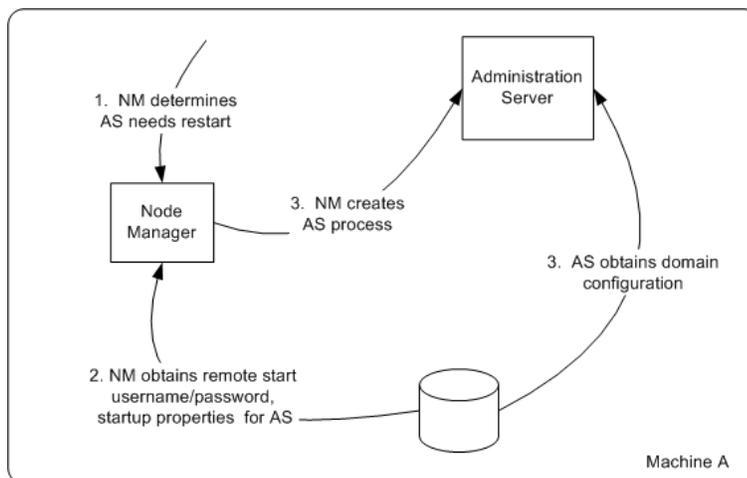
How Node Manager Restarts an Administration Server

Figure 3-4 illustrates the process of restarting an Administration Server with Node Manager.

Node Manager is running on the machine that hosts the Administration Server. The Administration Server, which was initially started with Node Manager, has exited. The Administration Server's `AutoRestart` attribute is set to `true`.

Note: If a server instance's `AutoRestart` attribute is set to `false`, Node Manager will not restart it.

Figure 3-4 Restarting an Administration Server



1. Node Manager determines from the Administration Server process exit code that it requires restart.
2. Node Manager obtains the username and password for starting the Administration Server from the `boot.properties` file, and the server startup properties from the `<server_name>/data/nodemanager/startup.properties` file.
3. Node Manager starts the Administration Server.
4. The Administration Server reads its configuration data and starts up.

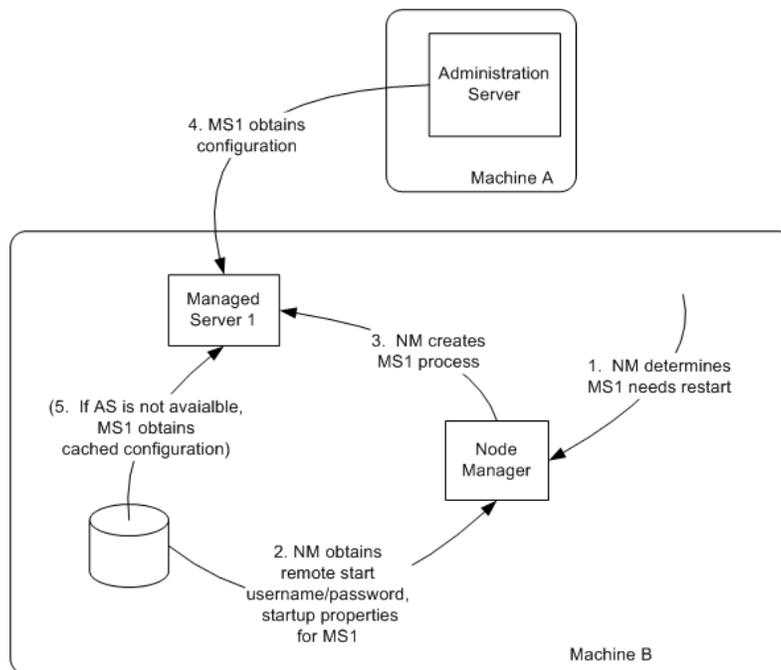
How Node Manager Restarts a Managed Server

Figure 3-5 illustrates process of restarting a Managed Server with Node Manager.

Node Manager is running on Machine B, which hosts Managed Server 1. Managed Server 1, which was initially started with Node Manager, has exited. Managed Server 1's `AutoRestart` attribute is set to `true`.

Note: If a server instance's `AutoRestart` attribute is set to `false`, Node Manager will not restart it.

Figure 3-5 Restarting a Managed Server



1. Node Manager determines from Managed Server 1's last known state that it requires restarting.
2. Node Manager obtains the username and password for starting Managed Server 1 from the `boot.properties` file, and the server startup properties from the `startup.properties` file. These server-specific files are located in the server directory for Managed Server 1.
3. Node Manager starts Managed Server 1.

Note: Node Manager waits `RestartDelaySeconds` after a server instances fails before attempting to restart it.

4. Managed Server 1 attempts to contact the Administration Server to check for updates to its configuration data. If it contacts the Administration Server and obtains updated configuration data, it updates its local cache of the `config` directory.
5. If Managed Server 1 fails to contact the Administration Server, and if Managed Server Independence mode (MSI) is enabled, Managed Server 1 uses its locally cached configuration data.

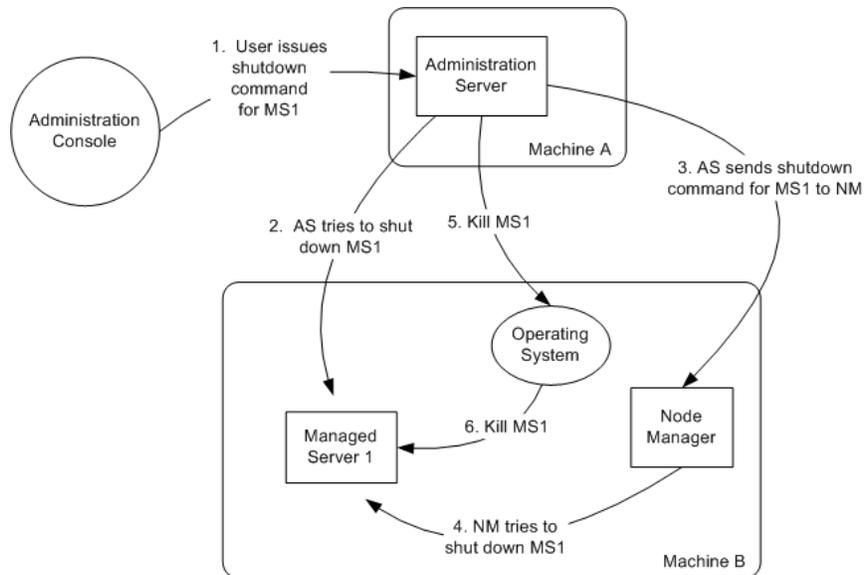
Note: Managed Server Independence mode is enabled by default.

How Node Manager Shuts Down a Server Instance

Figure 3-6 illustrates the communications involved in shutting down a Managed Server that is under Node Manager control. Depending on the state and availability of the Managed Server, Node Manager might need to try alternative strategies to successfully initiate the shutdown.

Node Manager is running on Machine B, which hosts Managed Server 1.

Figure 3-6 Shutting Down a Server Instance Under Node Manager Control



1. Through the Administration Console, an authorized user issues a shutdown command for Managed Server 1.

2. The Administration Server issues the shutdown command directly to Managed Server 1. If it successfully contacts Managed Server 1, Managed Server 1 performs the shutdown sequence described in “[Graceful Shutdown](#)” in *Managing Server Startup and Shutdown*.
3. If, in the previous step, the Administration Server failed to contact Managed Server 1, it issues a shutdown command for Managed Server 1 to Node Manager on Machine B.
4. Node Manager issues a request to the operating system to kill Managed Server 1.
5. The operating system ends the Managed Server 1 process.

Node Manager and System Crash Recovery

To ensure that Node Manager properly restarts servers after a system crash, you must perform the following:

- Ensure that `CrashRecoveryEnabled` is set to true.
The `CrashRecoveryEnabled` configuration property allows Node Manager to restart servers after a system crash. The property is not enabled by default.
- You should start the Administration Server via Node Manager.
- All managed servers should be started via the Administration Server. You can accomplish this via WLST or the Administration Console.

After the system is restarted, Node Manager checks each managed domain specified in the `nodemanager.domains` file to determine if there are any server instances that were not cleanly shutdown. This is determined by the presence of any lock files which are created by Node Manager when a WebLogic Server process is created. This lock file contains the process identifier for WebLogic Server startup script. If the lock file exists, but the process ID is not running, Node Manager will attempt to automatically restart the server.

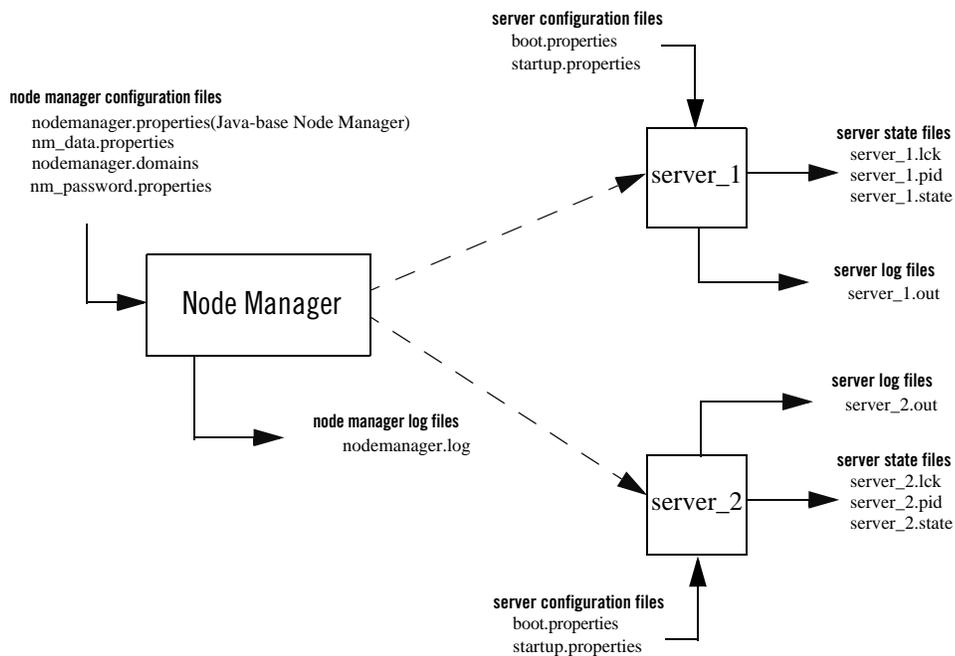
If the process is running, Node Manager performs an additional check to access the management servlet running in the process to verify that the process corresponding to the process ID is a WebLogic Server instance.

Note: When Node Manager performs a check to access the management servlet, an alert may appear in the server log regarding improper credentials.

Node Manager Configuration and Log Files

In managing multiple servers, Node Manager uses multiple configuration files and outputs log files to multiple directories, as shown in the following figure. For a description of these files, see “Node Manager Configuration and Log Files” on page 3-13.

Figure 3-7 Node Manager Configuration and Logging Environment



The following sections describe Node Manager configuration and log files:

- “Configuration Files” on page 3-13
- “Log Files” on page 3-16

Configuration Files

Except where noted, configuration files apply to both Java-based and script-based Node Manager.

nodemanager.properties

This is the configuration file used by the Java-based version of Node Manager. See [“Reviewing nodemanager.properties” on page 3-25](#).

This file is located in `WL_HOME/common/nodemanager`.

nodemanager.domains

This file contains mappings between the names of domains managed by Node Manager and their corresponding directories. See [“Configuring nodemanager.domains File” on page 3-19](#).

This file is located in `WL_HOME/common/nodemanager`.

nm_data.properties

This file stores the encryption data the Node Manager uses a symmetric encryption key. The data is stored in encrypted form.

This file is located in `WL_HOME/common/nodemanager`.

nm_password.properties

This file stores the Node Manager user name and password. See [“Specifying Node Manager Username and Password” on page 3-19](#).

This file is located in `DOMAIN_HOME/config/nodemanager`.

boot.properties

Node Manager uses this file to specify a boot identity when starting a server. See [“General Node Manager Configuration” on page 3-17](#).

This file is located in `domain-name/servers/server_name/data/nodemanager`.

startup.properties

Each Managed Server instance has its own `startup.properties` file with properties that control how Node Manager starts up and controls the server. Node Manager automatically creates this file by using properties passed to Node Manager when the Administrative Server was last used to start the server. This allows a Node Manager client or startup scripts to restart a Managed Server using the same properties last used by the Administrative Server.

For more information on startup.properties, see [“Setting Server Startup Properties” on page 3-21](#). These properties correspond to the server startup attributes contained in ServerStartMBean and the health monitoring attributes in ServerStartMBean.

This file is located in `domain-name/servers/server_name/data/nodemanager`.

server_name.addr

`server_name.addr` stores the IP address added when a server starts or is migrated. This file is generated after the server IP address is successfully brought online during migration. `server_name.addr` is deleted when the IP address is brought offline. The server IP address is used to validate remove requests to prevent addresses being erroneously removed while shutting down the server.

This file is located in `domain-name/servers/server_name/data/nodemanager`.

server_name.lck

`server_name.lck` is generated by each server and contains an internally used lock ID.

This file is located in `domain-name/servers/server_name/data/nodemanager`

server_name.pid

`server_name.pid` is generated by each server and contains the process ID of the server. Node Manager checks the process ID generated by the server during crash recovery.

This file is located in `domain-name/servers/server_name/data/nodemanager`

server_name.state

`server_name.state` is generated by the server and contains the server's current state. Node Manager monitors the contents of this file to determine the current state of the server.

Note: Do not delete or alter this file. Without this file Node Manager cannot determine the current state of the server.

This file is located in `domain-name/servers/server_name/data/nodemanager`.

Log Files

Use the Node Manager and WebLogic Server log files to help troubleshoot problems in starting or stopping individual Managed Servers.f

Table 3-1 Node Manager Log File Locations

Log File	Location
Node Manager Log File	NodeManagerHome/nodemanager.log
Node Manager Server Instance Log Files	domain-name/servers/<server-name>/logs/<server-name>.out
Web Logic Server Log Files	domain-name/servers/<server-name>/logs/<server_name>.log

nodemanager.log

Node Manager creates a log file located in `NodeManagerHome/nodemanager.log`. This log file stores data about all of the domains administered by Node Manager.

This log file is generated by Node Manager and contains data for all domains that are controlled by Node Manager on a given physical machine. See [“nodemanager.log” on page 3-16](#).

This file is located in `WL_HOME/common/nodemanager`.

Log output is appended to the current `nodemanager.log`. Log rotation is disabled by default, but can be enabled by setting `LogCount` in `nodemanager.properties`.

You can view the Node Manager log file by:

- Selecting Machines—>Monitoring—>Node Manager Log page in the Administration Console
- Using the WLST `nmLog` command

server_name.out

For each server instance that it controls, Node Manager maintains a log file that contains `stdout` and `stderr` messages generated by the server instance. If the remote start debug property is enabled as a remote start property for the server instance, or if the `NodeManager debug` property is enabled, Node Manager will include additional debug information in the server output log information.

Note: You cannot limit the size of the log files Node Manager creates. Logging to `stdout` is disabled by default.

This file is located in `domain_name/servers/<server_name>/logs`

Node Manager creates the server output log for a server instance in the server instance's `logs` directory, with the name:

`server-name.out`

where `server-name` is the name of the server instance.

You can view the Node Manager log file for a particular server instance by:

- Selecting Diagnostics —>Log Files.
- Using the WLST `nmServerLog` command.

There is no limit to the number of server output logs that Node Manager can create.

WebLogic Server Log Files

A server instance under Node Manager control has its own log file, in addition to the log file created by Node Manager.

You can view the regular log file for a server instance by selecting Diagnostics->Log Files, selecting the server log file, and clicking View.

General Node Manager Configuration

This section describes general Node Manager configuration that applies to the Java and script version of Node Manager. You should ensure that you have performed all of the items outlined in the following sections.

After you have performed general Node Manager configuration, you should perform the configuration procedures outlined in [“Configuring Java-based Node Manager” on page 3-23](#) or [“Configuring Script-based Node Manager” on page 3-36](#) depending on which version of Node Manager you are using.

Configuring Your Computer to Run Node Manager

Node Manager must run on each computer that hosts WebLogic Server instances that you want to control with Node Manager. Configure each computer as a Machine in WebLogic Server, and assign each server instance that you will control with Node Manager to the machine upon which it runs.

Ideally, Node Manager should run as an operating system service or daemon, so that it is automatically restarted in the event of system failure or reboot. For more information, see [“Installing the Node Manager as a Windows Service”](#) in the *Installation Guide*.

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.
- Node Manager monitors the Managed Servers that it has started.
- Automatic restart of Managed Servers is enabled. Node Manager restarts server instances that it killed or were killed by another method.

Controlling and Configuring Node Manager Using WLST

The WebLogic Scripting Tool (WLST) is a command-line scripting interface that system administrators and operators use to monitor and manage WebLogic Server instances and domains. You can start, stop, and restart server instances remotely or locally, using WLST as a Node Manager client. In addition, WLST can obtain server status and retrieve the contents of the server output log.

Using nmConnect() in a Production Environment

By default, the `nmConnect()` command cannot be used in a production environment. You must perform the following procedures to use `nmConnect` in a production environment.

1. Start the administration server.
2. Using the Administration Console update the Node Manager credentials from the Advanced options under *domain_name*—>Security—>General.
3. Start WLST in online mode.
4. Run `nmEnroll()` using the following as an example:

```
nmEnroll('C:/bea/user_projects/domains/prod_domain',  
        'C:/bea/wlserver_10.0/common/nodemanager')
```

Running `nmEnroll()` ensures that the correct NodeManager user and password token are supplied to each managed server. Once these are available for each managed server, you can use `nmConnect()` in a production environment.

Note: You must run `nmEnroll()` on each machine that is running a managed server. Additionally, you should run `nmEnroll()` for each domain directory on each machine.

Specifying Node Manager Username and Password

The `nm_password.properties` file contains the Node Manager username and password. These are used to authenticate connection between a client (for example, the Administration server) and Node Manager.

Note: This username and password are only used to authenticate connections between Node Manager and clients. They are independent from the server admin ID and password.

This file is created when you use `nmEnroll()` to copy the necessary configurations files from one machine to another when creating a domain. After `nm_password.properties` is created, you can change the values for the Node Manager password and properties using the Administration Console. Changes are propagated to the `nm_password.properties` file and are picked up by Node Manager.

Note: If you edit `nm_password.properties` manually, you must restart Node Manager in order for the changes to take effect.

The `nm_password.properties` file must exist on each physical machine that runs Node Manager. However, the Node Manager username and password do not have to be identical on every machine within your domain.

Configuring a Machine to Use Node Manager

A WebLogic Server Machine resource associates a particular machine with the server instances it hosts, and specifies the connection attributes for the Node Manager process on that system.

Configure a machine definition for each machine that runs a Node Manager process using the Environment—>Machines—><*machine_name*>—>Node Manager page in the Administration Console. Enter the DNS name or IP address upon which Node Manager listens in the Listen Address box.

Configuring nodemanager.domains File

The `nodemanager.domains` file specifies the domains that a Node Manager instance controls. Thus stand-alone clients do not need to specify the domain directory explicitly.

This file must contain an entry specifying the domain directory for each domain the Node Manager instance controls, in this form:

```
<domain-name>=<domain-directory>
```

When a user issues a command for a domain, Node Manager looks up the domain directory from `nodemanager.domains`.

This file provides additional security by restricting Node Manager client access to the domains listed in this file. The client can only execute commands for the domains listed in `nodemanager.domains`.

If you created your domain with the Configuration Wizard, the `nodemanager.domains` file was created automatically. If necessary, you can manually edit `nodemanager.domains` to add a domain.

Note: If you use the backslash character (`\`) in `nodemanager.domains`, you must escape it as `(\\)`.

Configuring Remote Startup Arguments

In the Server—>Configuration—>Server Start page for the Managed Server, specify the startup arguments that Node Manager will use to start a Managed Server. If you do not specify startup arguments for a Managed Server, Node Manager uses its own properties as defaults to start the Managed Server. For more information, see [Table 3-4, “Node Manager Properties,” on page 3-26](#). 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 instance.

If you will run Node Manager as a Windows Service, as described in “[Installing the Node Manager as a Windows Service](#)” in the *Installation Guide*, 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 Jrocket

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 shut down.
2. During the interval between the Managed Server being killed, and the Node Manager service being shut down, 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 Server with the `-xrs` or `-xnohup` option avoids this sequence of events by preventing the immediate shutdown of the Managed Server during machine shutdown.

Setting Server Startup Properties

You can use Node Manager to set the startup properties for a server. These properties can be defined in `startup.properties` or passed as an object using administrative utilities such as WLST. The methods of setting startup properties and their valid values are outlined in the sections below.

`startup.properties`

Node Manager uses the `startup.properties` file to determine the startup and configuration when starting a server. This file is defined for each server instance and is located in:

```
domain_home/servers/server_name/data/nodemanager/startup.properties
```

The contents of `startup.properties` are derived from the Server Mbean, or the Cluster Mbean if the server is part of a cluster. For more information, see the Mbean reference.

Setting Startup Properties Using Administration Utilities

When using the WLST `nmStart()` command, the server configuration can not be determined directly. Therefore, you must pass the server start properties as a WLST properties object to the `nmStart()` command.

Server Startup Properties

The following server startup properties can be passed to a server when started via Node Manager.

Table 3-2 Server Startup Properties

Property	Description
JavaHome	Defines the Java home directory used when starting the server.
Arguments	The arguments used when starting the server.
SSLArguments	These arguments are used when you have enabled the domain-wide administration port.

Table 3-2 Server Startup Properties

RestartMax	The number of times Node Manager can attempt to restart the server.
RestartDelaySeconds	The number of seconds Node Manager should wait before attempting to restart the server.
ClassPath	The classpath to use when starting a server.
BEAHome	The BEA home directory to use when starting a server.
AdminURL	The URL of the administration server. Note: This value should only be specified in the startup.properties file for a managed server.
AutoRestart	Specifies whether Node Manager can automatically restart this server if it fails.
AutoKillIfFailed	Specifies whether Node Manager should automatically kill the server if its health status is <code>failed</code> .
SecurityPolicyFile	Specifies the security policy file to use when starting this server.
ServerIP	The IP address of the server.

Ensuring 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 an Administration Server is not defined, when Node Manager starts a Managed Server it will direct the Managed Server to contact localhost for its configuration information.

Set the Listen Address using the Servers—>Configuration—>General page in the Administration Console.

Setting Node Manager Environment Variables

Node Manager requires you to set several environment variables before you start it.

You can set these variables manually on the command line or you can create a start script that sets them automatically. The sample start scripts provided with WebLogic Server — `startNodeManager.cmd` and `startNodeManager.sh` — set the required variables.

Table 3-3 Node Manager Environment Variables

Environment Variable	Description
JAVA_HOME	JDK root directory used by Node Manager. For example: <pre>set JAVA_HOME=c:\bea\jdk131</pre> Node Manager has the same JDK version requirements as WebLogic Server.
WL_HOME	WebLogic Server installation directory. For example: <pre>set WL_HOME=c:\bea\wlserver_10.0</pre>
PATH	Must include the WebLogic Server bin directory and path to your Java executable. For example: <pre>set PATH=%WL_HOME%\server\bin;%JAVA_HOME%\bin;%PATH%</pre>
LD_LIBRARY_PATH (UNIX only)	For HP UX and Solaris systems, you must include the path to the native Node Manager libraries. Solaris example: <pre>LD_LIBRARY_PATH:\$WL_HOME/server/lib/solaris:\$WL_HOME/server/lib/solaris/oci816_8</pre> HP UX example: <pre>SHLIB_PATH=\$SHLIB_PATH:\$WL_HOME/server/lib/hpux11:\$WL_HOME/server/lib/hpux11/oci816_8</pre>
CLASSPATH	You can set the Node Manager CLASSPATH either as an option on the <code>java</code> command line used to start Node Manager, or as an environment variable. Windows NT example: <pre>set CLASSPATH=.;%WL_HOME%\server\lib\weblogic_sp.jar;%WL_HOME%\server\lib\weblogic.jar</pre>

Configuring Java-based Node Manager

It is recommended that you configure Node Manager to run as an operating system service or a Windows service on Windows systems. By default, the operating system service starts up Node Manager to listen on `localhost:5556`.

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.

Depending on your platform, follow the instructions in [“Reconfigure Startup Service for Windows Installations”](#) or [“Configuring Java-based Node Manager Security”](#).

The following sections provide configuration information specific to Java-based Node Manager:

- [“Reconfigure Startup Service for Windows Installations”](#) on page 3-24
- [“Configuring Java-based Node Manager Security”](#) on page 3-24
- [“Configuring Java-based Node Manager Security”](#) on page 3-24
- [“Reviewing nodemanager.properties”](#) on page 3-25
- [“Configuring Node Manager to Use Start and Stop Scripts”](#) on page 3-33
- [“Deprecated Node Manager Properties”](#) on page 3-31

Reconfigure Startup Service for Windows Installations

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.

Configuring Java-based Node Manager Security

Node Manager security relies on a one-way SSL connection between the client and server.

If you are establishing a command line connection to the Java Node Manager using the WebLogic Server Scripting Tool (WLST) `nmConnect` command, you provide the Node Manager user name

and password. Node Manager verifies the username and password against the domain's `nm_password.properties` file. For more information on `nm_password.properties`, see [“Specifying Node Manager Username and Password” on page 3-19](#).

Node Manager credentials are located on the Security>General>Advanced Options Console page.

Administration Console users do not need to explicitly provide credentials to connect to Node Manager—the Node Manager user name and password are available in the domain configuration and are provided automatically.

Remote Server Start Security for Java-based Node Manager

A remote start user name and password is required to start a server instance with Node Manager. These credentials are provided differently for Administration Servers and Managed Servers.

- Credentials for Managed Servers—When you invoke Node Manager to start a Managed Server it obtains its remote start name and password from the Administration Server.
- Credentials for Administration Servers—When you invoke Node Manager to start an Administration Server, the remote start user name can be provided on the command line, or obtained from the Administration Server’s `boot.properties` file. The Configuration Wizard initializes the `boot.properties` file and the `startup.properties` file for an Administration Server when you create the domain.

Any server instance started by Node Manager encrypts and saves the credentials with which it started in a server-specific `boot.properties` file, for use in automatic restarts.

Reviewing `nodemanager.properties`

Node Manager properties define a variety of configuration settings for a Java-based 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`.

`nodemanager.properties` is created in the directory specified in `NodeManagerHome`. If `NodeManagerHome` is not defined, `nodemanager.properties` is created in the current directory.

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.

[Table 3-4](#) describes Node Manager properties.

In many environments, the SSL-related properties in `nodemanager.properties` may be the only Node Manager properties that you must explicitly define. 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 `StartScriptEnabled` 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`.

Table 3-4 Node Manager Properties

Node Manager Property	Description	Default
<code>LogFile</code> (New)	Location of the Node Manager log file.	<code>NodeManagerHome/nodemanager.log</code>
<code>LogLimit</code> (New)	Maximum size of the Node Manager Log specified as an integer. When this limit is reached, a new log file is started.	<code>unlimited</code>
<code>LogCount</code> (New)	Maximum number of log files to create when <code>LogLimit</code> is exceeded.	<code>1</code>
<code>LogAppend</code> (New)	If set to <code>true</code> , then a new log file is not created when the Node Manager restarts; the existing log is appended instead.	<code>true</code>
<code>LogToStderr</code> (New)	If set to <code>true</code> , the log output is also sent to the standard error output.	<code>false</code>
<code>LogLevel</code> (New)	Severity level of logging used for the Node Manager log. Node Manager uses the same logging levels as WebLogic server.	<code>INFO</code>
<code>LogFormatter</code> (New)	Name of formatter class to use for NM log messages.	<code>weblogic.nodemanager.server.LogFormatter</code>
<code>CrashRecoveryEnabled</code> (New)	Enables system crash recovery.	<code>false</code>
<code>SecureListener</code> (New)	If set to <code>true</code> , use the SSL listener, otherwise use the plain socket	<code>true</code>

Table 3-4 Node Manager Properties

Node Manager Property	Description	Default
<code>CipherSuite</code> (New)	The name of the cipher suite to use with the SSL listener.	<code>TLS_RSA_EXPORT_WITH_RC4_40_MD5</code>
<code>StartScriptEnabled</code> (New)	If true, use the start script specified by <code>StartScriptName</code> to start a server. For more information, see “Configuring Node Manager to Use Start and Stop Scripts.”	<code>false</code>
<code>StartScriptName</code> (New)	The name of the start script, located in the domain directory	<code>startWebLogic.sh</code> (UNIX) or <code>startWebLogic.cmd</code> (Windows)
<code>StopScriptEnabled</code> (New)	If true, execute the stop script specified by <code>StopScriptName</code> after the server has shutdown. For more information, see “Configuring Node Manager to Use Start and Stop Scripts.”	<code>false</code>
<code>StopScriptName</code> (New)	The name of the script to be executed after server shutdown.	<code>none</code>
<code>DomainsFile</code> (New)	The name of the <code>nodemanager.domains</code> file	<code>NodeManagerHome/nodemanager.domains</code>
<code>DomainsFileEnabled</code> (New)	If set to true, use the file specified in <code>DomainsFile</code> . If false, assumes the domain of the current directory or of <code>WL_HOME</code> .	<code>true</code>
<code>StateCheckInterval</code>	Specifies the interval Node Manager waits to perform a check of the server state.	<code>500 milliseconds</code>
<code>CustomIdentityAliases</code>	Specifies the alias when loading the private key into the keystore. This property is required when the <code>Keystores</code> property is set as <code>CustomIdentityandCustomTrust</code> or <code>CustomIdentityAndJavaStandardTrust</code> .	<code>none</code>

Table 3-4 Node Manager Properties

Node Manager Property	Description	Default
CustomIdentityKeyStoreFileName	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 CustomIdentity and CustomTrust or CustomIdentityAndJavaStandardTrust.	none
CustomIdentityKeyStorePassPhrase	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
CustomIdentityKeyStoreType	Specifies the type of the Identity keystore. Generally, this is JKS. This property is optional.	default keystore type from java.security
CustomIdentityPrivateKeyPassPhrase	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 CustomIdentityandCustomTrust or CustomIdentityAndJavaStandardTrust.	none
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

Table 3-4 Node Manager Properties

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 CustomIdentityandJavaStandardTrust 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: <ul style="list-style-type: none"> 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) CustomIdentityAndJavaStandardTrust 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

Table 3-4 Node Manager Properties

Node Manager Property	Description	Default
<code>ListenAddress</code>	Any address upon which the machine running Node Manager can listen for connection requests. This argument deprecates <code>weblogic.nodemanager.listenAddress</code> .	<code>null</code> With this setting, Node Manager will listen on any IP address on the machine
<code>ListenPort</code>	The TCP port number on which Node Manager listens for connection requests. This argument deprecates <code>weblogic.nodemanager.listenPort</code> .	<code>5556</code>
<code>NativeVersionEnabled</code>	A value of <code>true</code> causes native libraries for the operating system to be used. For UNIX systems other than Solaris, HP-UX, or Linux, set this property to <code>false</code> to run Node Manager in non-native mode. This will cause Node Manager to use the start script specified by the <code>StartScriptEnabled</code> property to start Managed Servers.	<code>true</code>
<code>NodeManagerHome</code>	Node Manager root directory which contains the following configuration and log files: <ul style="list-style-type: none"> <code>nm_data.properties</code> <code>nodemanager.domains</code> <code>nodemanager.log</code> <code>nodemanager.properties</code> For more information on these files, see “Node Manager Configuration and Log Files.” Note: By default, <code>NodeManagerHome</code> is <code>WL_HOME/common/nodemanager</code> . In a production environment, you may want to customize the location of the Node Manager root directory.	<code>NodeManagerHome</code>

Table 3-4 Node Manager Properties

Node Manager Property	Description	Default
weblogicHome	Root directory of the WebLogic Server installation. This is used as the default value of <code>-Dweblogic.RootDirectory</code> 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
keyFile	The path to the private key file to use for SSL communication with the Administration Server. Note: This property is used only in the process of upgrading from WebLogic Server, Version 7.x to Version 9.x.	none
keyPassword	The password used to access the encrypted private key in the key file. Note: This property is used only in the process of upgrading from WebLogic Server, Version 7.x to Version 9.x.	none
certificateFile	Specifies the path to the certificate file used for SSL authentication. Note: This property is used only in the process of upgrading from WebLogic Server, Version 7.x to Version 9.x.	none

Deprecated Node Manager Properties

This section lists the Node Manager properties that are deprecated in WebLogic Server 9.x.

Note: These properties are published for backwards compatibility and should not be used. SSL configurations will continue to work when migrating to WebLogic Server 9.x. However, the trusted key store is not used when running Node Manager.

Table 3-5 Depreciated Node Manager Properties

Node Manager Property	Description	Reason Depreciated
CustomTrustKeyPass Phrase (Deprecated)	The password used to access the encrypted private key in the key file.	Using 1-way SSL, Node Manager does not need access to a trusted key store.
CustomTrustKeyStore FileName (Deprecated)	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 CustomIdentityandCustomTrust.	Using 1-way SSL, Node Manager does not need access to a trusted key store.
CustomTrustKeyStore PassPhrase (Deprecated)	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.	Using 1-way SSL, Node Manager does not need access to a trusted key store.

Table 3-5 Deprecated Node Manager Properties

Node Manager Property	Description	Reason Deprecated
CustomTrustKeyStore Type (Deprecated)	Specifies the type of the Trust keystore. Generally, this is JKS. This property is optional.	Using 1-way SSL, Node Manager does not need access to a trusted key store.
JavaStandardTrustKey StorePassPhrase (Deprecated)	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 CustomIdentityandJavaStandardTrust or DemoIdentityAndDemoTrust.	Using 1-way SSL, Node Manager does not need access to a trusted key store.

Configuring Node Manager to Use Start and Stop Scripts

You can configure Node Manager to use a script to start a managed server or to execute a script after server shutdown has completed. These scripts can be used to perform tasks that need to be performed before a server is started or after it is shutdown. Mounting and unmounting remote disks is one example of a task that can be performed using scripts.

Note: Node Manager uses startup scripts to perform any required configuration, then start the server. In contrast, stop scripts are executed after the server has shutdown.

Using Start Scripts

You can use a start script allows you to specify required startup properties and perform any other work you need performed at start up. To define a start script:

Using Node Manager to Control Servers

1. In the `nodemanager.properties` file, set the `StartScriptEnabled` property to `true`. (The default is `false`.) If your start script is named `startWebLogic.sh` or `startWebLogic.cmd`, Node Manager uses one of those scripts as the default.
2. If you want to specify a custom start script, set the `StartScriptName` property to the name of your script in the `nodemanager.properties` file

Using Stop Scripts

You can use a stop script to perform any tasks that are required after the server has shutdown. To define a stop script:

1. In the `nodemanager.properties` file, set the `StopScriptEnabled` property to `true`.
2. Set the `StopScriptName` property to the name of your script in the `nodemanager.properties` file.

The following example shows a stop script that can be used to unmount a disk on UNIX systems:

```
#!/bin/sh
FS=/cluster/d2
if grep $FS /etc/mnttab > /dev/null 2>&1 ; then
sync
    PIDS=`/usr/local/bin/lsof $FS | awk
    '{if ($2 ~/[0-9]+)/} { print $2} '` | sort -u`
kill -9 $PIDS
sleep 1
sync
    /usr/sbin/umount -f $FS
fi
```

Using SSL With Java-based Node Manager

Administration Servers and Managed Servers communicate with Java-based Node Manager using one-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 `WL_HOME/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*.

Configuring Node Manager on Multiple Machines

If you have a domain that has managed servers on multiple physical machines, you must ensure that Node Manager is installed and configured on each machine. You can use the WLST command `nmEnroll` to copy all of the required domain and configuration information from one machine to another. For more information, see [“Controlling and Configuring Node Manager Using WLST”](#) on page 3-18 and [“nmEnroll\(\)”](#) in *WebLogic Scripting Tool*.

Configuring Node Manager as an inetd Service

When configuring Node Manager to run as an `inetd` or `xinetd` service, the following considerations apply:

- Ensure that `NodeManagerHome` and other system properties are defined.
- If `xinetd` is configured with `libwrap`, you should add the `NOLIBWRAP` flag.
- Ensure that the `hosts.deny` and `hosts.allow` files are configured correctly.
- Depending on your network environment, additional configuration may be necessary.

The following example shows how Node Manager can be configured within `xinetd`:

```
# default: off
# description: nodemanager as a service
service nodemgrsvc
{
    type                = UNLISTED
    disable             = no
    socket_type        = stream
    protocol           = tcp
    wait               = yes
    user               = <username>
    port               = 5556
    flags              = NOLIBWRAP
    log_on_success     += DURATION HOST USERID
```

```
server          = <path-to-jave>/java
env             = CLASSPATH=<cp> LD_LIBRARY_PATH=<ldpath>
server_args     = -client -DNodeManagerHome=<NMHome> <java options>
               <nodemanager options> weblogic.NodeManager -v
}
```

Configuring Script-based Node Manager

The SSH Node Manager is a shell script, `wlscontrol.sh`, located in `{WL_HOME}/common/bin/.wlscontrol.sh` must exist on each machine that hosts server instances that you want to control with Node Manager. This script can be customized to meet site-specific requirements.

You must have an SSH client executable on each machine where Node Manager or a Node Manager client runs. This script must also be in the path of the user id running it. Typically, an SSH client is a standard part of a Unix or Linux installation.

The following sections describe how to configure script-based Node Manager:

- [“Using SSL With Java-based Node Manager” on page 3-34](#)
- [“Creating a Node Manager User” on page 3-36](#)
- [“Configuring Script-based Node Manager Security” on page 3-37](#)

Creating a Node Manager User

Before running Node Manager, you should create a dedicated UNIX user account for performing Node Manager functions. This user should be added to all machines that will host the SSH Node Manager and to all machines that will host a Node Manager client, including the Administration Server.

Overriding the Default SSH Port

The default SSH port used by Node Manager is 22. You can override that setting in the following ways:

- Set the `Port=` parameter in the `~/.ssh/config` file to set the default port for an individual user.
- Set the `Port=` parameter in the `/etc/ssh_config` file to set the default port across the entire system.

- Start the Administration Server using the following system property:

```
-Dweblogic.nodemanager.ShellCommand="ssh -o PasswordAuthentication=no -p
%P %H wlscontrol.sh -d %D -r %R -s %S %C"
```

After starting the server, you can edit the SSH port in the Administration Server's configuration file.

Configuring Script-based Node Manager Security

The Node Manager SSH shell script relies on SSH user-based security to provide a secure trust relationship between users on different machines. Authentication is not required. You create a UNIX user account—typically one per domain—for running Node Manager commands and scripts. A user logged in as this user can issue Node Manager commands without providing a username and password.

Note: You must also ensure that the Node Manager and WebLogic Server commands are available in the path of the UNIX user ID used to run them.

Security for WebLogic Server Scripts

To perform Server migration and other tasks, the user ID executing scripts such as `wlscontrol.sh` must have enough sufficient security permissions. This includes being able to bring an IP address online or take an IP address offline via a network interface.

Server migration is performed by the cluster master when it detects that a server has failed. It then uses SSH to launch a script on the target machine to begin the migration. The script on the target machine runs as the same user ID running the server on the cluster master.

The commands required to perform server migration are `ifconfig` and `arping`. Since these scripts require elevated OS privileges, it is important to note that this can prevent a potential security hole.

Using `sudo`, you can configure your SSH to only allow `ifconfig` and `arping` to be run using elevated privileges.

Remote Server Start Security for Script-based Node Manager

A remote start user name and password is required to start a server instance with Node Manager. These credentials are provided differently for Administration Servers and Managed Servers.

- Credentials for Managed Servers—When you invoke Node Manager to start a Managed Server it obtains its remote start name and password from the Administration Server.

- **Credentials for Administration Servers**—When you invoke Node Manager to start an Administration Server, the remote start user name can be provided on the command line, or obtained from the Administration Server's `boot.properties` file. The Configuration Wizard initializes the `boot.properties` file and the `startup.properties` file for an Administration Server when you create the domain.

Any server instance started by Node Manager encrypts and saves the credentials with which it started in a server-specific `boot.properties` file, for use in automatic restarts.

Generating and Distributing Key Value Pairs

The script-based Node Manager uses two types of key value pairs. This section contains instructions for distributing key value pairs to the machines that will host a Node Manager client or server.

- [“Shared Key Value Pair” on page 3-38.](#)
- [“Individual Key Value Pairs” on page 3-39](#)

Shared Key Value Pair

This option distributes the same key value pair to all machines that will host a Node Manager client or server.

The simplest way to accomplish this is to set up your LAN to mount the Node Manager user home directory on each of the machines. This makes the key value pair available to the machines.

Otherwise

1. Generate an RSA key value pair for the user with the `ssh-keygen` command provided with your SSH installation.

The default location for the private and public keys are `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub` respectively.

If these keys are stored in a different location, modify the `ShellCommand` template, adding an option to the `ssh` command to specify the location of the keys.

2. Append the public key to the `~/.ssh/authorized_keys` file on the Node Manager machine. For example:

```
command="/home/bea/server90/common/nodemanager/nodemanager.sh" 1024 33
23...2323
```

in which the you substitute the public key that you generated, as stored in `id_rsa.pub`, for the string shown in the example as

```
1024 33 23...2323
```

Note: The prefix `command=<command>` ensures that a user that establishes a session with the machine using the public key can only run the command specified—`nodemanager.sh`. This ensures that the user can only perform Node Manager functions, and prevents unauthorized access to data, system utilities, or other resources on the machine.

3. Manually distribute the key value pair to each machine that will host a Node Manager server instance or client.
4. Execute the following command on the client machine to check that the Node Manager client can access the Node Manager:

```
/home/bea$ ssh montgomery VERSION
```

This response indicates that the client accessed Node Manager successfully:

```
+OK NodeManager v9.1.0
```

Individual Key Value Pairs

On each machine that will host a Node Manager client:

1. Generate a separate RSA key value pair for the Node Manager user as described in step one in the previous section.
2. Append the public key to the machine's `~/.ssh/authorized_keys` file user as described in step two in the previous section.

Starting Node Manager and Servers

The following sections provide information on how to start and run Java-based and script-based Node Manager. It also provides information on the recommended procedures for starting servers using Node Manager.

Running Node Manager as a Startup Service

It is recommended that you install Node Manager to run as a startup service. This allows Node Manager to start up automatically each time the system is restarted.

By default, Node Manager listens only from the local host. If you want 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.

Starting Java-based Node Manager Using 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 [“Setting Node Manager Environment Variables” on page 3-22](#).

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 Java-based Node Manager

The syntax for starting Java-based Node Manager is:

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

where:

- `java_option` is a direct argument to the `java` executable, such as `-ms` or `-mx`.
Note: If you did not set the `CLASSPATH` environment variable, use the `-classpath` option to identify required Node Manager classes.
- `nodemanager_property` is a Node Manager property. Instead of supplying Node Manager property values on the command line, you can edit the `nodemanager.properties` file, which is installed in the directory where you start Node Manager. For more information, see [Table 3-4, “Node Manager Properties,” on page 3-26](#).

Node Manager property values you supply on the command line override the values in `nodemanager.properties`.

- `server_property` is a server-level property that Node Manager accepts on the command line, including:
 - `bea.home`—the BEA home directory that server instances on the current machine use.

- `java.security.policy`— path to the security policy file that server instances on the current machine use.

Notes: For UNIX systems:

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"
```

For UNIX systems other than Solaris, HP-UX, and Linux operating systems, you must disable the `weblogic.nodemanager.nativeVersionEnabled` option at the command line when starting Node Manager (or set the property in `nodemanager.properties`) to use the pure Java version. For more information, see [“Reviewing nodemanager.properties” on page 3-25](#).

Running Script-based Node Manager

To use the SSH Node Manager Command Shell, start the Administration Server using the following command line option:

```
-Dweblogic.nodemanager.ShellCommand='ssh -o PasswordAuthentication=no %H wlscontrol.sh -d %D -r %R -s %S %C'
```

The `weblogic.nodemanager.ShellCommand` attribute specifies the command template to use to communicate with a remote SSH Node Manager and execute Node Manager functions for server instances under its control.

The template assumes that `wlscontrol.sh` is in the default path on the remote machine hosting Node Manager.

The `ShellCommand` syntax is:

```
ssh -o PasswordAuthentication=no %H wlscontrol.sh -d %D -r %R -s %S %C'
```

The possible command line options are listed in [Table](#) . The possible parameter values are listed in [Table](#) .

For example, if you type this command,

```
ssh -o PasswordAuthentication=no wlscontrol.sh myserver start
```

The listen address and port of the SSH server default to the listen address and port used by Node Manager on the remote machine. The domain name and domain directory are assumed to be the root directory specified for the target server instance, `myserver`.

Using Node Manager to Control Servers

This command:

```
ssh -o PasswordAuthentication=no 172.11.111.11 wlscontrol.sh -d  
ProductionDomain -r ProductionDomain -s ServerA'
```

issues a `START` command to the server instance named `ServerA`, in the domain called `ProductionDomain`, located in the `domains/ProductionDomain` directory.

The `ssh` command must include the string:

```
-o PasswordAuthentication=no
```

This string passes the `ssh PasswordAuthentication` option. A value of `yes` causes the client to hang when it tries to read from the console.

Table 3-6 wlscontrol.sh Command Line Options

Parameter	Description
-n	Specifies the Node Manager root directory
-s	Specifies the server name
-r	Specifies the domain directory
-x	Sets the Node Manager debug flag.
-c	Enables a server start script.
-f	The name of the server start script.
-p	The name of the server stop script.
-h	Prints the usage for <code>wlscontrol.sh</code> .

Table 3-7 Shell Command Templates

Parameter	Description	Default
%H	Host name of the SSH server	<code>NodeManagerMBean.ListenAddress</code>
%P	Port number of SSH server	<code>NodeManagerMBean.ListenAddress</code> 22
%S	WebLogic server name	none

Table 3-7 Shell Command Templates

%D	WebLogic domain name	ServerStartMBean.RootDirectory
%R	Domain directory (server root)	ServerStartMBean.RootDirectory
%C	Node manager script command	none
	<ul style="list-style-type: none">• <code>START</code>—Start server• <code>KILL</code>—Kill server• <code>STAT</code>—Get server status• <code>GETLOG</code>—Retrieve server output log.• <code>VERSION</code>—Return Node Manager version.	

Stopping Node Manager

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

Using Node Manager to Control Servers

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” on page 4-2](#)
- [“Verifying the Setup” on page 4-16](#)
- [“Using the Control Panel to Stop or Restart a Server Instance” on page 4-17](#)
- [“Removing a Server as a Windows Service” on page 4-18](#)
- [“Changing Startup Credentials for a Server Set Up as a Windows Service” on page 4-19](#)

Setting Up a Windows Service: Main Steps

To set up a Windows service:

1. 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 4-2](#).
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 4-5](#).
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 4-6](#).
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 4-8](#).
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 4-11](#).
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 4-15](#).

Creating a Server-Specific Script

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 script 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 4-1, “Example Script for Setting Up a Server as a Windows Service,”](#) on page 4-5.

To create a server-specific script:

1. In the root directory for the domain’s Administration Server 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).

```
– 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

Setting Up a WebLogic Server Instance as a Windows Service

without needing to modify the Windows service. For more information, see [“Boot Identity Files” on page 2-10](#).

```
– 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, see [Creating a New WebLogic Domain](#) in *Creating WebLogic 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=[-XmsNumberm] [-XmxNumberm]
```

where *Number* is a numerical value in megabytes (MB). The `-XmsNumberm` argument establishes a minimum heap size for the JVM and the `-XmxNumberm` 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.

The Administration Server (which is not a service) must be started before installing and starting Managed Server as a Windows service.

To configure a connection to the Administration Server:

1. In a text editor, open the server-specific script.
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, see [“Configuring Managed Server Connections to the Administration Server” on page 2-16](#)

For an example, refer to the bold text in [Listing 4-1](#).

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

Listing 4-1 Example Script for Setting Up a Server as a Windows Service

```
echo off
SETLOCAL
```

Setting Up a WebLogic Server Instance as a Windows Service

```
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 4-18](#).
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 4-9](#).
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 4-2](#) 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 4-2](#) 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 `ServerRuntimeMBean`, refer to the WebLogic Server [Javadoc](#).

Listing 4-2 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:

Setting Up a WebLogic Server Instance as a Windows Service

```
"%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 4-12](#).

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.

Setting Up a WebLogic Server Instance as a Windows Service

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:

- Place the value of the `set CLASSPATH` command in a separate text file.
- In the `WL_HOME\server\bin\installSvc.cmd` master script, find the `set CMDLINE` command.
- 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"
```

- 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 privileges. For more information, see [“Verifying the User Account Under Which the Service Runs” on page 4-16](#).

To run the server-specific script:

Setting Up a WebLogic Server Instance as a Windows Service

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 4-8.

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.

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 script 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 4-3, "Script to Remove a Windows Service,"](#) on page 4-19.

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 4-3 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

Setting Up a WebLogic Server Instance as a Windows Service

Server default security realm. For more information, see [“Boot Identity Files” on page 2-10](#).

- 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 4-18](#).
 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.

Avoiding and Recovering From Server Failure

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.

For high availability requirements, 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.

The following sections provide information and procedures for recovering failed server instances:

- [“Failure Prevention and Recovery Features”](#) on page 5-2
- [“Directory and File Backups for Failure Recovery”](#) on page 5-4
- [“WebLogic Server Exit Codes and Restarting After Failure”](#) on page 5-6
- [“Restarting a Failed Administration Server”](#) on page 5-6
- [“Restarting a Failed Managed Server”](#) on page 5-9
- [“Additional Failure Topics”](#) on page 5-12

Failure Prevention and Recovery Features

WebLogic Server offers several features that facilitate recovery from and protection against server failure.

Overload Protection

WebLogic Server detects increases in system load that can affect application performance and stability, and allows administrators to configure failure prevention actions that occur automatically at predefined load thresholds.

Overload protection helps you avoid failures that result from unanticipated levels of application traffic or resource utilization.

WebLogic Server attempts to avoid failure when certain conditions occur:

- Workload manager capacity is exceeded
- HTTP session count increases to a predefined threshold value
- Impending out of memory conditions

Failover for Clustered Services

You can increase the reliability and availability of your applications by hosting them on a WebLogic Server cluster. Clusterable services, such as EJBs and Web applications, can be deployed uniformly—on each Managed Server—in a cluster, so that if the server instance upon which a service is deployed fails, the service can fail over to another server in the cluster, without interruption in service or loss of state.

For more information, see [“Failover and Replication in a Cluster”](#) in *Using WebLogic Server Clusters*.

Automatic Restart for Failed Server Instances

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.

Using 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.

For more information, see [“Using Node Manager to Control Servers” on page 3-1](#).

Server-Level Migration

WebLogic Server provides the capability to migrate clustered server instances. A clustered server that is configured to be migratable can be moved in its entirety from one machine to another, at the command of an administrator, or automatically, in the event of failure. The migration process makes all of the services running on the server instance available on a different machine, but not the state information for the singleton services that were running at the time of failure. For more information, see [“Whole Server Migration”](#) in *Using WebLogic Server Clusters*.

Service-Level Migration

WebLogic Server supports migration of a individual singleton service as well as the server-level migration capability described in the previous section. Singleton services are services that run in a cluster but must run on only a single instance at any given time, such as JMS and the JTA transaction recovery system.

An administrator can migrate a JMS server or the JTS transaction recovery from one server instance to another in a cluster, either in response to a server failure or as part of regularly-scheduled maintenance. This capability improves the availability of pinned services in a cluster, because those services can be quickly restarted on a redundant server should the host server fail.

For more information, see [“Service Migration”](#) in *Using WebLogic Server Clusters*.

Managed Server Independence Mode

Managed Servers maintain a local copy of the domain configuration. When a Managed Server starts, it contacts its Administration Server to retrieve any changes to the domain configuration that were made since the Managed Server was last shut down. If a Managed Server cannot connect to the Administration Server during startup, it can use its locally cached configuration

information—this is the configuration that was current at the time of the Managed Server’s most recent shutdown. A Managed Server that starts up without contacting its Administration Server to check for configuration updates 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*.

Directory and File Backups for Failure Recovery

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.

Recovery from the failure of a server instance requires access to the domain’s configuration and security data. The WebLogic Security service stores its configuration data in the `config.xml` file, and also in an LDAP repository and other files.

For more information, see “[Domain Configuration Files](#)” and in *Understanding Domain Configuration*.

Back Up Domain Configuration Directory

By default, an Administration Server stores a domain’s configuration data in the `domain_name\config` directory, where `domain_name` is the root directory of the domain.

Back up the `config` directory 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.

Each time a Managed Server starts up, it contacts the Administration Server and if there are changes in to the domain configuration, the Managed Server updates its local copy of the domain `config` directory.

During operation, if changes are made to the domain configuration, the Administration Server notifies the Managed Servers which update their local `/config` directory. So, each Managed Server always has an current copy of its configuration data cached locally.

Back Up 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\servers\adminServer\data\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 Server 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, because 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 Help*.

Back Up SerializedSystemIni.dat and Security Certificates

Each server instance creates a file named `SerializedSystemIni.dat` and locates it in the `/security` 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.

WebLogic Server Exit Codes and Restarting After Failure

When a server instance stops, it issues an exit code. The value of the exit code provides information about the conditions under which the server process ended. When a server instance under Node Manager control exits, Node Manager uses the exit code to determine whether or not to restart the server instance. The server exit code can be used by other high-availability agents or scripts to determine what, if any action, to take after a server instance exits. Server exit codes are defined in the following table:

Table 5-1 WebLogic Server Exit Codes

Exit Code Value	Meaning	Restart Recommendation
Less than 0	A negative value indicates that the server instance failed during a state transition, and did not terminate in a stable condition. Example: If a Start in Standby command is issued for a server instance whose configuration is invalid, the server instance fails in the transitional <code>STARTING</code> state, and does not achieve the <code>STANDBY</code> state.	Do not attempt to restart the server. Diagnose the problem that caused the server process to exit.
0	Indicates that the server process terminated normally, as a result of a shutdown command, either graceful or forced.	None.
Greater than 0	A positive value indicates that the server instance stopped itself after determining that one or more of its subsystems were unstable. Example: A server instance detects an out of memory condition or stuck threads, and shuts itself down.	The server instance can be restarted.

Restarting a Failed Administration Server

The following sections describe how to start an Administration Server after a failure.

Note: You can use Node Manager to automatically restart a failed Administration Server. For more information see [“Using Node Manager to Control Servers” on page 3-1](#).

Restarting an Administration Server

See [“Starting and Stopping Servers” on page 2-1](#).

Restarting an Administration Server on the Same Machine

Table 5-2 Administration Server Restart Scenarios

Listen Address Definition	Administrations Server Restart Scenario	
	Same Machine	Different Machine
Not defined	<ol style="list-style-type: none"> 1. Start the Administration Server. Running MSs will reconnect automatically at the next <code>ReconnectIntervalSecs</code> To start an MS that was not running when AS failed, no change in command is required. 	<ol style="list-style-type: none"> 1. Install WLS. 2. Move data. 3. Start the Admin Server. Running MSs will learn the new AS address when they are contacted by the AS after it has been started. To start an MS that was not running when AS failed, supply the new AS Listen Address on command line.

Table 5-2 Administration Server Restart Scenarios

Listen Address Definition	Administrations Server Restart Scenario	
	Same Machine	Different Machine
DNS name or IP address of the host	<ol style="list-style-type: none"> 1. Start the Admin Server. <p>Running MSs will reconnect automatically at the next <code>ReconnectIntervalSecs</code></p> <p>To start an MS that was not running when AS failed, no change in command is required</p>	<ol style="list-style-type: none"> 1. Install WLS. 2. Move data. 3. Move IP address. <p>Running MSs will reconnect automatically at next <code>ReconnectIntervalSecs</code></p> <p>To start an MS that was not running when AS failed, no change in command is required</p> <ol style="list-style-type: none"> 4. If you do not move the IP address <p>Running MSs will learn the new AS address when they are contacted by the AS after it has been started.</p> <p>To start an MS that was not running when AS failed, you must supply the new Listen Address on the command line.</p>
DNS name mapped to multiple hosts	<ol style="list-style-type: none"> 1. Start the Admin Server. <p>Running MSs will reconnect automatically at the next <code>ReconnectIntervalSecs</code></p> <p>To start an MS that was not running when AS failed, no change in command is required</p>	<ol style="list-style-type: none"> 1. Install WLS. 2. Move data. <p>Running MSs will reconnect automatically at next <code>ReconnectIntervalSecs</code></p> <p>To start an MS that was not running when AS failed, no change in command is required.</p>

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 [“Directory and File Backups for Failure Recovery”](#) on page 5-4.
4. Restart the Administration Server on the new machine.

Managed Servers and Re-started Administration Server

If an Administration Server stops running while the Managed Servers in the domain continue to run, each Managed Server periodically attempts to reconnect to the Administration Server, at the interval specified by the `ServerMBean` attribute `AdminReconnectIntervalSeconds`. By default, `AdminReconnectIntervalSeconds` is ten seconds.

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 a Failed Managed Server

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 5-12.

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 [“Start, Shut Down, Suspend, and Restart Managed Servers”](#) on page 3-4.
- Start it manually with a command or script—For instructions, see [“Starting and Stopping Servers”](#) on page 2-1.

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 its locally cached configuration data from the `config` directory. A Managed Server that starts in this way is running in Managed Server Independence (MSI) mode.

Understanding 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 local `config` directory for `config.xml`—a replica of the domain’s `config.xml`.
- looks in its `security` directory for `SerializedSystemIni.dat` and for `boot.properties`, which contains an encrypted version of your username and password. For more information, see [“Boot Identity Files”](#) on page 2-10.

If `config.xml` and `SerializedSystemIni.dat` are not in these locations in the server’s domain directory, you can copy them from the Administration Server’s domain directory.

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 a Managed Server’s host machine to start the Managed Server.

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 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 continues to write messages to its local server log file but does not forward messages to the domain log file.

For more information, see [“How a Server Instance Forwards Messages to the Domain Log”](#) in *Configuring Log Files and Filtering Log Messages*.

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.

Starting a Managed Server in MSI Mode

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 Managed Server’s root directory contains the `config` subdirectory.

If the `config` directory does not exist, copy it from the Administration Server’s root directory or from a backup to the Managed Server’s root directory.

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 a Failed Administration Server”](#) on page 5-6.

Additional Failure Topics

For information related to recovering JMS data from a failed server instance, see [“Configuring Clustered WebLogic JMS Resources”](#) in *Programming WebLogic JMS*.

For information about transaction recovery after failure, see [“Transaction Recovery After a Server Fails”](#) in *Programming WebLogic JTA*.

Understanding Server Life Cycle

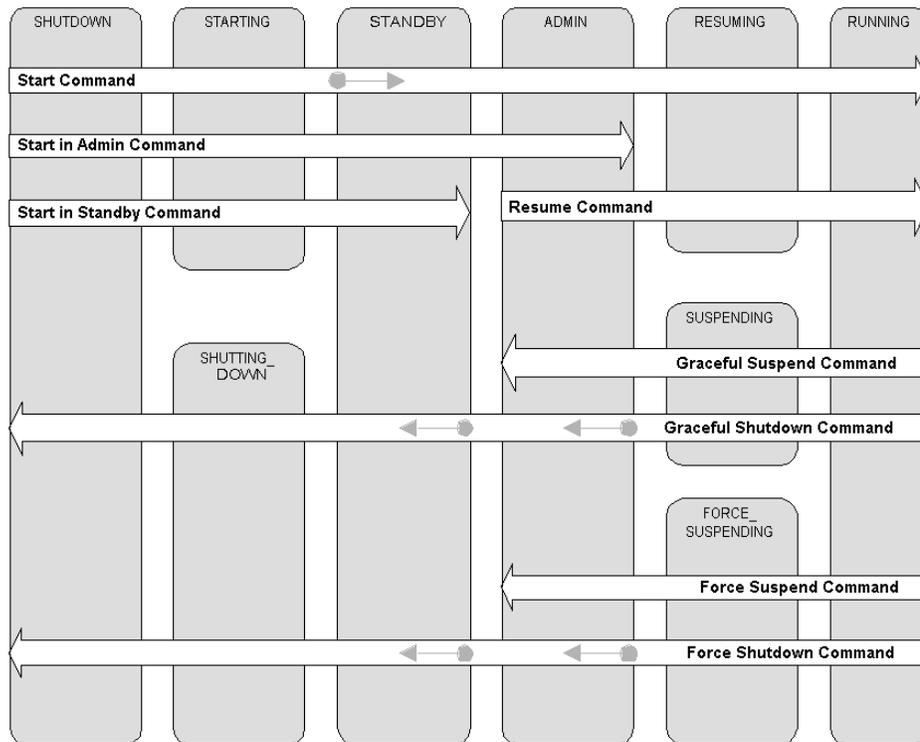
The series of states through which a WebLogic Server instance can transition is called the *server life cycle*. At any time, a WebLogic Server instance is in a particular operating state. Commands—such as start, stop, and suspend—cause specific changes to the operational state of a server instance. The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

- [“Diagram of the Server Life Cycle” on page 6-1](#)
- [“Getting and Using Server State” on page 6-2](#)
- [“Understanding Server States in the Server Life Cycle” on page 6-3](#)
- [“Using Server Life Cycle Commands” on page 6-12](#)
- [“Processing In-Flight Work During Suspend and Shutdown” on page 6-17](#)

Diagram of the Server Life Cycle

[Figure 6-1](#) illustrates the server life cycle and the relationships between states and life cycle commands.

Figure 6-1 State Transitions for Server Life Cycle Commands



To understand each state and the relationships among states, see [“Understanding Server States in the Server Life Cycle”](#) on page 6-3. For information on life cycle commands, see [“Using Server Life Cycle Commands”](#) on page 6-12.

Getting and Using Server State

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 corrective actions, such as migration of services, when a server instance fails or crashes

Get server state as follows:

- Administration Console—Multiple pages display state information:
 - On the Summary of Servers page (Environment→Servers), the Servers table displays the current state of each server instance in the current domain.
 - The `SERVER_NAME`→Monitoring page displays the state of the currently running server instance, and the date and time it entered the state.
 - Diagnostics→Log Files, includes timestamped messages for state transitions that have occurred since the server instance was last started.
- Programmatically—Use the `getState()` method on the server's `weblogic.management.runtime.ServerRuntimeMBean`. For example, to monitor the progress of a long-running graceful shutdown process, issue a `getState` inquiry on a separate thread. For more information, see [ServerRuntimeMBean](#) in *WebLogic Server MBean Reference*.
- WebLogic Scripting Tool—See [Getting Runtime Information](#) in *WebLogic Scripting Tool*.

Understanding Server States in the Server Life Cycle

These sections describe each state in the WebLogic Server life cycle.

SHUTDOWN State

In the `SHUTDOWN` state, a WebLogic Server instance is configured but inactive.

A server instance enters the `SHUTDOWN` state as result of a Shutdown or Force Shutdown command. In addition, a server instance can kill itself when it detects, as a result of self-health monitoring, that it has become unstable. Only a server instance with its `Auto Kill If Failed` attribute is true will kill itself when it detects that it is failed. For more information, see [“Automatic Restart for Failed Server Instances” on page 5-2](#).

You can transition a server instance in the `SHUTDOWN` state to the `STARTING` state with the `Start`, `Start in Admin`, or `Start in Standby` commands.

STARTING State

During the `STARTING` state, a WebLogic Server instance transitions from `SHUTDOWN` to `STANDBY`, as a result of a `Start`, `Start in Admin`, or `Start in Standby` command.

In the `STARTING` state, a server instance cannot accept any client or administrative requests.

The server instance obtains its configuration data:

- An Administration Server retrieves domain configuration data, including the domain security configuration, from its `config` directory.
- A Managed Server contacts the Administration Server for its configuration and security data. If the Managed Server is configured for SSL communications, it uses its own certificate files, key files, and other SSL-related files and contacts the Administration Server for the remaining configuration and security data.

Note: If the Managed Server cannot contact its Administration Server, by default, it starts up in Managed Server Independence mode, using its locally cached copy of the domain `config` directory. See [“Understanding Managed Server Independence Mode” on page 5-10](#).

The server instance starts the services listed in [Table 6-1](#), in the order listed.

Table 6-1 Services Started in `STARTING` State

Service	Function
<code>weblogic.management.provider.internal.BeanInfoAccessService</code>	
<code>weblogic.management.PropertyService</code>	
<code>weblogic.management.internal.DomainDirectoryService</code>	
<code>weblogic.upgrade.domain.DomainUpgradeServerService</code>	
<code>weblogic.management.upgrade.ConfigurationMigrationService</code>	
<code>weblogic.deploy.service.internal.DeploymentService</code>	

Table 6-1 Services Started in STARTING State

Service (Continued)	Function
<code>weblogic.management.provider.internal.RuntimeAccessDeploymentReceiverService</code>	
<code>weblogic.management.provider.internal.RuntimeAccessService</code>	
<code>weblogic.diagnostics.lifecycle.DiagnosticInstrumentationService</code>	
<code>weblogic.t3.srvr.LicenseService</code>	
<code>weblogic.t3.srvr.BootService</code>	Includes basic services such as kernel, execute queues, and the server runtime.
<code>weblogic.management.provider.internal.DomainAccessService</code>	The root for Administration Server-only services.
<code>weblogic.diagnostics.lifecycle.DiagnosticFoundationService</code>	The container service for logging and debugging.
<code>weblogic.nodemanager.NMService</code>	The Node Manager service, responsible for reporting changes to server status to Node Manager via the server output stream.
<code>weblogic.timers.internal.TimerService</code>	
<code>weblogic.rjvm.RJVMService</code>	During shutdown, closes all RJVMs except the Administration Server connection.
<code>weblogic.protocol.ProtocolService</code>	
<code>weblogic.t3.srvr.DomainLibService</code>	Registers configured protocols, making them available for outbound traffic and inbound configuration. Managed Servers require this service to be available early in the startup sequence, to allow them to provide correct addressing information to the Administration Server.

Table 6-1 Services Started in STARTING State

Service (Continued)	Function
weblogic.server.channels.Channel Service	<p>This service is dependent on consistent configuration, and protocols being registered. By this point in the startup sequence, all protocols should have been registered.</p> <p>After this service starts, addressing information, such as <code>ServerChannelManager.findDefaultLocalServerChannel()</code>, is available.</p>
weblogic.server.channels.AdminPort Service	
weblogic.t3.srvr.ListenerService	
weblogic.transaction.internal.PrimordialTransactionService	<p>The transaction helper is initialized, providing utilities that associate transactions with threads, obtaining the Transaction Manager, obtain the <code>UserTransaction</code> object, and perform other tasks.</p> <p>Note: The transaction service itself is not enabled at this point in the startup sequence.</p>
weblogic.rmi.internal.RMIServerService	The RMI boot service that is used for initialization only.
weblogic.jndi.internal.NamingService	
weblogic.iiop.IIOPClientService	Installs VM-wide delegates.
weblogic.management.PrimordialManagementService	
weblogic.ldap.EmbeddedLDAP	
weblogic.security.SecurityService	
weblogic.jndi.internal.RemoteNamingService	
weblogic.security.acl.internal.RemoteSecurityService	

Table 6-1 Services Started in STARTING State

Service (Continued)	Function
<code>weblogic.rmi.cluster.RemoteBinderFactoryService</code>	
<code>weblogic.cluster.ClusterService</code>	
<code>weblogic.iiop.IIOPService</code>	
<code>weblogic.protocol.ProtocolHandlerService</code>	
<code>weblogic.management.internal.AdminService</code>	
<code>weblogic.xml.registry.XMLService</code>	
<code>weblogic.messaging.interception.MessageInterceptionService</code>	
<code>weblogic.cluster.migration.rmiservice.MigratableRMIService</code>	
<code>weblogic.messaging.interception.configuration.Configurator</code>	
<code>weblogic.drs.internal.DataReplicationService</code>	
<code>weblogic.management.provider.internal.EditAccessService</code>	Start the Management Edit Service.
<code>weblogic.health.HealthMonitorService</code>	
<code>weblogic.cluster.migration.MigrationService</code>	
<code>weblogic.t3.srvr.T3InitializationService</code>	Initializes deprecated T3 server services such as <code>BootServicesImpl</code> .
<code>weblogic.server.channels.ChannelRuntimeService</code>	Addressing information, such as <code>ServerRuntime.getListenAddress()</code> , and dynamic updates are available after this point in the startup sequence.

Table 6-1 Services Started in STARTING State

Service (Continued)	Function
weblogic.store.admin.DefaultStore Service	
weblogic.transaction.internal. TransactionService	
weblogic.jdbc.common.internal. JDBCService	
weblogic.connector.common. ConnectorService	
weblogic.store.admin.Store DeploymentService	
weblogic.jms.JMSServiceServerLifeCycle Impl	
weblogic.jms.BridgeService	
weblogic.application.Application ShutdownService	Checks pending application work during graceful shutdown. Applications are also shutdown here.
weblogic.messaging.saf.internal. SAFServerService	
weblogic.ejb20.deployer.EJB20Service	
weblogic.io.common.internal.File Service	
weblogic.time.server.TimerService	Cancels application triggers during shutdown.
weblogic.rmi.internal.HeartbeatHelpers ervice	Supports heartbeats in protocol-only clients.
weblogic.servlet.internal.WebService	
weblogic.webservice.conversation. internal.ConversationServiceImpl	
weblogic.wtc.gwt.WTCServerLife CycleImpl	

Table 6-1 Services Started in STARTING State

Service (Continued)	Function
<code>com.beasys.CORBA.pool.weblogic.WLECSERVICE</code>	
<code>weblogic.management.service.ManagedServerNotificationService</code>	
<code>weblogic.webservice.WSServerService</code>	
<code>weblogic.management.mbeanservers.runtime.internal.RuntimeServerService</code>	Run-time JMX services.
<code>weblogic.management.mbeanservers.edit.internal.EditServerService</code>	
<code>weblogic.management.mbeanservers.compatibility.internal.CompatibilityMBeanServerService</code>	
<code>weblogic.management.snmp.SNMPService</code>	
<code>weblogic.management.deploy.classdeployment.ClassDeploymentService</code>	Adds handling of startup and shutdown classes.
<code>weblogic.server.ServerLifecycleService</code>	Handles creation of the server life cycle runtime mbeans to allow for control of the domain.
<code>weblogic.server.channels.EnableAdminListenersService</code>	Enables Admin port before server goes into ADMIN state.
<code>domainweblogic.diagnostics.lifecycle.DiagnosticSystemService</code>	

STANDBY State

A server instance in *STANDBY* does not process any request—its regular Listen Port is closed. The Administration Port is open, and accepts life cycle commands that transition the server instance to either the *RUNNING* or the *SHUTDOWN* state. Other Administration requests are not accepted.

Starting a server instance in standby is a method of keeping it available as a “hot” backup, a useful capability in high-availability environments.

The only life cycle command that causes a server instance to enter the `STANDBY` state and remain in that state is the `Start in Standby` command. A server instance transitions through the `STANDBY` state when you issue a `Start` or a `Start in Admin` command.

ADMIN State

In the `ADMIN` state, WebLogic Server is up and running, but available only for administration operations, allowing you to perform server and application-level administration tasks. When a server instance is in the `ADMIN` state:

- The Administration Console is available.
- The server instance accepts requests from users with the `admin` role. Requests from `non-admin` users are refused.
- Applications are activated in the application `ADMIN` state. They accept requests only from users with the `admin` role. A user with the `admin` role accessing an application in the application `ADMIN` state continues to have access to all application functionality, not just administrative functions.
- The `JDBC`, `JMS`, and `JTA` subsystems are active, and administrative operations can be performed upon them. However, you do not have to have administrator-level privileges to access these subsystems when the server is in the `ADMIN` state.
- Deployments or re-deployments are allowed, and take effect when you transition the server instance from the `ADMIN` to the `RUNNING` state (using the `Resume` command).
- `ClusterService` is active and listens for heartbeats and announcements from other cluster members. It can detect that other Managed Servers have joined the cluster, but is invisible to other cluster members.

You can transition a server instance to the `ADMIN` state using the `Start in Admin`, `Suspend`, or `Force Suspend` commands.

A server instance transitions through the `ADMIN` state as a result of `Start`, `Shutdown`, and `Force Shutdown` commands.

You can transition a server instance in the `ADMIN` state to `RUNNING` with the `Resume` command, or to `SHUTDOWN`, with the `Shutdown` or `Force Shutdown` command.

RESUMING State

During this transitional state, WebLogic Server performs the operations required to move itself from the `STANDBY` or `ADMIN` state to the `RUNNING` state.

A server instance transitions to the `RESUMING` state when you issue the `Resume` command. A server instance transitions through the `RESUMING` state when you issue the `Start` command.

RUNNING State

In the `RUNNING` state, WebLogic Server is fully functional, offers its services to clients, and can operate as a full member of a cluster.

A server instance transitions to the `RUNNING` state as a result of the `Start` command, or the `Resume` command from the `ADMIN` or `STANDBY` states.

You can transition a server instance in the `RUNNING` state to the `SUSPENDING` state or the `FORCE_SUSPENDING` state using `graceful` and `force Suspend` and `Shutdown` commands.

SUSPENDING State

During this transitional state, WebLogic Server performs the operations required to place itself in the `ADMIN` state, suspending a subset of WebLogic Server subsystems and services in an ordered fashion, and completing a predefined portion of the application work currently in process (“in-flight” work).

A server instance transitions to the `SUSPENDING` state when you issue the `Suspend` command. A server instance transitions through the `SUSPENDING` state when you issue a `Shutdown` command.

For information about in-flight work, see [“Processing In-Flight Work During Suspend and Shutdown” on page 6-17](#).

Note: While in the `SUSPENDING` state, Work Managers complete in-flight processing for pending work in application threads. For more information, see [“Understanding Work Managers”](#) in *Configuring WebLogic Server Environments*.

FORCE_SUSPENDING State

During this transitional state, WebLogic Server performs the operations required to place itself in the `ADMIN` state, suspending a subset of WebLogic Server subsystems and services in an ordered fashion. During the `FORCE_SUSPENDING` state, WebLogic Server does not complete in-flight work gracefully; application work in progress is abandoned.

A server instance transitions through the `FORCE_SUSPENDING` state when you issue the `Force Suspend` or `Force Shutdown` command.

SHUTTING_DOWN State

During this transitional state, WebLogic Server completes the suspension of subsystems and services and does not accept application or administration requests.

A server instance transitions to the `SHUTTING_DOWN` state when you issue a Shutdown or Force Shutdown command.

FAILED State

A running server instance can fail as a result of out-of-memory exceptions or stuck application threads, or if one or more critical services become dysfunctional. A server instance monitors its health, and upon detecting that one or more critical subsystems are unstable, it declares itself `FAILED`.

A `FAILED` server instance cannot satisfy administrative or client requests.

When a server instance enters the `FAILED` state, it attempts to return to a non-failed state. If it failed prior to reaching the `ADMIN` state, the server instance shuts itself down with an exit code that is less than zero. For information about server exit codes, see [“WebLogic Server Exit Codes and Restarting After Failure” on page 5-6](#).

If the server instance fails after reaching the `ADMIN` state, but before reaching the `RUNNING` state, by default, it returns to the `ADMIN` state, if the administration port is enabled.

Note: If desired, you can configure a server instance that fails after reaching the `ADMIN` state, to shut itself down, rather than return to the `ADMIN` state

A server instance can enter the `FAILED` state from any other state. However, once a server instance has entered the `FAILED` state, it cannot return to a running state directly. The `FAILED` state is fatal and a server must go into the `ADMIN` or `SHUTDOWN` state before returning to the `RUNNING` state.

Note: It is theoretically possible that a server could become available again once the `FAILED` state is entered, for example if hung threads causing a failed state become unhung.

However, once `FAILED` state is entered, a server must go into the `ADMIN` or `SHUTDOWN` states before returning to `RUNNING`.

Using Server Life Cycle Commands

This section describes each life cycle command, how to issue it, and its effect on the state of a server instance. For more information on:

- How to issue life cycle commands, see:
 - “[Life Cycle Commands](#)” and “[Managing Servers and Server Life Cycle](#)” in *WebLogic Scripting Tool*
 - “[Start and Stop Servers](#)” in the *Administration Console Online Help*
 - “[Starting and Stopping Servers](#)” on page 2-1
- Node Manager processing related to key life cycle events in environments that use Node Manager, see “[How Node Manager Works in the WebLogic Server Environment](#)” on page 3-5.
- The processing that occurs during each life cycle state, see “[Understanding Server States in the Server Life Cycle](#)” on page 6-3.

For an illustration of the relationship between server states and server life cycle, see [Figure 6-1](#).

Start

The Start command transitions a server instance from the SHUTDOWN state to the RUNNING state. Depending on the initial state of a server instance, the Start command causes these state transitions:

SHUTDOWN→STARTING→STANDBY→ADMIN→RESUMING→RUNNING

The `ServerMBean.StartupMode` attribute lets you specify the state in which a server instance should be started. Its values are displayed and configurable in the Administration Console, using WLST, or when specified as a `java weblogic.Server` startup option. If you do not specify a startup mode value (either on the command line, in the Administration Console, or in `config.xml`), the default is to start in the RUNNING state.

For more information, see “[Specify a Startup Mode](#)” in the *Administration Console Online Help* and “[Options for Configuring Server Attributes](#)” in *WebLogic Server Command Reference*.

Command Usage

See “[start](#)”, “[startServer](#)”, and “[nmStart](#)” in *WebLogic Scripting Tool* and “[Start Managed Servers from the Administration Console](#)” in the *Administration Console Online Help*.

Start in Standby

The Start command, with Standby mode enabled, transitions a server instance from the SHUTDOWN state to the STANDBY state, with this sequence of state transitions.

SHUTDOWN→STARTING→STANDBY

Command Usage

See `-Dweblogic.management.startupmode` in *WebLogic Server Command Reference* and “[Start Managed Servers in Standby Mode](#)” in the *Administration Console Online Help*.

Start in Admin

The Start command, with Admin mode enabled, transitions a server instance from the SHUTDOWN state to the ADMIN state, with this sequence of state transitions:

SHUTDOWN→STARTING→STANDBY→ADMIN

Command Usage

See `-Dweblogic.management.startupmode` in *WebLogic Server Command Reference* and “[Start Managed Servers in Admin Mode](#)” in the *Administration Console Online Help*.

Resume

The Resume command transitions a server instance from the STANDBY or ADMIN state to the RUNNING state, with this sequence of state transitions:

STANDBY→ADMIN→RESUMING→RUNNING

Command Usage

See “[resume](#)” in *WebLogic Scripting Tool* and “[Resume a Server](#)” in the *Administration Console Online Help*.

Graceful Suspend

The Graceful Suspend command transitions a server instance from the RUNNING state to the ADMIN state, allowing work in process to be handled gracefully, with this sequence of state transitions:

RUNNING→SUSPENDING→ADMIN

Command Usage

See “[suspend](#)” in *WebLogic Scripting Tool* and “[Suspend a Server](#)” in the *Administration Console Online Help*.

Force Suspend

The Force Suspend command transitions a server instance from the `RUNNING` state to the `ADMIN` state, without handling work in process gracefully, with this sequence of state transitions:

`RUNNING`→`FORCE_SUSPENDING`→`ADMIN`

Command Usage

See “[Forcibly Suspend Servers](#)” in the *Administration Console Online Help*.

Graceful Shutdown

The Graceful Shutdown command transitions a server instance from the `RUNNING` state to the `SHUTDOWN` state, allowing work in process to be handled gracefully, with this sequence of state transitions:

`RUNNING`→`SUSPENDING`→`ADMIN`→`SHUTTING_DOWN`→`SHUTDOWN`

Command Usage

See “[shutdown](#)” in *WebLogic Scripting Tool* and “[Shut Down a Server Instance](#)” in the *Administration Console Online Help*.

Controlling Graceful Shutdown

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

- **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.

See “[Control Graceful Shutdowns](#)” and “[Shut Down Servers in a Cluster](#)” in the *Administration Console Online Help*.

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.

Force Shutdown

The Force Shutdown command transitions a server instance from the any state to the SHUTDOWN state, without allowing work in process to be handled gracefully. When run for a server instance in the RUNNING state, the Force Shutdown command results in these state transitions:

RUNNING→FORCE_SUSPENDING→ADMIN→STANDBY→SHUTDOWN

Command Usage

See “[shutdown](#)” in *WebLogic Scripting Tool* and “[Forcibly Shutdown Servers](#)” in the *Administration Console Online Help*.

A forced shutdown is immediate—WebLogic Server subsystems stop 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`.

Note: When you force shutdown a server instance in a cluster, a clustered service will fail over to another server instance in the cluster, if its state is replicated on another server instance. However:

- If you issue a Forced Shutdown command on a server instance that hosts an HTTP session for which a secondary session has not yet been created, the session will be lost.
- If you issue a Forced Shutdown command on a server instance that hosts the replicated state of a stateful session EJB, and the server instance that hosts the EJB fails (the primary), the EJB will not fail over, because its replicated state no longer exists.

For information about undeployment processes during a forced shutdown, and related programming considerations, see “[Shutdown Operations and Application Undeployment](#)” on [page 6-16](#).

Processing In-Flight Work During Suspend and Shutdown

The following sections describe how each subsystem handles work in process during `SUSPENDING` and `SHUTTING_DOWN` operations.

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 Container 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 `SERVER_NAME—>Control—>Start/Stop` page in the Administration Console, or the `-ignoreSessions` option with the WLST `shutdown` command.

In a cluster, when a primary session is dropped, the corresponding replicated sessions on another clustered instance will be also destroyed, in addition to the primary session on the server that is being gracefully shut down.

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 shut down 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 persistent 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 runtime 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 [“Force Shutdown” on page 6-16](#).

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. Force Shutdown suspends all pending work in all subsystems.

Understanding Server Life Cycle

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-1](#)
- [“Shutting Down Instances of WebLogic Server” on page A-3](#)

For a comprehensive discussion of starting and shutting down WebLogic Server instances, see [“Starting and Stopping Servers” on page 2-1](#).

Starting Instances of WebLogic Server

In the following table, *WL_HOME* refers to the top-level installation directory for WebLogic Server, such as `c:\bea\wlserver_10.0\`.

Table A-1 Starting Server Instances

To Start	Do The Following
The MedRec server	<p>Invoke:</p> <p><i>WL_HOME</i>\samples\domains\medrec\bin\startWebLogic.cmd (Windows)</p> <p><i>WL_HOME</i>/samples/domains/medrec/bin/startWebLogic.sh (UNIX)</p> <p>The server starts as an Administration Server in the medrec domain.</p> <p>On Windows, you can start the Medical Records Server from the Start menu (Examples→WebLogic Server→Start Medical Records Server). By default, the username and password for the medrec domain are set to weblogic.</p>
The Examples server	<p>Invoke:</p> <p><i>WL_HOME</i>\samples\domains\wl_server\bin\startWebLogic.cmd (Windows)</p> <p><i>WL_HOME</i>/samples/domains/wl_server/bin/startWebLogic.sh (UNIX)</p> <p>The server starts as an Administration Server in the wl_server domain.</p> <p>On Windows, you can start the Examples Server from the Start menu (Examples→WebLogic Server→Start Examples Server). By default, the username and password for the wl_server domain are set to weblogic.</p>
An Administration Server that you have created	<p>Invoke:</p> <p><i>DOMAIN_NAME</i>\bin\startWebLogic.cmd (Windows)</p> <p><i>DOMAIN_NAME</i>/bin/startWebLogic.sh (UNIX)</p> <p>where <i>DOMAIN_NAME</i> is the name of the directory in which you located the domain, typically <i>BEA_HOME</i>\user_projects\domains\<i>DOMAIN_NAME</i>.</p> <p>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, see “Provide User Credentials to Start and Stop Servers” on page 2-8.</p> <p>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.</p> <p>On Windows, the Configuration Wizard creates a shortcut on the Start menu to start the Administration Server that you created (User Projects→<i>DOMAIN_NAME</i>→Start Admin Server for WebLogic Domain).</p>

Table A-1 Starting Server Instances (Continued)

To Start	Do The Following
Managed Servers	<ol style="list-style-type: none"> 1. Start the domain's Administration Server. 2. Start the Node Manager on the computer that will host the Managed Server you want to start. The WebLogic Server custom installation process optionally installs and starts Node Manager as a Windows service on Windows systems. If it's not already running, you can start Node Manager manually at a command prompt or with a script. See "Starting Node Manager and Servers" on page 3-39. On Windows, you can use a shortcut on the Start menu to start the Node Manager (Tools→Node Manager). 3. Start the domain's Administration Console. See "Starting the Administration Console" in <i>Introduction to WebLogic Server and WebLogic Express</i>. 4. Associate Managed Servers with Node Manager by assigning them to a Machine upon which Node Manager runs. See "Configure Machines" and "Assign Servers Instances to Machines" in the <i>Administration Console Online Help</i>. 5. In the left pane of the Administration Console, expand Environment and select Servers. 6. In the Servers table, click the name of the Managed Server you want to start. 7. Select Control→Start/Stop. 8. In the Server Status table, select the check box next to the name of the server you want to start and click Start. 9. Click Yes to confirm. <p>For information on additional ways to start Managed Servers, see "Starting and Stopping Servers" on page 2-1.</p>
A cluster of Managed Servers	To start clustered Managed Servers with Node Manager, see "Start Managed Servers in a Cluster" in the <i>Administration Console Online Help</i> .

Shutting Down Instances of WebLogic Server

It is recommended that you shutdown WebLogic Server instances through the Administration Console:

Starting and Stopping Servers: Quick Reference

- See [“Shut Down a Server Instance”](#) and [“Control Graceful Shutdowns”](#) in the *Administration Console Online Help*.
- For information on gracefully shutting down the Managed Servers in a cluster, see [“Shut Down Servers in a Cluster”](#) in the *Administration Console Online Help*.

Alternatively, invoke a Weblogic Server stop script to shutdown the server. See [“Shutting Down Servers with a Stop Script”](#) on page 2-20.

On Windows, you can stop the Medical Records Server, Examples Server, and Administration Servers that you have created using the Configuration Wizard, from the Start menu.