BEA WebLogic Server™®

Managing Server Startup and Shutdown

Version 9.0 BETA
Revised: December 15, 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

3. Avoiding and Recovering From Server Failure

A. Understanding Server Life Cycle

B. Starting and Stopping Servers: Quick Reference

Index

CHAPTER 1

Introduction and Roadmap

This section describes 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 and shutdown and the server life cycle. In addition, it describes WebLogic features for preventing and recovering from server failure.

This document is a resource for system administrators and operators responsible for monitoring and managing a WebLogic Server installation. This document 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 this guide and lists related documentation.
- Chapter 2, "Overview of Starting and Stopping Servers," describes several ways to start and stop server instances.
- Chapter 3, "Avoiding and Recovering From Server Failure," describes failover procedures for WebLogic Server instances.
- Appendix A, "Understanding Server Life Cycle," describes the operational phases of a WebLogic Server instance, from start up to shut down.
- Appendix B, "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 Startup and Shutdown

- WebLogic Server 9.0 provides a new command-line scripting interface for managing the server life cycle. The WebLogic Scripting Tool (WLST) lets you start and stop Administration Servers and Managed Servers without using Node Manager.

  Alternatively, WLST can serve as a Node Manager command-line client. You can use WLST commands with Node Manager to start, stop, and monitor server instances remotely.

  For more information, see "*WebLogic Scripting Tool*".

- In WebLogic Server 9.0, you can install Node Manager as a Windows service on a Windows host computer. BEA Systems recommends running Node Manager as an operating system service so that it is automatically restarted in the event of system failure or reboot and using Node Manager to start or restart servers.

  For more information, see "Installing the Node Manager as a Windows Service" in the *Installation Guide*.

- New Configuration Directory—In WebLogic Server 9.0, there are changes to where domain configuration data is stored. `config.xml` still exists, and is stored in the new *`domain_name`*`\config` directory, where *`domain_name`* is the root directory of the domain. In addition, the `config` directory contain new domain configuration files.

  For more information on domain directory files, see "Domain Configuration Files" in *Understanding Domain Configuration*.

  For new backup procedure recommendations, see "Directory and File Back Ups for Failure Recovery" on page 3-4.

- Managed Servers Cache Configuration—In previous versions, a Managed Servers only saved a copy of its configuration data if Managed Server Independence was enabled. In WebLogic Server 9.0, Managed Servers 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.

  For more information, see "Understanding Managed Server Independence Mode" on page 3-10.

- Server Exit Codes for Failure Detection—In WebLogic Server 9.0 there are new exit codes that help administrators and system support personnel determine whether a server instance exited as a result of failure.

  For more information, see "WebLogic Server Exit Codes and Restarting After Failure" on page 3-5.

- WebLogic Server 9.0 provides 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 Appendix A, "Understanding Server Life Cycle."

CHAPTER 2

Overview of 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 a graphical or command-line interface, and on whether you are using the 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.

The following sections describe starting and stopping server instances:

For a concise overview of starting and stopping servers, see "Starting and Stopping Servers: Quick Reference" on page B-1.

Version Requirements for Starting Servers

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

Starting Administration Servers

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*.

To start an Administration Server:

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

   For more information, 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*.

4. Run one of the following scripts:
   - `startWebLogic.cmd` (Windows)
   - `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`*`\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>
```

Alternate Ways to Start Administration Servers

The following sections describe alternate ways to start an Administration Server:

Starting an Administration Server from the Windows Start Menu

When you create an Administration Server on a Windows computer, the Configuration Wizard installs the server in the Windows Start Menu. You can start the server instance from the Windows Start Menu.

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 Administration Servers" on page 2-2. 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 can start a server instance by directly invoking `weblogic.Server` in a Java command or by creating your own scripts that invoke the `weblogic.Server` class. (The WebLogic Server startup scripts invoke `weblogic.Server` in a Java command.)

For information about invoking `weblogic.Server` in a Java command, see "weblogic.Server Command-Line Reference" in *WebLogic Server Command Reference*.

Starting an Administration Server Using 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 start, stop, and suspend Administration Servers. You can access these Node Manager features using the WebLogic Scripting Tool commands and scripts.

For more information, see "Starting an Administration Server With Node Manager" in *Designing and Configuring WebLogic Server Environments*.

Using the WebLogic Server custom installation process, you can install Node Manager as a Windows service on a Windows host computer. BEA Systems recommends running Node Manager as an operating system service so that it is automatically restarted in the event of system failure or reboot, and using Node Manager to start or restart servers.

For more information, see "Installing the Node Manager as a Windows Service" in the *Installation Guide* and "Restart Administration Servers and Managed Servers" in *Designing and Configuring WebLogic Server Environments*.

Starting an Administration Server Using WLST

Using WLST, you can start an Administration Server, with or without Node Manager. The WLST `startServer` command starts the Administration Server stand-alone, without using Node Manager. The server runs in a separate process from WLST; exiting WLST does not shut down the server. If you use the `startServer` command with WLST connected to Node Manager, Node Manager supports restarting, stopping, and monitoring the Administration Server.

For more information, see "Starting an Administration Server Without Node Manager" and "Using WLST and Node Manager to Manage Servers" in *WebLogic Scripting Tool*.

Starting Managed Servers from the Administration Console

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*.

To use the Administration Console to start a Managed Server:

1. If you have not already done so, create a Managed Server.

   For more information, see *Creating WebLogic Domains Using the Configuration Wizard*.

2. Associate Managed Servers with Node Manager by assigning them to a Machine upon which Node Manager runs.

   For more information, see "Configuring Machines" and "Assigning Servers to Machines" in *Creating WebLogic Domains Using the Configuration Wizard*.

3. Configure the Managed Server to communicate with Node Manager.

   For more information, see "Configure Server Instances for Node Manager Control" in *Designing and Configuring WebLogic Server Environments* and "Configuring Startup Arguments for Managed Servers" on page 2-6.

4. Start the Node Manager on the computer that hosts the Managed Server.

   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.

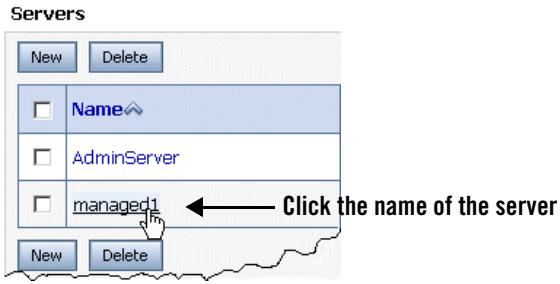
   For more information, see "Starting and Stopping Node Manager" in *Designing and Configuring WebLogic Server Environments*.

5. Start the domain's Administration Server and the Administration Console.

6. In the left pane of the Administration Console, expand the Environments node and click Servers.

7. Click the name of the Managed Server you want to start. (See Figure 2-1.)

Figure 2-1 Click Server Name

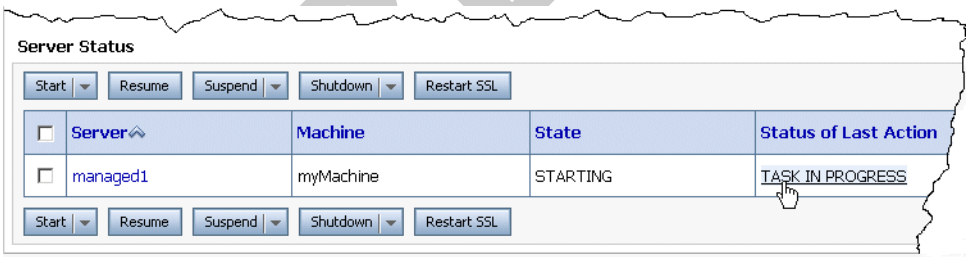


8. Select Control →Start/Stop.
9. In the Server Status table, select the check box next to the name of the server you want to start.
10. Select Start and Run and click Yes to confirm.

    The Node Manager starts the server on the target machine. When the Node Manager finishes its start sequence, the server's state is indicated in the State column in the Server Status table. (See Figure 2-2.)

11. To view messages that the Node Manager generated while starting the server, click the Task Completed link in the Status of Last Action column.

Figure 2-2 View the Node Manager Output



These messages are also written to the Node Manager log file for that server, as described in "Node Manager Log File For Each Server Instance" in *Designing and Configuring WebLogic Server Environments*.

Configuring Startup Arguments for Managed Servers

In most environments, the Node Manager can start a server without requiring you to specify startup options. However, if you have modified your environment; for example, if you have

added classes to the WebLogic Server classpath, you must specify startup options before you use the Administration Console to start a server.

To configure the startup options that Node Manager uses to start a Managed Server:

1. Start the domain's Administration Console.
2. In the left pane of the Administration Console, expand the Environments node and click Servers.
3. In the right pane, select a server, and select Configuration →Server Start.
4. In the Change Center (upper left pane of the Administration Console), click Lock & Make Changes.

   To protect your changes and to prevent others from making changes, a region in the Administration Console called the Change Center requires you to lock the Administration Console before you begin to modify a domain configuration. When you finish making changes, you save and distribute them to all server instances in the domain (Activate Changes), or you can roll back changes and release the lock (Undo All Changes).

   For more information, see "Change Management in the Administration Console" in *Understanding Domain Configuration*.
5. On the Server Start page, the User Name and Password fields contain values that you supplied when you used the Administration Console or the Configuration Wizard to create the server.

   If you want the server instance to run under a different WebLogic Server user account, enter the name of an existing user. The user must be in a role that has permission to start servers.

   For information on roles and permissions, see "Security Roles" in *Securing WebLogic Resources*.
6. Use the remaining fields on this page only if you want to override the default values that the Node Manager provides.

   For more information, see "Setting Up Node Manager" in *Designing and Configuring WebLogic Server Environments*.

   The Administration Console **replaces** the Node Manager defaults with the values you provide; it does not append the values to the Node Manager defaults. If you provide values for the Classpath field, make sure that you provide the full class path required to start the Managed Server.

Note: All paths refer to paths on the Node Manager machine.

For more information about the values to enter in these fields, see "Attributes" in the *Administration Console Online Help.*

7. Click Save and click Activate Changes in the Change Center.
8. If the server is running, restart it.

Starting Managed Servers From a WebLogic Server 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.

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 a Managed Server:

1. If you have not already done so, create a Managed Server.

   For more information, see *Creating WebLogic Domains Using the Configuration Wizard* or "Adding and Removing Servers in an Existing Domain" in the *Administration Console Online Help.*

2. Start the domain's Administration Server.
3. In a shell (command prompt) on the computer that you want to host the Managed Server, change to the directory that contains the `startManagedWebLogic` script:
   `domain-name``\startManagedWebLogic.cmd` (Windows)
   `domain-name``/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*.

4. 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\startManagedWebLogic.cmd
myManagedServer http://AdminHost:7001
```

For information on running the Managed Server on a remote WebLogic Server host, see "Setting Up and Starting Managed Servers on a Remote Machine" in *Creating WebLogic Domains Using the Configuration Wizard*.

For information on configuring a connection to the Administration Server, see "Configuring a Connection to the Administration Server" on page 2-19.

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*\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>
```

Alternate Ways to Start Managed Servers

The following sections describe alternate ways to start a Managed Server:

Starting a Managed Server in the STANDBY State

You can start a Managed Server so that at the end of its startup cycle, the server is in the STANDBY state. In this state, the server listens for administrative requests only on the domain-wide administration port and accepts life cycle commands that transition the server instance to either the RUNNING or the SHUTDOWN state. Other Administration requests are not accepted. For more information, see "STANDBY State" on page A-9.

When you are ready for the server to receive other types of requests on other listen ports, you resume it as described in "Resuming a Server" on page 2-21.

To start a Managed Server in the STANDBY state:

1. Start the domain's Administration Server.
2. Enable the domain-wide administration port. See "Enabling the Domain-Wide Administration Port" in the *Administration Console Online Help*.
3. In the left pane of the Administration Console, expand the Environments node and click Servers.
4. In the right pane, click the name of the Managed Server you want to start in the STANDBY state. (See Figure 2-1.)
5. Select Control →Start/Stop.
6. In the Server Status table, select the check box next to the name of the Managed Server.
7. Select Start in Standby Mode and click Yes to confirm.

   When you are ready for this server to receive non-administrative requests, see "Resuming a Server" on page 2-21.

Starting a Managed Server in the ADMIN State

You can start a Managed Server so that at the end of its startup cycle, the server is in the `ADMIN` state. In this 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. The `ADMIN` state is useful for application redeployment, maintenance and troubleshooting. For more information, see "ADMIN State" on page A-10.

When you are ready for the server to receive other types of requests on other listen ports, you resume it as described in "Resuming a Server" on page 2-21.

To start a Managed Server in the `ADMIN` state:

1. Start the domain's Administration Server.
2. Enable the domain-wide administration port. See "Enabling the Domain-Wide Administration Port" in the *Administration Console Online Help*.
3. In the left pane of the Administration Console, expand the Environments node and click Servers.
4. In the right pane, click the name of the Managed Server you want to start in the `ADMIN` state. (See Figure 2-1.)
5. Select Control →Start/Stop.
6. In the Server Status table, select the check box next to the name of the Managed Server.
7. Select Start in Admin Mode and click Yes to confirm.

   When you are ready for this server to receive non-administrative requests, see "Resuming a Server" on page 2-21.

Creating Scripts That Use the Node Manager

You can create your own scripts that use the Node Manager to start Managed Servers. The scripts must incorporate the `weblogic.Admin START` command (deprecated in this release of WebLogic Server) or the WLST `start` command.

For more information on WLST commands, see "WLST Command and Variable Reference" in *WebLogic Scripting Tool*.

Starting a Managed Server With the java weblogic.Server Command

The `weblogic.Server` class is the main class for a WebLogic Server instance. You can start a server instance by directly invoking `weblogic.Server` in a Java command or by creating your own scripts that invoke the `weblogic.Server` class. (The scripts that WebLogic Server creates invoke `weblogic.Server` in a Java command.)

For more information, see "weblogic.Server Command-Line Reference" in *WebLogic Server Command Reference*.

Starting a Managed Server If the Administration Server is Unavailable

Usually, a Managed Server contacts the Administration Server during its startup sequence to retrieve its configuration information. For 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 3-10.

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

Providing Usernames and Passwords to Start and Stop a Server

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 information on user credentials, roles, and permissions, see "Security Roles" in *Securing WebLogic Resources*.

This section describes the following tasks:

- "Specifying an Initial Administrative User for a Domain" on page 2-13
- "Boot Identity Files" on page 2-13
- "Specifying User Credentials When Starting a Server with the Node Manager" on page 2-18

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 "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, which contains an encrypted version of the username and password. For more information, see "Boot Identity Files" on page 2-13.

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 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. However, if you use the Node Manager to start a Managed Server, Node Manager encrypts and saves the credentials with which it started the server in a server-specific `boot.properties` file.

For more information about Node Manager security, see "Managed Server Remote Start Security" in *Designing and Configuring WebLogic Server Environments*.

The following sections describe working with boot identify 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 Configuration Files" 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 "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 "Using a Boot Identity File to Start a Server Instance" on page 2-16.

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

Alternative Technique for Creating a Boot Identity File for an Administration Server

If you invoke the `weblogic.Server` class directly on the command line, instead of following the steps in the previous section, you can create a boot identity file by 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=username
-Dweblogic.management.password=password
-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*.

Note: 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.

Creating a Boot Identity File for a Managed Server

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. For information about a server's root directory, see "A Server's Root Directory" in the *Administration Console Online Help*.

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 Security" in *Designing and Configuring WebLogic Server Environments*.

To create a boot identity file for a Managed Server:

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 "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 "Using a Boot Identity File to Start a Server Instance" on page 2-16.

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

Using a Boot Identity File to Start a Server Instance

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 the *Administration Console Online Help*.
- 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 2-14 or "Creating a Boot Identity File for a Managed Server" on page 2-15.

Removing a Boot Identity File 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`.

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 When Starting a Server with the Node Manager

If you use the Node Manager to start a Managed Server, you must provide user credentials on the 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 your credentials to the server's Server Start page.

If you want the server instance to run under a different WebLogic Server user account:

1. In the left pane of the Administration Console, expand the Environments node and click Servers.
2. In the right pane, click the name of the server instance.
3. Select Configuration →Server Start.
4. In the Username field, enter the name of an existing user.

   The user must be in a role that has permission to start servers. See "Security Roles" in *Securing WebLogic Resources*.
5. In the Password field, enter the password for the user that you specified.
6. Click Apply.
7. Restart the server in order to use these credentials.

Other Startup Tasks

The following sections describe miscellaneous startup tasks:

- "Configuring a Connection to the Administration Server" on page 2-19
- "Resuming a Server" on page 2-21
- "Specifying Java Options for a WebLogic Server Instance" on page 2-21
- "Changing the JVM that Runs Servers" on page 2-23

Configuring a Connection to the Administration Server

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, 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 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 by default:
   - `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.

     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 127.0.0.1:7002 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 "Configuring SSL" in *Securing WebLogic Server.* For more information on Administration Ports, see "Administration Port and Administrative Channel" in *Designing and Configuring WebLogic Server Environments*.

Resuming a Server

If you have started a server in the `STANDBY` or `ADMIN` state, when you are ready for the server to receive requests other than administration requests, do the following:

1. In the left pane of the Administration Console, expand the Environments node and click Servers.
2. In the right pane, click the name of a server that is in the `STANDBY` or `ADMIN` state.
3. Select Control →Start/Stop.
4. Select Resume, then click Yes to confirm and resume the server.

For information on how the server transitions from `STANDBY` or `ADMIN` to the `RUNNING` state, see "Life Cycle Commands" on page A-12.

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.

The following sections describe how to specify Java options for the JVM that runs a WebLogic Server instance:

- "Specifying Java Options for a WebLogic Server Startup Script" on page 2-22
- "Specifying Java Options for a Managed Server that the Node Manager Starts" on page 2-22

Specifying Java Options for a WebLogic Server Startup Script

If you use a WebLogic Server script to start servers, do the following:

1. Create a backup copy of the WebLogic Server start scripts:
   - For scripts that start an Administration Server, back up *domain-name*\startWebLogic.cmd (startWebLogic.sh on UNIX)
   - For scripts that start a Managed Server, back up *domain-name*\startManagedWebLogic.cmd (startManagedWebLogic.sh on 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*
2. Open the start scripts 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.
   - "JRockit Java Virtual Machine User Guide" 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. Restart the server.

Specifying Java Options for a Managed Server that the Node Manager Starts

If you use the Node Manager to start Managed Servers, do the following for each server:

1. In the left pane of the Administration Console, expand the Environments node and click Servers.
2. In the right pane, click the name of the Managed Server. (See Figure 2-1.)
3. Select Configuration →Server Start.

4. In the Arguments field, specify the Java options. If you specify multiple options, separate each option by a space.

   For more information, see:

   – "weblogic.Server Command-Line Reference" for information on the Java options that set runtime behavior of a WebLogic Server instance.

   – "JRockit Java Virtual Machine User Guide" 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.

5. Click Apply.

6. Restart the Managed 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 WebLogic 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 Supported Platforms 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

You can do any of the following to shut down a WebLogic Server instance:

- Using the Administration Console:
  - "Shutting Down a Server" on page 2-24
  - "Controlling Graceful Shutdowns" on page 2-25
- Using WLST
  - `shutdown`
- "Killing the JVM" on page 2-26

Shutting Down a Server

To shut down a server from the Administration Console:

1. In the left pane of the Administration Console, expand the Environments node and click Servers.
2. In the right pane, click the name of a server you want to shutdown. (See Figure 2-1.)
3. Select Control →Start/Stop.
4. In the Server Status table, select the check box next to the name of server you want to shutdown.
5. Select one of the following:
   - Shutdown →When work completes.

     This command initiates a graceful shutdown, which gives WebLogic Server subsystems time to complete certain application processing currently in progress. For more information, see "Controlling Graceful Shutdowns" on page 2-25.

– Shutdown →Force shutdown now.

  This command initiates a forced shutdown, in which the server instructs subsystems to immediately drop in-flight requests. For more information, see "Force Shutdown" on page A-14.

6. Click Yes to confirm and shut down the server.

If you shut down the Administration Server, the Administration Console is no longer active.

Controlling Graceful Shutdowns

A graceful shutdown gives WebLogic Server subsystems time to complete certain application processing currently in progress. See "Graceful Shutdown" on page A-13.

To control the length of the graceful shutdown process:

1. In the left pane of the Administration Console, expand the Environments node and click Servers.
2. In the right pane, click the name of a server you want to shutdown. (See Figure 2-1.)
3. Select Control →Start/Stop.
4. If you want the server's graceful shutdown to drop all HTTP sessions immediately instead of waiting for them to complete or timeout, select the Ignore Sessions During Shutdown check box.

   Waiting for abandoned HTTP sessions to timeout can significantly lengthen the graceful shutdown process because the default session timeout is 24 hours.
5. The Graceful Shutdown Timeout field 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.

   To limit the amount of time the server takes to complete a graceful shutdown, enter the maximum number of seconds in the Graceful Shutdown Timeout box.

   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.

   If you do not supply a timeout value, the server waits indefinitely to complete a graceful shutdown.

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

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.

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

WebLogic Server Availability and Failure Recovery Features

This section describes WebLogic features for preventing and recovering from failure.

Overload Protection

WebLogic Server 9.0 detects increases in system load that can effect application performance and stability, and allows administrators to configure failure prevention actions that occur automatically at predefined load thresholds.

Overload protection is a key way of avoiding 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 (JRockit only)

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 homogeneously—on each Managed Server—in a cluster, so that if the server instance upon which a service fails, the service can fail over to another server in the cluster, without interruption in service or loss of state.

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.

When used in combination with Node Manager, server self-health monitoring enables you to automatically reboot servers that have failed. This improves the overall reliability of a domain,

and requires no direct intervention from an administrator. For more information, see "Using Node Manager to Control Servers" in *Designing and Configuring WebLogic Server Environments*.

Server-Level Migration

WebLogic Server 9.0 provides the capability to migrate clustered server instances that host *singleton services*. 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. 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.

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 Back Ups 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 3-1.

Table 3-1 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" in *Designing and Configuring WebLogic Server Environments*.

Restarting an Administration Server

See "Overview of Starting and Stopping Servers" on page 2-1.

Restarting an Administration Server on the Same Machine

Table 3-2 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 Back Ups for Failure Recovery" on page 3-4.
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 3-12.

Starting a Managed Server When the Administration Server Is Accessible

If the Administration Server is reachable by Managed Server that failed, you can:

- Restart it manually or automatically using Node Manager—You must configure Node Manager and the Managed Server to support this behavior. For details, see "Configure Monitoring, Shutdown, and Restart for Managed Servers" in *Designing and Configuring WebLogic Server Environments*.
- Start it manually with a command or script—For instructions, see "Overview of 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 2-13.

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 3-6.

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 "Moving a Server to Another Machine" and "Transaction Recovery After a Server Fails" in *Administration Console Online Help*.

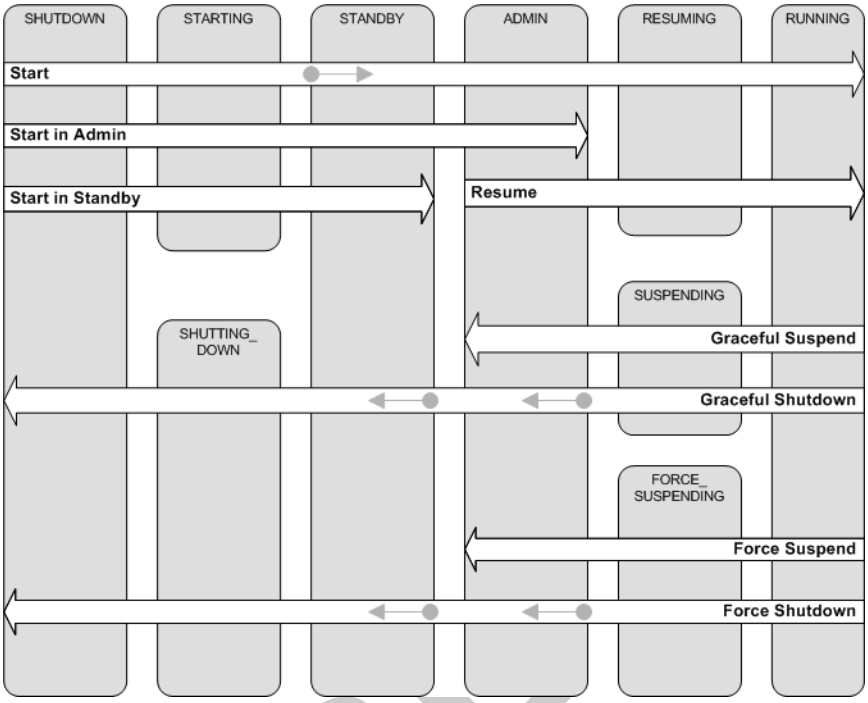
Understanding Server Life Cycle

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

Life Cycle State Transitions

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

Figure A-1 State Transitions for Server Life Cycle Commands



To understand each state and the relationships among states, see "WebLogic Server States" on page A-2. For information on life cycle commands, see "Life Cycle Commands" on page A-12.

WebLogic Server States

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

- Monitor the availability of server instances and the applications they host.
- Perform day-to-day operations tasks, including startup and shutdown procedures.
- Diagnose problems with application services.
- Plan correction actions, such as migration of services, when a server instance fails or crashes.

Getting Server State

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

- Administration Console—Multiple pages display state information:
  - The Servers table on the Summary of Servers page displays the current state of each server instance in the current domain.
  - The Server—Monitoring page displays the state of the current server instance, and the date and time it entered the state.
  - The log file for a server, available from Diagnostics—Log Files, includes timestamped messages for state transitions that have occurred since the server instance was last started.
- Programmatically—You can obtain the state of a server instance programmatically, using the `getState()` method on the server's `weblogic.management.runtime.ServerRuntimeMBean`. For example, to monitor the progress of a long-running graceful shut down process, issue a `getstate` inquiry on a separate thread. For more information see ServerRuntimeMBean in *WebLogic Server MBean Reference*.
- WebLogic Scripting Tool—For information about obtaining state information using WLST, see "Information Commands" in *WLST Command and Variable Reference.*

Understanding Server States

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 server instances whose `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 3-2.

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 of 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. For more information, see "Understanding Managed Server Independence Mode" on page 3-10.

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

Table A-1 Services Started in STARTING State

Service	Notes
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)	Notes
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)	Notes
`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 to that provide services such as getting associating transactions associated with threads, obtaining the transaction manager, and obtaining the `UserTransaction` object. **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`	

Service (Continued)	Notes
weblogic.rmi.cluster.RemoteBinder FactoryService	
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	

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

Service (Continued)	Notes
weblogic.webservice.WSServerService	
weblogic.management.mbeanservers.runtime.internal.RuntimeServerService	Run-time JMX services.
weblogic.management.mbeanservers.edit.internal.EditServerService	
weblogic.management.mbeanservers.compatability.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.EnableAdminListenersService	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 phase 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's Administration Port is open, and accepts requests from users with the admin role. Requests from non-admin users are refused.
- Applications are activated in the application ADMIN state—this means they accept request only from users with the admin role. A user with the admin role accessing an application in the application ADMIN state has access to all application functionality, not just administrative functions.
- The JDBC, JMS and JTA subsystems are active, and administrative operations can be performed upon them.
- 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.

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 a Start command, from the STANDBY or ADMIN 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 "In-Flight Work Processing" on page A-15.

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 *Designing and 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 Shutdown or a 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 3-5.

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 Admin 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.

Life Cycle Commands

This section describes WebLogic Server life cycle commands and their effect on the state of a server instance. For information on:

- How to issue life cycle commands, see "Overview of Starting and Stopping Servers" on page 2-1.
- Node Manager processing related to key life cycle events in environments that use Node Manager, see "Node Manager Processes and Communications" in *Designing and Configuring WebLogic Server Environments.*
- The processing that occurs during each life cycle state, see "WebLogic Server States" on page A-2.

Start

The Start command transitions a server instance from the SHUTDOWN or STANDBY 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

Start in Standby

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

SHUTDOWN→STARTING→STANDBY

Start in Admin

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

SHUTDOWN→STARTING→STANDBY→ADMIN

Resume

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

ADMIN→RESUMING→RUNNING

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

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

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

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→Control→Start/Stop tab:

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

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`

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 A-14.

In-Flight Work Processing

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 Contains waits until a secondary session is created, or until the session times out, whichever occurs first.
- JDBC persistence and file persistence—The Web Container immediately suspends sessions that are replicated in a database or file store.

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

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 shutdown has a JMS server:

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

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

- Client connections are closed.

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

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

JDBC Service

The JDBC Service closes idle connections in the connection pools.

Note: If connections are still in use, the shutdown of the JDBC service will fail, and the graceful shutdown will not complete. To shut down a server instance while applications still hold connections, use a forced shutdown command, described in "Force Shutdown" on page A-14.

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 Force Shutdown command. A Force Shutdown suspends all pending work in all subsystems.

Starting and Stopping Servers: Quick Reference

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

For a comprehensive discussion of starting and shutting down WebLogic Server instances, see "Overview of 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 Platform.

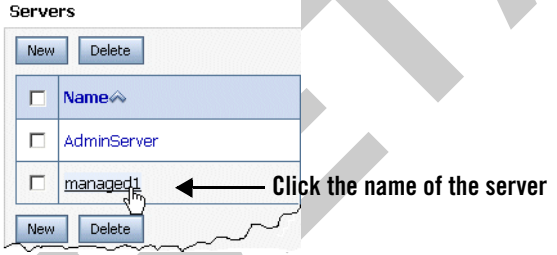
Table B-1 Starting Server Instances

To Start	Do The Following
The MedRecServer sample server	Invoke: *WL_HOME*\samples\domains\medrec\startWebLogic.cmd (Windows) *WL_HOME*/samples/domains\medrec/startWebLogic.sh (UNIX) The server starts as an Administration Server in the MedRec domain. On Windows, you can also start the Medical Records Server from the Start menu.
The Examples server	Invoke: *WL_HOME*\samples\domains\wl_server\startWebLogic.cmd (Windows) *WL_HOME*/samples/domains/wl_server/startWebLogic.sh (UNIX) The server starts as an Administration Server in the Examples domain. On Windows, you can also start the Examples Server from the Start menu.

Table B-1 Starting Server Instances (Continued)

To Start	Do The Following
An Administration Server that you have created	Invoke: *domain_directory*\startWebLogic.cmd (Windows) *domain_directory*/startWebLogic.sh (UNIX) where *domain_directory* is the directory in which you created the domain. If the server prompts you to enter a username and password, enter the name of a WebLogic Server user who has permission to start servers. For more information, see "Providing Usernames and Passwords to Start and Stop a Server" on page 2-12. **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 server that you create.

Table B-1 Starting Server Instances (Continued)

To Start	Do The Following
A Managed Server that you have created	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. (If you have configured the Node Manager as a Windows service, it starts automatically when you boot the Windows host computer.) For more information, see "Starting and Stopping Node Manager" in *Designing and Configuring WebLogic Server Environments*. 3. Start the domain's Administration Console. See "Starting the Administration Console" in the *Overview of WebLogic Server System Administration*. 4. In the left pane of the Administration Console, expand the Environments node and click Servers. 5. Click the name of the Managed Server you want to start. (See Figure B-1.) **Figure B-1 Click Server Name**  6. Select Control →Start/Stop. 7. In the Server Status table, select the check box next to the name of the server you want to start. 8. Select Start and Run and click Yes to confirm. For information on additional ways to start Managed Servers, see "Overview of Starting and Stopping Servers" on page 2-1.

Shutting Down Instances of WebLogic Server

The recommended procedure for shutting down a server is as follows:

1. Start the domain's Administration Console. See "Starting the Administration Console" in the *Overview of WebLogic Server System Administration*.

2. In the left pane of the Administration Console, expand the Environments node and click Servers.
3. Click on the name of a server you want to shutdown. (See Figure B-1.)
4. Select Control →Start/Stop.
5. In the Server Status table, select the check box next to the name of the server you want to shutdown.
6. Select Shutdown when work completes and click Yes to confirm.

   This initiates a graceful shutdown, in which the server notifies subsystems to complete all in-flight work requests. After the subsystems complete their work, the server stops.

Index