BEA WebLogic Server®

Managing Server Startup and Shutdown

Version 9.0
Revised: July 22, 2005

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

3. Using Node Manager to Control Servers

4. Avoiding and Recovering From Server Failure

5. Understanding Server Life Cycle

CHAPTER 1

Introduction and Roadmap

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

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 J2EE 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, "Avoiding and Recovering From Server Failure," describes failover procedures for WebLogic Server instances.
- Chapter 5, "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

WebLogic Server 9.0 includes a number of new and changed features:

- **The WebLogic Scripting Tool (WLST) command-line interface to manage server life cycle**—WLST starts, stops, suspends, and resumes Administration and Managed Servers. Use WLST commands to control the life cycle states through which a server instance transitions and retrieve information about the runtime state of WebLogic Server instances.

  WLST can serve as a Node Manager command-line client. Use WLST commands with Node Manager to start, stop, and restart Administration and Managed Server instances locally or remotely. See "Managing Servers and Server Life Cycle" in *Weblogic Scripting Tool*.

- **New configuration directory**—The `config.xml` file is stored in the new *`DOMAIN_NAME`*`\config` directory, where *`DOMAIN_NAME`* is the root directory of the domain. In addition, the `config` directory contains new domain configuration files. See "WebLogic

Domain Directory Structure Enhancements" in *Upgrading WebLogic Application Environments* and "Domain Configuration Files" in *Understanding Domain Configuration*.

For new backup procedure recommendations, see "Directory and File Backups for Failure Recovery" on page 4-4.

- **Managed Servers cache configuration**—In previous versions, a Managed Server only saved a copy of its configuration data if Managed Server Independence was enabled. In WebLogic Server 9.0, Managed Servers automatically maintain a local copy of the domain configuration. Each time a Managed Server starts up, it downloads changes made to the domain configuration since it last updated its local copy. The `ServerMBean` attribute that previously controlled whether a Managed Server maintained a local version of its configuration—`MSIFileReplicationEnabled`—no longer exists. See "Understanding Managed Server Independence Mode" on page 4-10.

- **Server exit codes for failure detection**—New exit codes help administrators and system support personnel determine whether a server instance exited as a result of failure. See "WebLogic Server Exit Codes and Restarting After Failure" on page 4-5.

- **Enhanced life cycle management functions**—A new life cycle state, `ADMIN`, facilitates application redeployment, maintenance, and troubleshooting. In the `ADMIN` state, WebLogic Server is running, but available only for administration operations, allowing you to perform server and application-level administration tasks without risk to running applications. For more information, see "Understanding Server Life Cycle" on page 5-1.

A number of enhancements make Node Manager more versatile and easier to use:

- **Shell Script Node Manager**—A new version of Node Manager, implemented as a shell script, provides the same functionality as the Java Node Manager and can be used with the Secure Shell (SSH) or Remote Shell (RSH) protocol for secure remote control of server instances running on UNIX or Linux systems.

- **WLST commands access Node Manager**—Start, stop, and suspend server instances remotely or locally; obtain server status; and retrieve the contents of the server output log, without requiring the presence of a running Administration Server. In addition, you can enroll the machine on which WLST is running to be monitored by Node Manager. See *WebLogic Scripting Tool*.

- **Administration Server control**—Node Manager can start, stop, and restart Administration Servers. In previous versions, Node Manager required access to a running Administration Server, and could control and monitor only Managed Servers.

- **Node Manager and server migration**—Node Manager can migrate migratable servers in a WebLogic Server cluster. See "Server Migration" in *Using WebLogic Server Clusters*.

- **Improved diagnostics and logging**—Node Manager diagnostics are improved, and the logging strategy for Node Manager and the server instances it controls are simplified.
- **Simplified setup**—Node Manager setup is simplified; in particular, Node Manager no longer requires two-way SSL. Only one-way SSL is required.
- **Node Manager runs as a Windows service**—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 recommends using Node Manager to start and restart servers. See "About Node Manager Installation as a Windows Service" in the *Installation Guide*.

CHAPTER 2

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 4, "Avoiding and Recovering From Server Failure."

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

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—"Overview of Node Manager Configuration" on page 3-14
- Configure Managed Server connections to the Administration Server—"Configuring Managed Server Connections to the Administration Server" on page 2-15
- Specify Java startup options—"Specifying Java Options for a WebLogic Server Instance" on page 2-17

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:

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*.

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 2-15.

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 4-10.

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 `boot.properties`.	Uses them and does not prompt you for either credential.
Neither username nor password on the command line.	• Prompts you for the username. • Prompts you for the password twice.
Username but no password on the command line.	• Uses the username from the command line. • Prompts you for the password twice.
Password but no username on the command line.	• 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

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 Log and Configuration Files" on page 3-36.

The following sections describe working with boot identity files:

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 2-10, 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 Log and Configuration Files" on page 3-36.

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\myd
  omain\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 it 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 2-10 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:

- "Configuring Managed Server Connections to the Administration Server" on page 2-15
- "Specifying Java Options for a WebLogic Server Instance" on page 2-17
- "Changing the JVM That Runs Servers" on page 2-18

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/suppconfigs/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.

CHAPTER 3

Using Node Manager to Control Servers

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

Overview of Node Manager

Note: For a summary of new Node Manager features in this release of WebLogic Server, see "New and Changed Features" on page 1-2.

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 3-14.

Java-based Node Manager provides more security than the script-based version. See "Configuring Java-based Node Manager Security" on page 3-15.

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. See "Configuring Script-based Node Manager" on page 3-25.

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.

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.

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

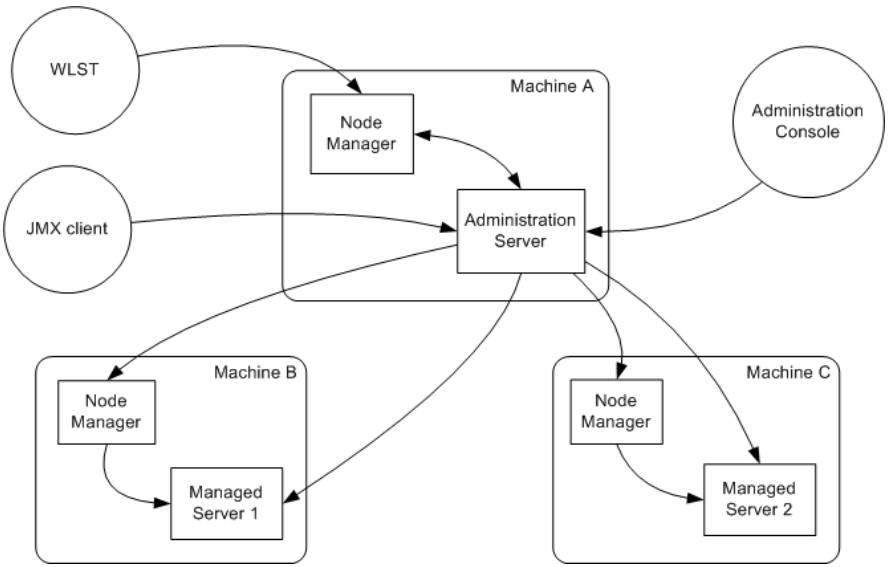
The following sections provide a "big picture" diagram of Node Manager's role in the WebLogic Server environment, as well as illustrations and descriptions of the processes Node Manager uses to communicate with servers:

- "Node Manager-Defined States for Restarting Managed Servers" on page 3-11
- "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



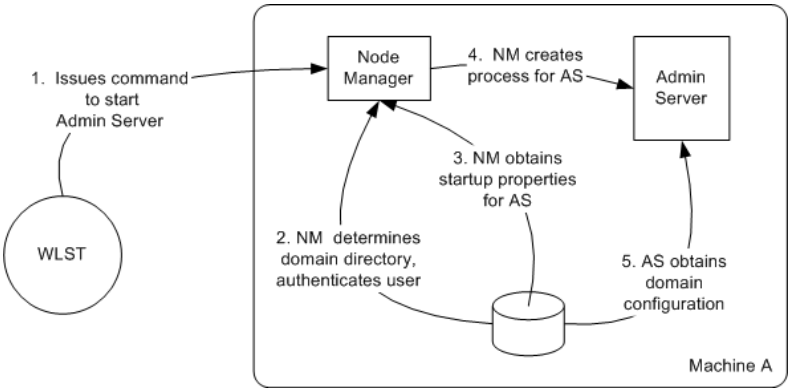
How Node Manager Starts an Administration Server

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.

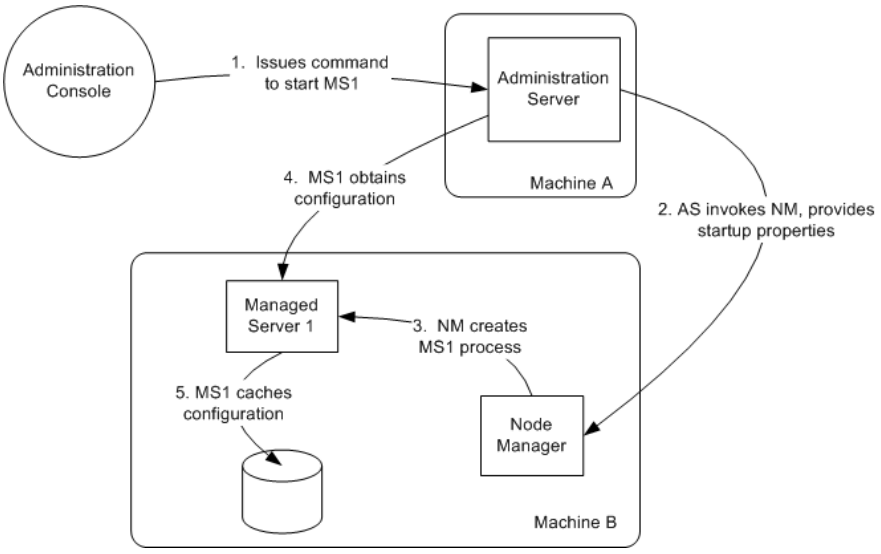
2. Node Manager looks up the domain directory in `nodemanager.domains`, and authenticates the user credentials using a local file that contains the encrypted username and password.
3. Node Manager creates the Administration Server process.
4. The Administration Server obtains the domain configuration 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 3-30.

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.

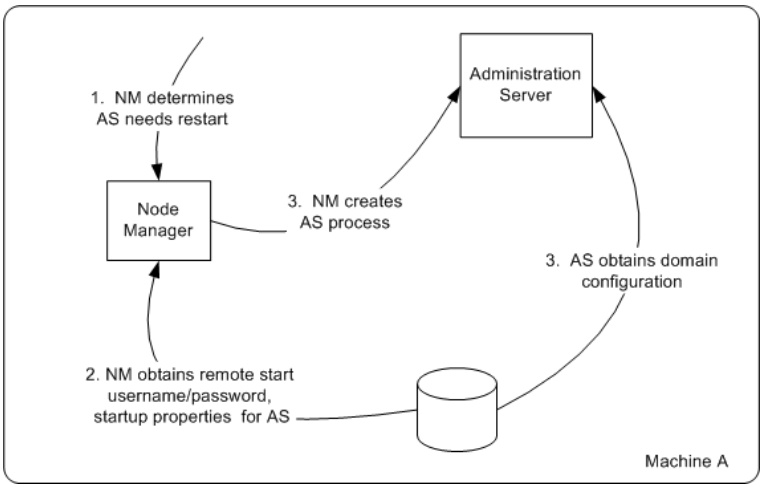
How Node Manager Restarts an Administration Server

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.

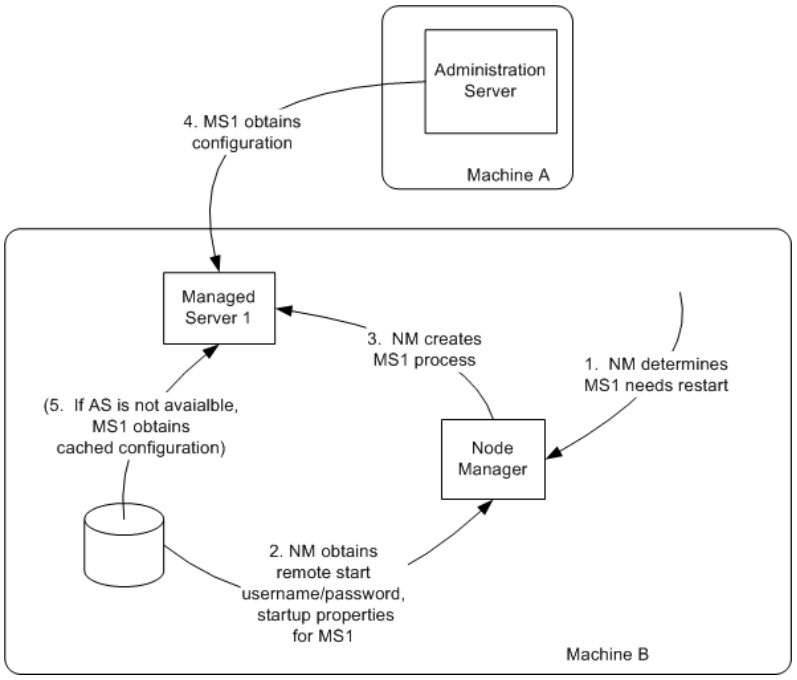
How Node Manager Restarts a Managed Server

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.

Node Manager-Defined States for Restarting Managed Servers

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

- `FAILED_RESTARTING`—Indicates that Node Manager is currently restarting a failed Managed Server.
- `FAILED_NOT_RESTARTABLE`—Indicates that the Managed Server has failed or was killed by Node Manager as a result of the Managed Server's AutoKillIfFailed attribute being set to True, but Node Manager cannot restart the Managed Server because its AutoRestart attribute is set to False.

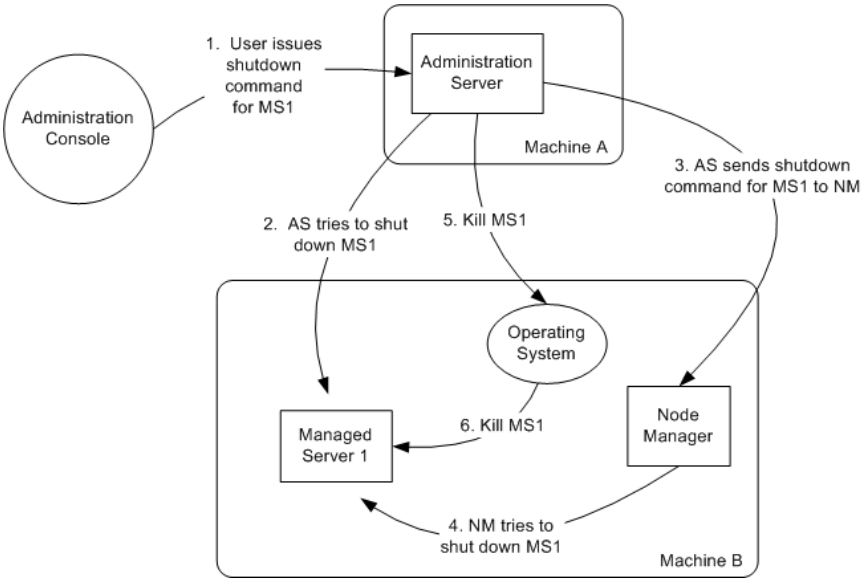
  This state may also occur if a server fails during startup.

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

The CrashRecoveryEnabled configuration property allows Node Manager to recover from a system crash. The property is enable by default.

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.

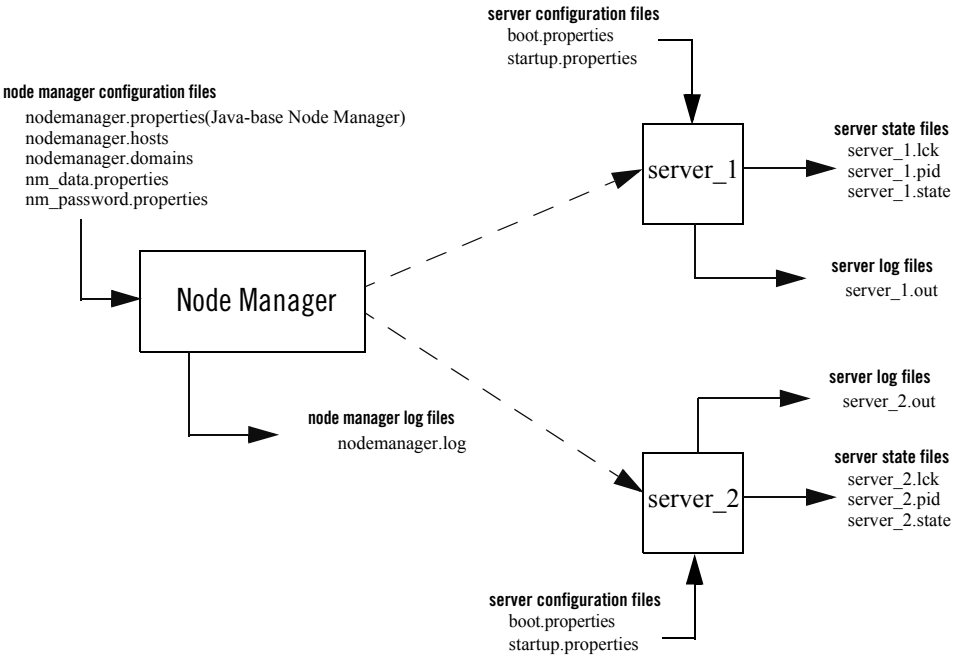
If the process is running, Node Manager performs an additional check to access the management servlet running in the process to verify that the process corresponding to the process ID is a WebLogic Server instance.

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

Diagram of 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 Log and Configuration Files" on page 3-36.

Figure 3-7 Node Manager Configuration and Logging Environment



Overview of Node Manager Configuration

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.

The following sections provide Node Manager configuration information:

- "Configuring Java-based Node Manager" on page 3-14
- "Configuring Script-based Node Manager" on page 3-25
- "Additional Configuration Information" on page 3-28

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

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.

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 3-1 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 3-1 Node Manager Properties

Node Manager Property	Description	Default
`LogFile (New)`	Location of the Node Manager log file.	`NodeManagerHome/nodemanager.log`
`LogLimit (New)`	Maximum size of the Node Manager Log specified as an integer. When this limit is reached, a new log file is started. Valid range for LogLimit is 0 to 2147483647 (int maximum).	`0`
`LogCount (New)`	Maximum number of log files to create when `LogLimit` is exceeded. Valid range for LogCount is 0 to 2147483647 (int maximum).	1
`LogAppend (New)`	If set to `true`, then a new log file is not created when the Node Manager restarts; the existing log is appended instead.	`true`
`LogToStderr (New)`	If set to `true`, the log output is also sent to the standard error output.	`false`
`LogLevel (New)`	Severity level of logging used for the Node Manager log. Node Manager uses the same logging levels as WebLogic server.	`INFO`
`LogFormatter (New)`	Name of formatter class to use for NM log messages.	`weblogic.nodemanager.server.LogFormatter`
`CrashRecoveryEnabled (New)`	Enables system crash recovery.	`true`
`SecureListener (New)`	If set to `true`, use the SSL listener, otherwise use the plain socket	`true`
`CipherSuite (New)`	The name of the cipher suite to use with the SSL listener.	`TLS_RSA_EXPORT_WITH_RC4_40_MD5`

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

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`

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

Node Manager Property	Description	Default
`ListenAddress`	Any address upon which the machine running Node Manager can listen for connection requests. This argument deprecates `weblogic.nodemanager.listenAddress`.	`null` With this setting, Node Manager will listen on any IP address on the machine
`ListenPort`	The TCP port number on which Node Manager listens for connection requests. This argument deprecates weblogic.nodemanager.listenPort.	`5556`
`NativeVersionEnabled`	A value of true causes native libraries for the operating system to be used. For UNIX systems other than Solaris, HP-UX, or Linux, set this property to false to run Node Manager in non-native mode. This will cause Node Manager to use the start script specified by the StartScriptEnabled property to start Managed Servers.	`true`

Node Manager Property	Description	Default
NodeManagerHome	Node Manager root directory which contains the following configuration and log files: • nm_data.properties • nodemanager.domains • nodemanager.log • nodemanager.properties For more information on these files, see "Node Manager Log and Configuration Files." **Note:** By default, NodeManagerHome is WL_HOME/common/nodemanager. In a production environment, you may want to customize the location of the Node Manager root directory.	NodeManagerHome
WeblogicHome	Root directory of the WebLogic Server installation. This is used as the default value of -Dweblogic.RootDirectory for a Managed Server that does not have a root directory configured in its Remote Start tab. If not specified in either place, Node Manager starts the Managed Server in the directory where Node Manager runs.	none

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 peformed 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 configration, 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:

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 unmout 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 "nmEnroll " in *WebLogic Scripting Tool*.

Deprecated Node Manager Properties

This section lists the Node Manager properties that are deprecated in WebLogic Server 9.0.

Note: These properties are published for backwards compatibility and should not be used. SSL configurations will continue to work when migrating to WebLogic Server 9.0. However, the trust key store is not used when running Node Manager. As in WebLogic Server 8.1, the Node Manager private key will have to be added to the trusted key store for all client machines accessing Node Manager.

Table 3-2 Deprecated Node Manager Properties

Node Manager Property	Description	Reason Deprecated
`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.

Node Manager Property	Description	Reason Deprecated
`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.
`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 Script-based Node Manager

The SSH Node Manager is a shell script, `wlscontrol.sh`, located in `NodeManagerHome/`. This file 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. 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 3-23
- "Creating a Node Manager User" on page 3-26
- "Configuring Script-based Node Manager Security" on page 3-26

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.

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 3-27.
- "Individual Key Value Pairs" on page 3-28

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

   ```
   024 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 v9.0.0
   ```

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.

Append the public key to the machine's `~/.ssh/authorized_keys` file user as described in step two in the previous section.

Additional Configuration Information

The following sections provide additional Node Manager configuration information:

- "Configuring a Machine to Use Node Manager" on page 3-29

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 3-1, "Node Manager Properties," on page 3-17. 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 Jrockit

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

1. A reboot causes a running Managed Server to be killed before the Node Manager and Administration Server operating system services are 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

The following server `startup properties` can be defined in startup.properties. These properties are passed to a Node Manager when a server is started. These are derived from the Server Mbean, or the Cluster Mbean if the server is part of a cluster. For more information, see Mbean reference.

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.
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.
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 `failed`.
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: `set JAVA_HOME=c:\bea\jdk131` Node Manager has the same JDK version requirements as WebLogic Server.
`WL_HOME`	WebLogic Server installation directory. For example: `set WL_HOME=c:\bea\weblogic700`
`PATH`	Must include the WebLogic Server `bin` directory and path to your Java executable. For example: `set PATH=%WL_HOME%\server\bin;%JAVA_HOME%\bin;%PATH%`

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

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

Starting and Running Node Manager

The following sections provide information on how to start and run Java-based and script-based Node Manager:

- "Configuring a Machine to Use Node Manager" on page 3-29
- "Starting Java-based Node Manager Using Scripts" on page 3-33
- "Running Script-based Node Manager" on page 3-34

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 3-32.

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 3-1, "Node Manager Properties," on page 3-17.

  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 3-16.

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 search 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'
```

where the parameter values are provided for each of the parameters, as defined in Table 3-4.

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`.

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 d`omains/ProductionDomain` directory.

Table 3-4 Shell Command

Parameter	Description	Default
%H	Host name of SSH server	`NodeManagerMBean.ListenAddress`
%P	Port number of SSH server	`NodeManagerMBean.ListenAddress` 22
%S	WebLogic server name	none
%D	WebLogic domain name	`ServerStartMBean.RootDirectory`

%R	Domain directory (server root)	`ServerStartMBean.RootDirectory`
%C	Node manager script command • `START`—Start server • `KILL`—Kill server • `STAT`—Get server status • `GETLOG`—Retrieve server output log. • `VERSION`—Return Node Manager version.	none

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.

Stopping Node Manager

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

Node Manager Log and Configuration Files

The following sections describe Node Manager configuration and log files:

- "Configuration Files" on page 3-36
- "Log Files" on page 3-38

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 3-16.

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

nodemanager.hosts

This file contains a list of all the trusted hosts that can issue commands to Node Manager.

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 3-29.

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 a username/password pair specific to the Node Manager server that is managing this domain. This is known as the Node Manager secret. The username and password are appended to a salt value (obtained from the SerializedSystemIni.dat of the domain) and SHA-hashed.

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 "Additional Configuration Information" on page 3-28.

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.

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.

Table 3-5 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 3-39.

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 debut 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.

Troubleshooting Node Manager

The table below describes common Node Manager problems and their solutions.

Table 3-6 Troubleshooting Node Manager.

Symptom	Explanation
Error message: `Could not start server 'MyServer' via Node Manager - reason: 'Target machine configuration not found'.`	You have not assigned the Managed Server to a machine. Follow the steps in "Configuring a Machine to Use Node Manager" on page 3-29.
Error message: `<SecureSocketListener: Could not setup context and create a secure socket on 172.17.13.26:7001>`	The Node Manager process may not be running on the designated machine. See "Starting and Running Node Manager" on page 3-33.

Symptom	Explanation
Self-health monitoring attributes are configured for a server, but Node Manager doesn't automatically restart the server.	To automatically reboot a server, you must configure the server's automatic restart attributes as well as the health monitoring attributes. "Starting and Running Node Manager" on page 3-33. In addition, in order to restart a Managed Server instance with Node Manager, you must have started the Managed Servers using Node Manager. You cannot automatically reboot servers that were started outside of the Node Manager process (for example, servers started directly at the command line).

Applications on the Managed Server are using the wrong directory for lookups.	Applications deployed to WebLogic Server should not operate on any assumptions about the current working directory. File lookups should generally take place relative to the Root Directory obtained with the `ServerMBean.getRootDirectory()` method (this defaults to the "." directory). For example, to perform a file lookup, use code similar to
	`String rootDir = ServerMBean.getRootDirectory();`
	`//application root directory`
	`File f = new File(rootDir + File.separator +`
	`"foo.in");`
	rather than simply:
	`File f = new File("foo.in");`
	If an application is deployed to a server that is started using Node Manager, use the following method calls instead:
	`String rootDir //application root directory`
	`if ((rootDir = ServerMBean.getRootDirectory()) ==`
	`null) rootDir =`
	`ServerStartMBean.getRootDirectory();`
	`File f = new File(rootDir + File.separator +`
	`"foo.in");`
	The `ServerStartMBean.getRootDirectory()` method obtains the Root Directory value that you specified when configuring the server for startup using Node Manager. (This corresponds to the `Root Directory` attribute specified the Configuration—>Remote Start page of the Administration Console.)

CHAPTER 4

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

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

Overload Protection

WebLogic Server 9.0 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 9.0 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 "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.

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 Hel*p.

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 Table 4-1.

Table 4-1 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 `STARTING` state, and does not achieve the `STANDBY` 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 4-2 Starting Up

Listen Address Definition	Admin Server Restart Scenario	
	Same Machine	**Different Machine**
Not defined	1. Start the Administration Server. Running MSs will reconnect automatically at the next `ReconnectIntervalSecs` To start an MS that was not running when AS failed, no change in command is required.	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.

Listen Address Definition	Admin Server Restart Scenario	
	Same Machine	**Different Machine**
DNS name or IP address of the host	1. Start the Admin Server. Running MSs will reconnect automatically at the next `ReconnectIntervalSecs` To start an MS that was not running when AS failed, no change in command is required	1. Install WLS. 2. Move data. 3. Move IP address. Running MSs will reconnect automatically at next `ReconnectIntervalSecs` To start an MS that was not running when AS failed, no change in command is required 4. If you do not move the IP address 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, you must supply the new Listen Address on the command line.
DNS name mapped to multiple hosts	1. Start the Admin Server. Running MSs will reconnect automatically at the next `ReconnectIntervalSecs` To start an MS that was not running when AS failed, no change in command is required	1. Install WLS. 2. Move data. Running MSs will reconnect automatically at next `ReconnectIntervalSecs` To start an MS that was not running when AS failed, no change in command is required.

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 4-4.
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 `AdminReconnectIntervalSecs`. By default, `AdminReconnectIntervalSecs` 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 4-12.

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 4-6.

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*.

CHAPTER 5

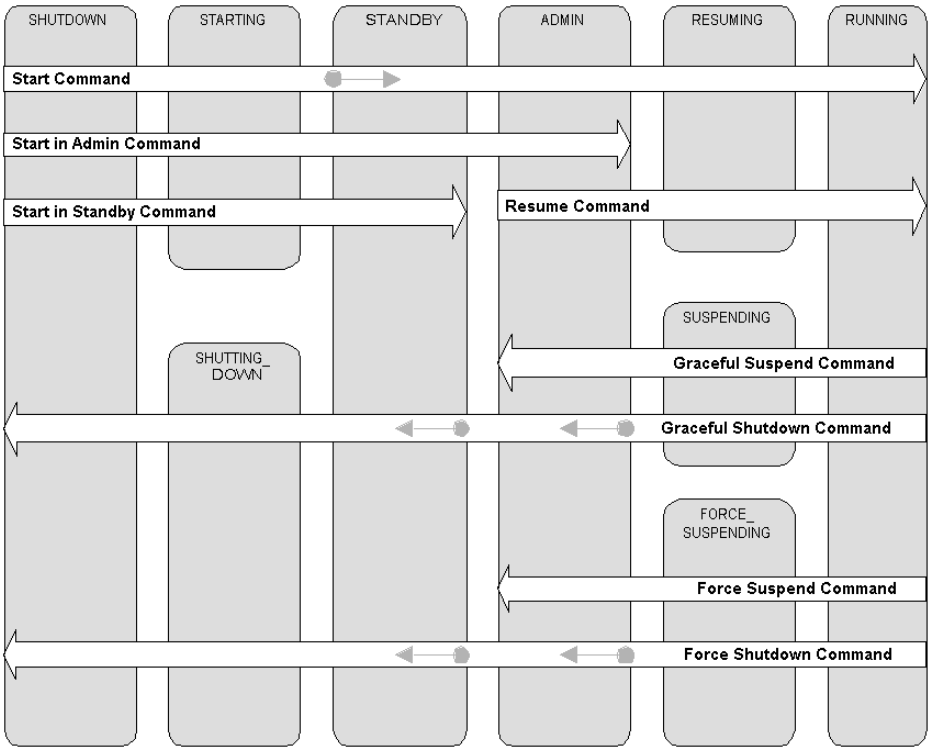
Understanding Server Life Cycle

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

Figure 5-1 illustrates the server life cycle and the relationships between states and life cycle commands.

Figure 5-1 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 5-3. For information on life cycle commands, see "Using Server Life Cycle Commands" on page 5-12.

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 "Monitoring Server State" 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 4-2.

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 4-10.

The server instance starts the services listed in Table 5-1, in the order listed.

Table 5-1 Services Started in STARTING State

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

Service (Continued)	Function
weblogic.management.provider.internal.RuntimeAccessDeploymentReceiverService	
weblogic.management.provider.internal.RuntimeAccessService	
weblogic.diagnostics.lifecycle.DiagnosticInstrumentationService	
weblogic.t3.srvr.LicenseService	
weblogic.t3.srvr.BootService	Includes basic services such as kernel, execute queues, and the server runtime.
weblogic.management.provider.internal.DomainAccessService	The root for Administration Server-only services.
weblogic.diagnostics.lifecycle.DiagnosticFoundationService	The container service for logging and debugging.
weblogic.nodemanager.NMService	The Node Manager service, responsible for reporting changes to server status to Node Manager via the server output stream.
weblogic.timers.internal.TimerService	
weblogic.rjvm.RJVMService	During shutdown, closes all RJVMs except the Administration Server connection.
weblogic.protocol.ProtocolService	
weblogic.t3.srvr.DomainLibService	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.

Service (Continued)	Function
`weblogic.server.channels.ChannelService`	This service is dependent on consistent configuration, and protocols being registered. By this point in the startup sequence, all protocols should have been registered. After this service starts, addressing information, such as `ServerChannelManager.findDefaultLocalServer Channel()`, is available.
`weblogic.server.channels.AdminPortService`	
`weblogic.t3.srvr.ListenerService`	
`weblogic.transaction.internal.PrimordialTransactionService`	The transaction helper is initialized, providing utilities that associate transactions with threads, obtaining the Transaction Manager, obtain the `UserTransaction` object, and perform other tasks. **Note:** The transaction service itself is not enabled at this point in the startup sequence.
`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`	
`weblogic.rmi.cluster.RemoteBinderFactoryService`	

Service (Continued)	Function
weblogic.cluster.ClusterService	
weblogic.iiop.IIOPService	
weblogic.protocol.ProtocolHandler Service	
weblogic.management.internal.AdminServ ice	
weblogic.xml.registry.XMLService	
weblogic.messaging.interception. MessageInterceptionService	
weblogic.cluster.migration.rmiservice. MigratableRMIService	
weblogic.messaging.interception. configuration.Configurator	
weblogic.drs.internal.DataReplication Service	
weblogic.management.provider.internal. EditAccessService	Start the Management Edit Service.
weblogic.health.HealthMonitorService	
weblogic.cluster.migration.Migration Service	
weblogic.t3.srvr.T3Initialization Service	Initializes deprecated T3 server services such as `BootServicesImpl`.
weblogic.server.channels.Channel RuntimeService	Addressing information, such as `ServerRuntime.getListenAddress()`, and dynamic updates are available after this point in the startup sequence.
weblogic.store.admin.DefaultStore Service	
weblogic.transaction.internal. TransactionService	

Service (Continued)	Function
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	
com.beasys.CORBA.pool.weblogic. WLECService	
weblogic.management.service.Managed ServerNotificationService	
weblogic.webservice.WSServerService	

Service (Continued)	Function
weblogic.management.mbeanservers. runtime.internal.RuntimeServerService	Run-time JMX services.
weblogic.management.mbeanservers. edit.internal.EditServerService	
weblogic.management.mbeanservers.compa tability.internal. CompatabilityMBeanServerService	
weblogic.management.snmp.SNMPService	
weblogic.management.deploy. classdeployment.ClassDeploymentService	Adds handling of startup and shutdown classes.
weblogic.server.ServerLifeCycleService	Handles creation of the server life cycle runtime mbeans to allow for control of the domain.
weblogic.server.channels.EnableAdmin ListenersService	Enables Admin port before server goes into ADMIN state.
domainweblogic.diagnostics.lifecycle. DiagnosticSystemService	

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 without risk to running applications. 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 can continue to run any application that is deployed on a server in a suspended state.
- The JDBC, JMS, and JTA subsystems are active, and administrative operations can be performed upon them.

  Note: You do not have to be an admin user to use these subsystems when the server is in a suspended 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 5-16.

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 4-5.

If the server instance fails after reaching the ADMIN state, in the RESUMING or 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.

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 5-3.

For an illustration of the relationship between server states and server life cycle, see Figure 5-1.

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 5-15.

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 5-15.

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.

APPENDIX A

Starting and Stopping Servers: Quick Reference

The following sections describe simple, frequently used ways to start and shut down instances of WebLogic Server:

- "Starting Instances of WebLogic Server" on page A-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\weblogic90\`.

Table A-1 Starting Server Instances

To Start	Do The Following
The MedRec server	Invoke: *WL_HOME*\samples\domains\medrec\bin\startWebLogic.cmd (Windows) *WL_HOME*/samples/domains/medrec/bin/startWebLogic.sh (UNIX) The server starts as an Administration Server in the medrec domain. 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.
The Examples server	Invoke: *WL_HOME*\samples\domains\wl_server\bin\startWebLogic.cmd (Windows) *WL_HOME*/samples/domains/wl_server/bin/startWebLogic.sh (UNIX) The server starts as an Administration Server in the wl_server domain. 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.
An Administration Server that you have created	Invoke: *DOMAIN_NAME*\bin\startWebLogic.cmd (Windows) *DOMAIN_NAME*/bin/startWebLogic.sh (UNIX) where *DOMAIN_NAME* is the name of the directory in which you located the domain, typically *BEA_HOME*\user_projects\domains*DOMAIN_NAME*. 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. **Note:** In a development environment, it is usually sufficient to start an Administration Server and deploy your applications directly onto the Administration Server. In a production environment, you typically create Managed Servers to run applications. On Windows, the Configuration Wizard creates a shortcut on the Start menu to start the Administration Server that you created (User Projects→*DOMAIN_NAME*→Start Admin Server for WebLogic Domain).

Table A-1 Starting Server Instances (Continued)

To Start	Do The Following
Managed Servers	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 and Running Node Manager" on page 3-33. 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 *Introduction to WebLogic Server and WebLogic Express*. 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 *Administration Console Online Help*. 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. For information on additional ways to start Managed Servers, see "Starting and Stopping Servers" on page 2-1.
A cluster of Managed Servers	To start clustered Managed Servers with Node Manager, see "Start Managed Servers in a Cluster" in the *Administration Console Online Help*.

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" 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 2-19.

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.