BEAProducts

Upgrading WebLogic Application Environments

BEA WebLogic Server® Version 10.0
Document Date: March 30, 2007

Contents

1. Introduction

2. Roadmap for Upgrading Your Application Environment

3. Upgrading a Security Provider

4. Upgrading Node Manager

5. Upgrading a WebLogic Domain

A. Upgrading WebLogic Server 9.0, 9.1, or 9.2 Application Environments to 10.0

B. WebLogic Server 10.0 Compatibility with Previous Releases

C. WebLogic Domain Directory Structure Enhancements

D. Upgrade Wizard Command-Line Reference

E. Silent Upgrade XML Script Reference

F. Upgrading a Domain at Administration Server Startup (Implicit Mode)

G. WebLogic Server Rolling Upgrade

Introduction

This document describes the procedures to upgrade your application environment to WebLogic 10.0. An application environment includes applications, the WebLogic domains in which they are deployed, and any application data associated with the domain, and may include external resources, such as database servers, firewalls, load balancers, and LDAP servers.

WebLogic 10.0 includes powerful tools to assist you with upgrading your application environments including the WebLogic Upgrade Wizard for upgrading domains, custom security providers, and custom node managers.

Most WebLogic Server applications can be run without modifications in the new WebLogic Server 10.0 application environment.

The following sections provide an overview of the topics discussed in this chapter:

- Overview of the Upgrade Process
- How the Upgrade Wizard Simplifies the Upgrade Process
- Interoperability and Compatibility with Previous Releases

Important Terminology

We recommend that, before proceeding, you familiarize yourself with the following terminology:

- Upgrade—To update BEA Products from a previous release or service pack to a more recent one. This process may include updating an existing application or domain configuration to run in a more recent version of WebLogic Server.

- Migrate—To move an application or domain configuration from a third-party product to a BEA product.

- Interoperability—(1) The ability of an application deployed in one release or service pack to communicate with another application that is deployed in a different release or service pack. (2) The ability of BEA product components to communicate with third-party software via standard protocols.

- Compatibility—The capability of an application built using one release or service pack to run in another release or service pack, regardless of whether the application was rebuilt.

Overview of the Upgrade Process

The process required to upgrade an application environment depends on the scope of the application. An *application environment* includes a WebLogic domain and any applications and application data associated with the domain. It may also include external resources, such as firewalls, load balancers, and LDAP servers. Figure 1-1 shows an example of a WebLogic application environment.

Figure 1-1 Example WebLogic Application Environment
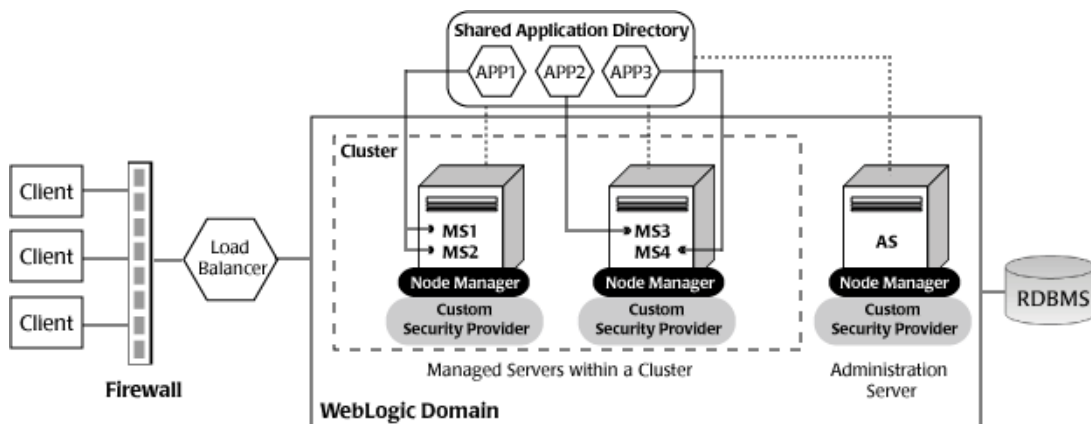
The following table lists the components of the WebLogic application environment shown in Figure 1-1, and the upgrade requirements for each.

Table 1-1 Upgrade Requirements for Components in Example WebLogic Application Environment

Component	Description	Upgrade Requirements
WebLogic domain	Includes the Administration Server (AS) and optionally one or more Managed Servers (for example, MS1, MS2, MS3, and MS4). The servers in a domain may span multiple machines. Furthermore, you can group Managed Servers into clusters to support load balancing and failover protection for critical applications. For more information about WebLogic domains, see "Understanding WebLogic Server Domains" in *Understanding Domain Configuration* at http://e-docs.bea.com/wls/doc100/domain_config/understand_domains.html.	Upgrade the domain directory on each machine in the domain.
Custom security provider	Supports custom security requirements. For information about developing custom security providers, see *Developing Security Providers for WebLogic Server* at http://e-docs.bea.com/wls/docs100/dvspisec/index.html.	Upgrade the custom security providers on each machine in the domain.
Node Manager	Provides high availability to Managed Servers. For more information about Node Manager, see "Using Node Manager to Control Servers" in *Managing Server Startup and Shutdown* at http://e-docs.bea.com/wls/docs100/server_start/nodemgr.html.	Upgrade custom Node Manager on each machine in the domain.

Table 1-1 Upgrade Requirements for Components in Example WebLogic Application Environment (Continued)

Component	Description	Upgrade Requirements
Applications	Any Java EE applications, including Web applications, EJBs, and so on. Typically, applications are deployed to one or more Managed Servers in a domain. Depending on the deployment strategy, applications may reside locally on a machine or be accessible via a shared directory. In addition, external client applications may access the application environment from outside a firewall.	Most WebLogic Server applications can be run without modifications in the new WebLogic Server 10.0 application environment. For more information, see "Interoperability and Compatibility with Previous Releases" on page 1-4.
External resources	Software components, such as databases for storing domain and application data, load balancers, and firewalls.	Verify that all external resources are compatible with WebLogic Server 10.0. For more information, see: *WebLogic Server 10.0 Supported Configurations*

How the Upgrade Wizard Simplifies the Upgrade Process

The WebLogic Upgrade Wizard guides you through the steps required to upgrade a WebLogic domain that is compatible with WebLogic Server 7.0, or 8.1 such that it runs in a WebLogic Server 10.0 application environment. As part of the upgrade process, you must upgrade any custom security providers and Node Managers used in the domain.

You can also use the WebLogic Upgrade Wizard to upgrade a WebLogic domain that is compatible with WebLogic Server 9.0, 9.1, or 9.2 to 10.0, but this is optional. This type of domain runs under WebLogic Server 10.0 without modification.

You can step through the upgrade process interactively, using the graphical user interface (GUI), or "silently," by creating an upgrade script and running it. Note that the Silent Mode is supported for upgrading WebLogic Server domains only.

Interoperability and Compatibility with Previous Releases

Application environments that run with WebLogic Server 10.0 can interact with application environments built on WebLogic Server 7.0, 8.1, or 9.*x*.

Most existing WebLogic Server applications can be run without modification in the new WebLogic Server 10.0 application environment. You should review the compatibility information described in "WebLogic Server 10.0 Compatibility with Previous Releases" on page B-1 to determine whether any feature changes affect the applications in your environment. If your application uses APIs that have been deprecated or removed, then you may encounter warnings or exceptions at run time.

Introduction

Roadmap for Upgrading Your Application Environment

This section describes how to prepare for and perform an upgrade of your WebLogic application environments. Topics include:

- Plan the Upgrade

- Prepare to Upgrade

- Upgrade Your Application Environment

- Complete Post-Upgrade Procedure

- What to Do If the Upgrade Process Fails

If upgrading from WebLogic 9.x, see Appendix A, "Upgrading WebLogic Server 9.0, 9.1, or 9.2 Application Environments to 10.0."

Plan the Upgrade

Planning how you will upgrade an application environment is an important step in the process. To ensure that your plan addresses all of the aspects of upgrading that are necessary for your environment, complete the following steps:

- Step 1: Inventory the Application Environment

- Step 2: Verify Supported Configuration Information

- Step 3: Review the Compatibility Information

- Step 4: Create an Upgrade Plan

Step 1: Inventory the Application Environment

Generate an inventory of the application environment, and identify the components defined in the following table. You can view a sample application environment in "Overview of the Upgrade Process" on page 1-2.

Table 2-1 Checklist for the Application Environment Inventory

	Identify the ...
☐	Administration Server and the machine on which it resides
☐	Managed Servers and the machine(s) on which they reside
☐	Custom security providers used in the domain
☐	Custom Node Managers used in the domain
☐	Location of the applications (including all external client applications)
☐	External resources, for example: • Databases used to store persisted and application data • Firewalls • Load balancers
☐	Tools, scripts, templates, and source code used for automating the tasks required to create the application environment

Step 2: Verify Supported Configuration Information

Verify support for all the hardware and software components in the application environment. The following table lists the key components for which you need to verify support.

Table 2-2 Verify Supported Configuration Information

To verify ...	See ...
OS and hardware support	"List of Supported Operating System Configurations" in *WebLogic Server 10.0 Supported Configurations.*
Database support	"Supported Database Configurations" in *WebLogic Server 10.0 Supported Configurations.*

Please note the following:

- WebLogic Server 10.0 supports PointBase 5.1.

  – If you are using a pre-5.1 version of PointBase, you can continue to use that version, if desired. See "Step 2: Re-apply Customizations to Startup Scripts" *after* you upgrade your domain.

    Note: You can use the pre-5.1 version of PointBase only for WebLogic Server domains.

  – If you are using a pre-5.1 version of PointBase and you want to update your environment to use PointBase 5.1, perform the steps below *before* you upgrade your domain:

    1. Update references to database names within configuration files.

    2. Before starting the pre-5.1 PointBase server, add the following information to the CLASSPATH (temporarily) to ensure that the pre-5.1 PointBase server is upgraded to 5.1:
       WL_HOME\common\eval\pointbase\lib\pbupgrade51.jar, where **WL_HOME** represents the root directory of the WebLogic Server 10.0 installation, for example, c:\bea\wlserver_10.0 (on Windows).

    3. Update your applications to use PointBase 5.1. Note that the PointBase 5.1 database software is installed as part of the 10.0 installation. See "Step 4: Install Required BEA Products" on page 2-7.

Table 2-2 Verify Supported Configuration Information (Continued)

To verify ...	See ...

Database Support (Continued)	• The WebLogic jDriver for Oracle was removed in the 9.0 release. If the JDBC connection pools in your domain use the WebLogic jDriver to create database connections, you will need to reconfigure the upgraded data sources to use a different JDBC driver, which may include changing the driver class name, the database URL format, and properties sent to the driver when creating database connections. As a replacement, you can use any JDBC driver that implements the specification and is thread safe. The driver you select must be installed (in the CLASSPATH) on all servers on which the data source is deployed. For more information about supported JDBC drivers, see "Supported Database Configurations" in *WebLogic Server 10.0 Supported Configurations*.
	
	The WebLogic Type 4 JDBC Driver for Oracle and the Oracle Thin Driver are installed with WebLogic Server and are ready for use. For more information about using these drivers, see "The Oracle Driver" in *WebLogic Type 4 JDBC Drivers* at `http://e-docs.bea.com/wls/docs100/jdbc_drivers/oracle.html` and "Third-Party JDBC Drivers Installed with WebLogic Server" in *Configuring and Managing WebLogic JDBC* at `http://e-docs.bea.com/wls/docs100/jdbc_admin/third_party_drivers.html#oracle_thin`.
	
	• If you are using an Oracle OCI database driver and want to change to use a Thin database driver, you must remove the server property (as illustrated below) from the generated JDBC module. For example:
	<pre><property>
	<name>server</name>
	<value>*servername*</value>
	</property></pre>
	• The Oracle 8.1.7 Oracle Thin driver (`oracle.jdbc.driver.OracleDriver`) is no longer supported in 10.0. If your domain contains JDBCConnectionPools that are configured to use this driver, it is recommended that you reconfigure the connection pools to use another driver. Use of this driver will cause domain upgrade to fail during database upgrade.

Web servers, browsers, and firewalls	"Supported Web Servers, Browsers, and Firewalls" *Supported Configurations for WebLogic Server 10.0*.

Step 3: Review the Compatibility Information

Most existing WebLogic Server applications can be run without modification in the new WebLogic Server 10.0 application environment. However, you should review "WebLogic Server 10.0 Compatibility with Previous Releases" on page B-1 to determine whether any feature changes affect the applications in your environment.

Step 4: Create an Upgrade Plan

Using the information gathered in the preceding steps, create a plan for upgrading your application environment. Identify the scope and timing of the upgrade process, based on your business needs. Please note the following:

- BEA does not recommend upgrading an application environment that is currently deployed in production. Instead, you should upgrade your application environment while it is under development or test and execute standard procedures for quality assurance and performance tuning before promoting the upgraded environment to production.

- If your application is complex, for example, if it includes multiple clustered domains and a large number of deployed applications, you may choose to upgrade the components of the application environment in stages.

- You may consider limiting the number of WebLogic Server versions used in any single application environment to minimize the diversity and cost of systems being administered.

- If you use transactional message-driven beans (MDBs) driven by non-WebLogic XA-enabled JMS providers, you must quiesce gracefully all pending transactions before shutting down and upgrading a server. Pending transactions cannot be recovered after an upgrade. For more information about message-driven beans, see "Message-Driven EJBs" in *Programming WebLogic Enterprise JavaBeans* at
  http://e-docs.bea.com/wls/docs100/ejb/message_beans.html.

Prepare to Upgrade

Before you upgrade the application environment, you must perform the following steps:

- Step 1: Undeploy Your Applications

- Step 2: Shut Down Servers in the Application Environment

- Step 3: Back Up the Application Environment

- Step 4: Install Required BEA Products

- Step 5: Prepare the Remote Managed Server Domain Directories
- Step 6: Set Up the Environment

Step 1: Undeploy Your Applications

WebLogic Server applications do not need to be undeployed before upgrading the domain. In most cases, WebLogic Server applications can be run without modifications in the new WebLogic Server 10.0 application environment. Review the compatibility information in "WebLogic Server 10.0 Compatibility with Previous Releases" on page B-1 to determine whether any features changes affect the applications in your environment. Note that if you use deprecated or removed APIs in the application, you might encounter warnings or exceptions at run time.

Step 2: Shut Down Servers in the Application Environment

Before you upgrade, you must shut down all servers in the application environment.

Step 3: Back Up the Application Environment

You have the option of backing up the domain during the upgrade process, as described in "Backup Domain" on page 5-10. However, the wizard archives the domain directory only; it does not preserve file permissions.

BEA recommends that before upgrading your application environment, you manually back up the components defined in the following table. You should back up the relevant information on all machines in the domain.

Table 2-3 Recommendations for Backing Up the Application Environment

Component	Recommendations
Domain directory	Back up the Administration Server and any remote Managed Server domain directories that are defined in the application environment.
	By default, the Configuration Wizard creates a domain directory in the `BEA_HOME\user_projects` directory, where `BEA_HOME` specifies the directory in which you installed WebLogic Server.
Applications and application-persisted data	Back up any applications and data that reside outside the domain directory.
	By default, applications are created in the `BEA_HOME\user_projects\applications` directory, where `BEA_HOME` specifies the directory in which you installed WebLogic Server.

Table 2-3 Recommendations for Backing Up the Application Environment (Continued)

Component	Recommendations
Custom security providers	Back up any custom security providers that you are using in your application environment.
	By default, security providers are located in `WL_HOME\server\bin\mbeantypes`, where `WL_HOME` specifies the root directory of the WebLogic Server installation.
Node Manager directory and scripts	Back up any Node Manager directories and scripts that you are using to manage your servers in a clustered environment.
	The names of the directory and script depend on your operating system:
	• Windows: `WL_HOME\common\nodemanager` `WL_HOME\server\bin\startNodeManager.cmd`
	• UNIX: `WL_HOME/common/nodemanager` `WL_HOME/server/bin/startNodeManager.sh`
	In the preceding pathnames, `WL_HOME` represents the root directory of the WebLogic Server installation, for example, `c:\bea\wlserver_10.0`.
Log files	If it is important for you to maintain a record of all messages that are logged, back up the log files. As log files can be large, you may want to delete them to conserve disk space, if it is not important to maintain them.

Step 4: Install Required BEA Products

Before upgrading your application environment, you must install the 10.0 BEA Products that you require on each machine in the domain. For more information about installing BEA Products 10.0, see *WebLogic Server Installation Guide* at

http://e-docs.bea.com/common/docs100/install/index.html.

Note: If you are using Node Manager in your pre-10.0 installation, when installing the 10.0 product, you should set the Node Manager listen port to match the port number used in the pre-10.0 installation, if possible. The default value for the listen port for Node Manager is 5556.

Step 5: Prepare the Remote Managed Server Domain Directories

Some configurations include Managed Servers running on one or more machines that are remote from the Administration Server for the domain. If you have this type of configuration, you must upgrade the domain directories on each of the machines that host the remote Managed Servers.

To prepare the remote domain directories, you must copy the following files from the root directory of the pre-upgraded domain directory on the host machine for the Administration Server to the **root** directory of the host domain(s) of the remote Managed Servers:

- `config.xml` (configuration file)
- `SerializedSystemIni.dat`

Note: If the database in your configuration is not compatible with WebLogic Server 10.0, the data must be upgraded to a database that is supported before it can be used in the new application environment. For more information, see "Step 4: Create an Upgrade Plan" on page 2-5.

Step 6: Set Up the Environment

To set up the environment for an upgrade:

1. Open an MS-DOS command prompt window (on Windows) or a command shell (on UNIX).

2. Add the WebLogic Server classes to the `CLASSPATH` environment variable and `WL_HOME`\server\bin to the `PATH` environment variable, where `WL_HOME` refers to the top-level installation directory for WebLogic Server.

   You can use the `WL_HOME`\server\bin\setWLSenv script to set both variables.

3. If you use JMS JDBC stores:

   a. Make sure the JDBC driver classes are added to the `CLASSPATH` environment variable.

   b. Start the corresponding database.

Upgrade Your Application Environment

The following provides a roadmap that identifies the steps required to upgrade your application environment.

Figure 2-1 Roadmap for Upgrading Your Application Environment
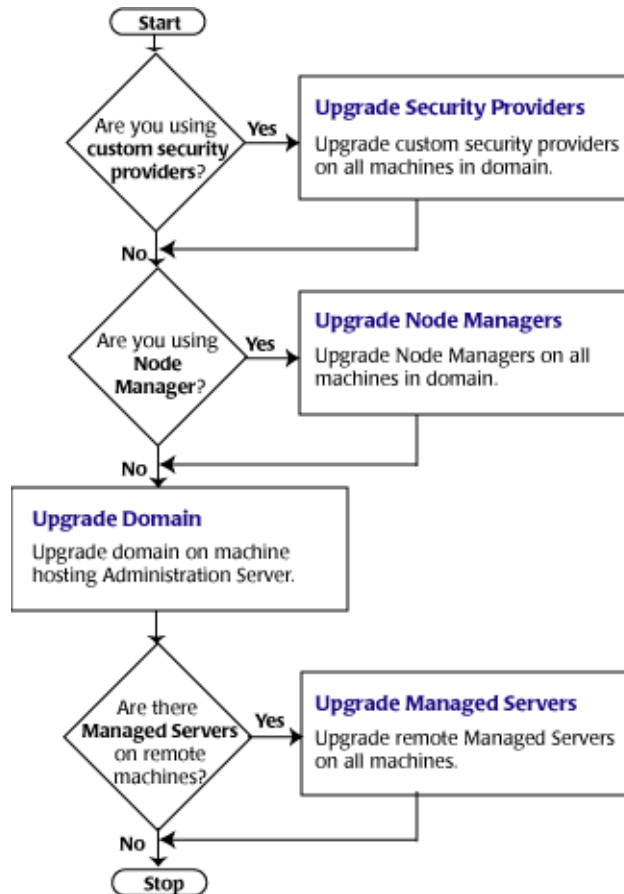
The following table summarizes the steps for updating an application environment. Some steps are mandatory, others are optional. Each step that is performed must be done on every machine in the domain.

Table 2-4 Procedure for Upgrading an Application Environment

Step	Description
1. Upgrade custom security providers	If are using custom security providers in your current application environment and you wish to continue using them in the new environment, upgrade them: • On all machines in the environment. • Before you upgrade the WebLogic domain. **Note:** If you are installing WebLogic Server 10.0 into an existing BEA Home directory that contains an installation of WebLogic Server 7.0 or 8.1, all custom security providers that reside in the default location, `WL_HOME\server\lib\mbeantypes`, where `WL_HOME` specifies the root directory of the pre-10.0 installation, are upgraded automatically. If all of your custom security providers reside in the default location, then the security provider upgrade step is complete, and you do not have to perform the steps described in this section.
2. Upgrade Node Managers	If you are currently using a customized version of Node Manager to provide high availability to Managed Servers and you wish to continue doing so in the new application environment, upgrade Node Manager: • On all machines in the environment • Before you start the servers in an upgraded WebLogic domain
3. Upgrade WebLogic domain (Administration Server)	Upgrade the WebLogic domain on the machine that hosts the Administration Server. **Note:** BEA recommends that you upgrade the Administration Server for a domain before the Managed Servers.
4. Upgrade WebLogic domain (remote Managed Servers)	Upgrade the WebLogic domain on every machine that hosts any Managed Servers. Be sure to copy the appropriate files to the Managed Servers before performing the upgrade, as described in "Step 5: Prepare the Remote Managed Server Domain Directories" on page 2-8. **Note:** Managed Servers that reside on the same machine as the Administration Server do not require additional upgrade steps.

Complete Post-Upgrade Procedure

Once you have used the WebLogic Upgrade Wizard to upgrade the application environment, you may need to perform the following steps:

- Step 1: Upgrade Your Application Infrastructure
- Step 2: Re-apply Customizations to Startup Scripts
- Step 3: Verify File Permissions
- Step 4: Enroll the Machine with Node Manager
- Step 5: Verify Remote Server Startup Options
- Step 6: Promote the Application Environment to Production

Not all of these steps are required for all situations. Review the sections to determine which, if any, of these steps are appropriate for your environment.

Step 1: Upgrade Your Application Infrastructure

Due to recent changes in the MBean hierarchy, BEA does not guarantee that existing configuration and administration scripts (such as WLST, `wlconfig`, `weblogic.Admin`, Ant, and so on) will run in all 10.0 environments. BEA recommends that you update your scripts to take advantage of the new features provided with WebLogic Server 10.0. For more information about new WebLogic Server features and changes in the MBean hierarchy, see "What's New in WebLogic Server 10.0" in the *WebLogic Server and WebLogic Express Release Notes* at `http://e-docs.bea.com/wls/docs100/notes/new.html`.

More information is provided in the following sections about scripting tools, custom configuration templates, and SNMP:

- "Deprecated Scripting Tools" on page 2-11
- "Using the WebLogic Scripting Tool" on page 2-12
- "Upgrading Your Custom Configuration Templates" on page 2-12
- "Using SNMP to Monitor WebLogic Server" on page 2-13

Deprecated Scripting Tools

The following configuration and administration tools are deprecated as of WebLogic Server 9.0:

- `weblogic.Admin` utility. For more information, see
  http://e-docs.bea.com/wls/docs100/admin_ref/cli.html.

If you are currently using one of these utilities, BEA recommends that you use the BEA WebLogic Scripting Tool, as described in the next section.

Using the WebLogic Scripting Tool

The WebLogic Scripting Tool (WLST) is a command-line scripting interface (built with Jython) that you can use to configure WebLogic domains. Using WLST, WebLogic Server administrators can perform administrative tasks and initiate WebLogic Server configuration changes interactively or by running an executable script.

With the release of WebLogic Platform 8.1, WLST was made available for evaluation from BEA's dev2dev site, but it was not formally included in the WebLogic Platform 8.1 product. It was offered in two formats: WLST online and WLST offline.

In WebLogic Server 10.0, the online and offline versions of WLST are delivered as a single tool. WLST fully supports the administrative and configuration features offered by 10.0. For more information about WLST, see *WebLogic Scripting Tool* at http://e-docs.bea.com/wls/docs100/config_scripting/index.html.

Note: WLST online can be run on WebLogic Server 9.0, 9.1, 9.2, and 10.0. WLST offline can be run on WebLogic Server 9.0, 9.1, 9.2, and 10.0.

As with the other pre-10.0 tools, BEA does not guarantee that existing WLST scripts will run in 9.2 or 10.0 due to recent changes in the MBean hierarchy. BEA recommends that you update your scripts to take advantage of the new features provided with WebLogic Server 9.2 and 10.0.

For more information about new WebLogic Server features and changes in the MBean hierarchy, see "What's New in WebLogic Server 10.0" in the *WebLogic Server and WebLogic Express Release Notes* at http://e-docs.bea.com/wls/docs100/notes/new.html.

Upgrading Your Custom Configuration Templates

The following table summarizes the steps required to upgrade custom configuration templates created with the WebLogic Platform 8.1 Template Builder.

Table 2-5 Upgrade Procedure for Custom Domain Templates

Step	More Information
1. Using the Upgrade Wizard, upgrade the domain that was created using the custom configuration and/or extension template.	Follow the steps described in "Upgrade Your Application Environment" on page 2-8.
2. Modify the domain configuration to leverage new features, as appropriate.	See "What's New in WebLogic Server 10.0" in the *WebLogic Server and WebLogic Express Release Notes* at http://e-docs.bea.com/wls/docs100/notes/new.html.
3. Use the Template Builder or `pack` command to create a 10.0 domain and/or extension template.	See: • *Creating Templates Using the Domain Template Builder* at http://e-docs.bea.com/common/docs100/tempbuild/index.html. • *Creating Domains and Templates Using the pack and unpack Commands* at http://e-docs.bea.com/common/docs100/pack/index.html.

Using SNMP to Monitor WebLogic Server

If you use an SNMP manager to monitor WebLogic Server:

1. Load the WebLogic Server 10.0 MIB into your SNMP manager.

   The MIB is located at
   BEA_HOME/wlserver_10.0/server/lib/BEA-WEBLOGIC-MIB.asn1. WebLogic Server does not change object identifiers (OIDs) for existing managed objects; it only adds new OIDs for new managed objects.

2. If you are generating traps for any deprecated managed objects, create new traps for the replacement objects.

   For a list of deprecated managed objects, see "Deprecated MBeans" in *WebLogic Server MBean Reference* at
   http://e-docs.bea.com/wls/docs100/wlsmbeanref/core/index.html. The description of each deprecated MBean includes a pointer to the replacement MBean. (Each SNMP managed object corresponds to an MBean attribute.)

> **Note:** A number of runtime MBeans that are internal to BEA have been removed from the MIB. These MBeans are not included in the deprecated MBeans list. For more information, see *WebLogic Server Known and Resolved Issues* at http://e-docs.bea.com/wls/docs100/issues/index.html.

Step 2: Re-apply Customizations to Startup Scripts

To complete the upgrade of your application environment to 10.0, you may need to re-apply any customizations to startup scripts. The following sections describe how to customize the default startup scripts as well as any custom startup scripts.

Default Startup Scripts

The Upgrade Wizard does not carry forward any customizations that have been made to the default startup scripts, such as the setting of the JAVA_OPTIONS environment variable. After the upgrade process is complete, you will need to customize the default scripts again.

If you are upgrading your domain to 10.0 and you want to continue using a pre-5.1 version of PointBase, you must add the JAR files for the pre-5.1 version of the PointBase database to the beginning of the CLASSPATH environment variable definition. To do so, update the set CLASSPATH statement in your setDomainEnv files.

Note: You can use the pre-5.1 version of PointBase only for WebLogic Server domains.

Custom Startup Scripts

If you have created custom startup scripts, you will need to update them manually, as follows:

- Set the JDK version to JDK 5.0.
- Update the CLASSPATH variable, as follows:
  - Add WebLogic Server 10.0 classes to the beginning of the variable.
  - Remove all *unused* pre-10.0 WebLogic classes.
  - If you want to continue using your pre-5.1 version of PointBase, include the pre-5.1 versions of the PointBase database JARs at the beginning of the CLASSPATH environment variable definition.

Step 3: Verify File Permissions

Verify the file permissions, as follows:

- If you backed up the domain directory as part of the upgrade, you now need to make your backup files secure, as they may contain confidential information.

- During the upgrade process, file permissions are not preserved. If non-default file permissions were set on files, they need to be verified and reset.

- On a UNIX system, ownership and permissions for any new files created during the upgrade process are assigned to the user performing the upgrade. For example, if the upgrade is performed by root, then root is assigned ownership of any new files. As a result, any user who subsequently wants to update these files in the domain must have root privileges. You may want to review and/or modify the permissions on files created during the upgrade process.

Step 4: Enroll the Machine with Node Manager

If you upgrade Node Manager during the upgrade process, enroll the machine that is hosting the WebLogic domain with Node Manager. This can be accomplished using the `nmEnroll` command.

Note: If the `nodemanager.domains` file resides on the machine where the Administration Server and Managed Server are configured to run and they share the same domain directory, you can manually edit the file to include an entry for the domain, in the following form: `<domain-name>=<domain-directory>`.

This file is located in `WL_HOME`/common/nodemanager, by default (where `WL_HOME` refers to the top-level installation directory for WebLogic Server).

For more information, see "Configuring nodemanager.domains File" in "Additional Configuration Information" in *Managing Server Startup and Shutdown* at http://e-docs.bea.com/wls/docs100/server_start/nodemgr.html#ReviewN MDomains.

The `nmEnroll` command updates the `nodemanager.domains` file under the `WL_HOME`/common/nodemanager directory with information about the domain, where `WL_HOME` refers to the top-level installation directory for WebLogic Server. The `nodemanager.domains` file specifies the domains that a Node Manager instance controls. This file is necessary so that standalone clients do not need to specify the domain directory explicitly.

This command also downloads the following files from the Administration Server:

- `nm_password.properties`, the Node Manager secret file, which contains the encrypted username and password that is used for server authentication

- `SerializedSystemIni.dat` file

To enroll the machine with Node Manager using the `nmEnroll` command:

1. Set up your environment, as described in "Setting Up Your Environment" in *WebLogic Scripting Tool* at
   http://e-docs.bea.com/wls/docs100/config_scripting/using_WLST.html#setting_up_your_environment.

2. Invoke WLST, as described in "Invoking WLST" in *WebLogic Scripting Tool* at
   http://e-docs.bea.com/wls/docs100/config_scripting/using_WLST.html#invoke_wlst.

   As described in step 2, start a WebLogic Server instance and connect WLST to the server using the `connect` command.

3. Once WLST is connected to the Administration Server, enter the `nmEnroll` command to enroll the machine on which WLST is running with Node Manager.

   You have the option of specifying:

   – Path of the domain directory in which you want to save the Node Manager secret file (`nm_password.properties`) and `SerializedSystemIni.dat` file. By default, these files are saved in the directory in which WLST was started.

   – Path of the Node Manager home directory. The `nodemanager.domains` file, containing the information about the domain, is written to this directory. By default, the directory used for this purpose is `WL_HOME`/common/nodemanager, where `WL_HOME` refers to the top-level installation directory for WebLogic Server.

   For example, if the domain directory is specified as `c:/bea/mydomain/common/nodemanager`, and the default home directory for Node Manager, `WL_HOME`/common/nodemanager, is used, then you enroll the machine on which WLST is running with Node Manager by entering the following command:

   ```
   wls:/mydomain/serverConfig>
   nmEnroll('c:/bea/mydomain/common/nodemanager')
   Enrolling this machine with the domain directory at
   c:\bea\mydomain\common\nodemanager....
   Successfully enrolled this machine with the domain directory at
   C:\bea\mydomain\common\nodemanager
   wls:/mydomain/serverConfig>
   ```

For more information, see `nmEnroll` in *WebLogic Scripting Tool* at
http://e-docs.bea.com/wls/docs100/config_scripting/reference.html#nmEnroll
.

Step 5: Verify Remote Server Startup Options

Once you start the Administration Server, verify the remote server start options, such as JAVA_HOME, BEA_HOME, and CLASSPATH, reference the WebLogic Server 10.0 installation on the target managed server. This can be accomplished using the Administration Console, as described in "Configure startup arguments for Managed Servers" in *Administration Console Online Help* at
http://e-docs.bea.com/wls/docs100/ConsoleHelp/taskhelp/startstop/Configure
StartupArgumentsForManagedServers.html.

WARNING: If the remote server startup options are not set correctly, when attempting to start a Managed Server using Node Manager, messages similar to the following may be written to the log file. Because these messages may be sent recursively, they may eventually consume all space available on the drive.

```
No config.xml was found.

Would you like the server to create a default configuration and boot?
(y/n):
java.io.IOException: The handle is invalid
at COM.jrockit.io.FileNativeIO.read(III)I(Native Method)
at COM.jrockit.io.NativeIO.read(Ljava.io.FileDescriptor;II)I(Unknown
Source)
at COM.jrockit.io.NativeIOInputStream.read(II)I(Unknown Source)
at COM.jrockit.io.NativeIOInputStream.read(I[BI)I(Unknown Source)
at COM.jrockit.io.NativeIOInputStream.read([BII)I(Unknown Source)
at java.io.FileInputStream.read([BII)I(Unknown Source)
```

Step 6: Promote the Application Environment to Production

Execute standard procedures for quality assurance and performance tuning before promoting an application environment to production. You should test the execution of your applications (including external client applications) in your test application environment. If your applications use APIs that have been deprecated or removed, then you may encounter warnings or exceptions at run time. If you do, you can make any required modifications before promoting your applications to production.

Once all test criteria have been met, you can promote the application environment to production, as outlined in your upgrade plan (defined previously in "Step 4: Create an Upgrade Plan" on page 2-5).

Once the new 10.0 application environment is deployed into production, you can start redirecting requests to the new environment from the existing environment. Gradually, you can quiesce the existing environment. This might be accomplished using a load balancer, for example.

What to Do If the Upgrade Process Fails

If any step in the upgrade process fails, the WebLogic Upgrade Wizard displays a message indicating the reason for the failure and terminates. To proceed, perform the following steps:

1. Restore the application environment to its original state using the backup files created in "Step 3: Back Up the Application Environment" on page 2-6.

2. Correct the failure reported by the WebLogic Upgrade Wizard.

3. Continue with the upgrade process from the step at which the failure occurred, as described in "Upgrade Your Application Environment" on page 2-8.

Upgrading a Security Provider

If you are using a custom security provider in a WebLogic Server 7.0 or 8.1 environment, you can use the WebLogic Upgrade Wizard to upgrade your security provider for use in a WebLogic Server 10.0 application environment.

Notes: As of 9.1, WebLogic Server includes two new security providers, the XACML Authorization provider and the XACML Role Mapping provider. Existing WebLogic domains that you upgrade to 10.0 will continue to use the authorization and role mapping providers currently specified, such as third-party partner providers or the original WebLogic Authorization and Role Mapping providers. If you wish, you can migrate existing domains from using WebLogic Server proprietary providers to the XACML providers, including performing bulk imports of existing policies. For more information, see *Security for BEA WebLogic Server 10.0* at `http://e-docs.bea.com/wls/docs100/security.html`.

The following sections describe how to use the WebLogic Upgrade Wizard for upgrading the Security Provider.

- What Happens During a Security Provider Upgrade

- Upgrading a Security Provider

For information about developing custom security providers, see *Developing Security Providers for WebLogic Server* at `http://e-docs.bea.com/wls/docs100/dvspisec/index.html`.

What Happens During a Security Provider Upgrade

For a security provider upgrade, you specify the source and destination directories for the upgrade, and the WebLogic Upgrade Wizard upgrades the existing JARs so that the security provider can run in a WebLogic Server 10.0 application environment.

Note: The security provider JAR must contain the appropriate MBean Definition File (MDF) that defines the MBean. An MDF is used to generate the .java files for a particular MBean type. For more information about creating MDFs, see *Developing Security Providers for WebLogic Server* at http://e-docs.bea.com/wls/docs100/dvspisec/index.html. If an MDF is not located in the JAR file, the upgrade process will fail for that specific security provider.

If an MDF contains undocumented tags, warnings will be generated during the upgrade process. These warnings will not affect the upgrade, and can be ignored. To avoid any further such warnings, however, you may want to remove undocumented tags from the MDF.

Security realms defined in pre-9.2 configurations must define a lockout manager (UserLockoutManagerMBean), and must conform to the following naming convention for JMX objects: Security:Name=*name*. Otherwise, the upgrade process will fail for the security provider.

During the upgrade, the Upgrade Wizard performs the following tasks:

- Adds required classes to the security provider JAR. These classes include MBeanImpl elements, schema files, and so on.

- Reads the MDF and creates the necessary schemas, MBean Implementation, and Binder classes.

- Stores the upgraded JARs for the security provider in the specified location.

- Appends _Upgraded to the security provider name to make the upgraded JARs distinct from the existing security provider JARs, which are maintained.

- Ignores security provider JARs that are shipped with BEA products, or JARs with names that contain _Upgraded, indicating that they have been upgraded already.

Upgrading a Security Provider

You must upgrade each custom security provider that you want to run in the WebLogic Server 10.0 environment.

Note: If you are installing WebLogic Server 10.0 into an existing BEA Home directory that contains an installation of WebLogic Server 7.0 or 8.1, all custom security providers that reside in the default location, `WL_HOME`\server\lib\mbeantypes, where `WL_HOME` specifies the root directory of the pre-10.0 installation, are upgraded automatically. If all of your custom security providers reside in the default location, then you do not have to perform the security provider upgrade step described in this section.

To verify that a custom security provider has been upgraded, locate the upgraded security provider, `security_provider_name_`Upgraded, in the `WL_HOME`\server\lib\mbeantypes directory, where `WL_HOME` specifies the root directory of the 10.0 installation, and `security_provider_name` specifies the name of the security provider.

You can upgrade a security provider using the WebLogic Upgrade Wizard in one of the following modes:

- **Graphical**—For upgrading a security provider interactively, using the graphical user interface.

- **Silent**—For upgrading a security provider silently, by specifying upgrade requirements in a file.

You must upgrade a security provider on each machine in the domain.

The following sections describe how to upgrade a security provider:

- Upgrading a Security Provider in Graphical Mode

- Upgrading a Security Provider in Silent Mode

Upgrading a Security Provider in Graphical Mode

The following sections describe how to upgrade a security provider by using the WebLogic Upgrade Wizard in graphical mode:

- Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade a Security Provider

- Procedure for Upgrading a Security Provider

Note: The console from which you are running the Upgrade Wizard in graphical mode must support a Java-based GUI. If you attempt to start the Upgrade Wizard in graphical mode on a system that cannot support a graphical display, the invocation fails and an error message is displayed.

Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade a Security Provider

Note: Before proceeding, make sure you have performed the prerequisite steps described in "Prepare to Upgrade" on page 2-5.

To start the WebLogic Upgrade Wizard in graphical mode and upgrade the security provider:

1. Verify that the WebLogic domain is not running.

2. Open an MS-DOS command prompt window (on Windows) or a command shell (on UNIX) and set up the environment as described in "Step 6: Set Up the Environment" on page 2-8.

3. At a command prompt, enter the following command:

   ```
   java weblogic.Upgrade -type securityproviders [-out file]
   ```

   The `-out` argument is optional. It allows you to designate a file in which you want all standard output (`stdout`) and error messages to be written. By default, these messages are written to the command window and a summary of them is displayed at the end of the upgrade process.

   After you run the command, the WebLogic Upgrade Wizard opens, as shown in the following figure.

Figure 3-1 WebLogic Upgrade Wizard for Security Providers
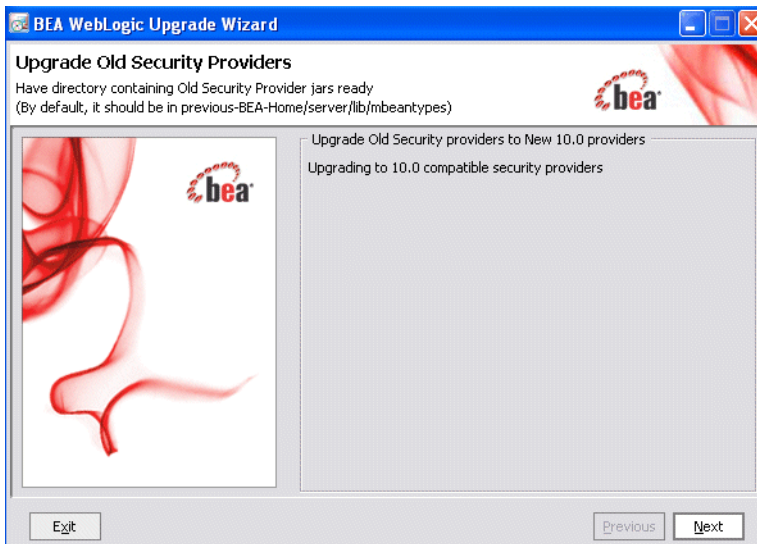
4. Click **Next** to proceed to the **Select Source Directory** window.

Procedure for Upgrading a Security Provider

The following table summarizes the steps in the procedure to upgrade a security provider using the WebLogic Upgrade Wizard.

Table 3-1 Procedure for Upgrading a Security Provider

In this step ...	You ...
Select Source Directory	Select the directory that contains the security provider JARs that need to be upgraded. By default, the selected directory is the current directory. By default, security providers are located in `WL_HOME\server\lib\mbeantypes`, where `WL_HOME` specifies the root directory of the pre-9.0 installation of WebLogic Server, as shown in the following example. **Figure 3-2 Select Directory Source From this Window** **Note:** The security provider JARs must contain the MBean Definition File (MDF) for the associated MBean. For more information about creating MDFs, see *Developing Security Providers for WebLogic Server* at `http://e-docs.bea.com/wls/docs100/dvspisec/index.html`. If JAR file does not contain an MDF, the upgrade process fails for the associated security provider. Click **Next** to proceed to the **Select Destination Directory** window.
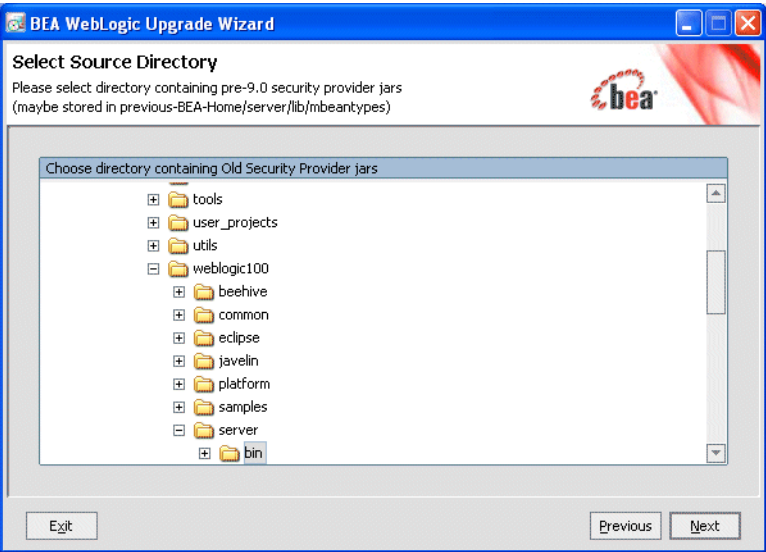
Table 3-1 Procedure for Upgrading a Security Provider (Continued)

In this step ...	You ...
Select Destination Directory	Select the directory in which you want to save the new security provider JAR files. The default directory is `WL_HOME\server\lib\mbeantypes`, where `WL_HOME` specifies the root directory of the WebLogic Server 10.0 installation. **Note:** To ensure the success of the domain upgrade, you must target the upgraded security providers to the default destination directory, `WL_HOME\server\lib\mbeantypes`. If you prefer to keep the security providers in a different location, you can move them once the domain upgrade process is complete. Click **Next** to proceed to the next window.

Upgrade Security Providers in Progress	Review progress of the wizard as it saves the upgraded JARs and deletes any temporary files that were created during the upgrade process. Progress messages are displayed in the window.
	
	The security provider JAR must contain the MBean Definition File (MDF) for the associated MBean. For more information about creating MDFs, see *Developing Security Providers for WebLogic Server* at http://e-docs.bea.com/wls/docs100/dvspisec/index.html. If a JAR file does not contain an MDF, the upgrade process fails for the associated security provider. For example:
	
	`Now processing mySecurityProviderToo.jar ...`
	
	`No MDFs (.xmls) found in the old security provider jar with name mySecurityProviderToo.jar`
	
	If an MDF contains undocumented tags, warnings are generated during the upgrade process. These warnings do not affect the upgrade; they can be ignored. To avoid further such messages, you may want to remove undocumented tags from the MDF.
	
	If the wizard locates a security provider JAR that was installed with the product, that has been upgraded already, or that is invalid, it does not upgrade that JAR. For example:
	
	`Not upgrading foo.txt because either this is a Out of the Box BEA Security Provider jar or this Security Provider jar is already upgraded or this is not a valid archive (may be not a .jar)`
	
	Click **Next** to proceed to the next window.

Upgrade Complete	Review the upgrade results, including any important messages that require further consideration. Click **Done** to close the wizard.

Upgrading a Security Provider in Silent Mode

In some circumstances, for example, when the security provider resides on a remote machine, it is not practical to use the WebLogic Upgrade Wizard in graphical mode. In such situations, you can use the wizard in silent mode to upgrade a security provider.

Note: Before proceeding, make sure you have performed the prerequisite steps described in "Prepare to Upgrade" on page 2-5.

To start the WebLogic Upgrade Wizard in silent mode and upgrade a security provider:

1. Verify that the WebLogic domain is not running.

2. Open an MS-DOS command prompt window (on Windows) or a command shell (on UNIX) and set up the environment as described in "Step 6: Set Up the Environment" on page 2-8.

3. (Optional) Create an XML script to define the upgrade requirements. For more information, see "Silent Upgrade XML Script Reference" on page E-1.

4. Navigate to the directory that contains the security provider that you want to upgrade.

5. At a command prompt, enter the following command:

   ```
   java weblogic.Upgrade -mode silent -type securityproviders [-responses
   xmlfile] [-out file]
   ```

   Two arguments are optional: `-responses` and `-out`. Include these arguments if you want to override the default values for the following:

   – The location of an XML file that defines the upgrade requirements. If you do not specify a file with the `-responses` option, the wizard uses the default values during the upgrade process. For more information about the format of the XML file and the default values, see "Silent Upgrade XML Script Reference" on page E-1.

   – The output file in which all standard output (`stdout`) and error messages are written. If you do not specify a file with the `-out` argument, these messages are written to the command window.

Upgrading Node Manager

If you are using a customized version of Node Manager in a pre-10.0 environment, you can use the WebLogic Upgrade Wizard to upgrade Node Manager for use in a WebLogic Server 10.0 application environment.

The following sections describe how to use the WebLogic Upgrade Wizard for this purpose:

- What Happens During a Node Manager Upgrade
- Upgrading Node Manager

Note: Before proceeding, make sure you have performed the prerequisite steps described in "Prepare to Upgrade" on page 2-5.

For more information about Node Manager, see "Using Node Manager to Control Servers" in *Managing Server Startup and Shutdown* at
`http://e-docs.bea.com/wls/docs100/server_start/nodemgr.html`.

What Happens During a Node Manager Upgrade

During a Node Manager upgrade, you specify the home directory of the Node Manager that you want to upgrade. The WebLogic Upgrade Wizard performs the following tasks:

- Upgrades Node Manager in the directory you specify so that Node Manager runs in a WebLogic Server 10.0 application environment. The Upgrade Wizard upgrades the `nodemanager.properties` file and converts the `NodeManagerSerializedSystemIni.dat` to `nm_data.properties`.

  Note: The nodemanager.properties file must be writable.

- Backs up existing log and state management files and stores them in a zip file, `weblogic-nodemanager-backup.zip`.

Any existing Node Manager files are overwritten during the upgrade process.

You are prompted to specify the username and password that will be used for Node Manager authorization when upgrading the domain. For more information, see "Enter Node Manager Credentials" on page 5-8. The listen port number used by Node Manager in the pre-10.0 installation is maintained during the upgrade process.

Note: When installing the 10.0 product, you should, if possible, set the Node Manager listen port to the same port number used in the pre-10.0 installation. The default listen port for Node Manager is 5556.

Once the upgrade process is complete, perform the following:

- Enroll the machine with Node Manager, as described in "Step 4: Enroll the Machine with Node Manager" on page 2-15.

- Verify that the username, password, and listen port settings for Node Manager are set as desired.

Upgrading Node Manager

You must upgrade each instance of Node Manager that you want to run in the WebLogic Server 10.0 environment. Specifically, you must upgrade Node Manager on every machine in the domain. You perform an upgrade using the WebLogic Upgrade Wizard in either of the following modes:

- **Graphical**—For upgrading Node Manager interactively, using the graphical user interface.

- **Silent**—For upgrading Node Manager silently, by specifying upgrade requirements in a file.

You must upgrade Node Manager on each machine in the domain.

The following sections describe how to upgrade Node Manager, including:

- Upgrading Node Manager in Graphical Mode

- Upgrading Node Manager in Silent Mode

Upgrading Node Manager in Graphical Mode

The following sections describe how to upgrade Node Manager using the WebLogic Upgrade Wizard in graphical mode:

- Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade Node Manager

- Procedure for Upgrading Node Manager

Note: The console from which you are running the Upgrade Wizard in graphical mode must support a Java-based GUI. If you attempt to start the Upgrade Wizard in graphical mode on a system that cannot support a graphical display, the invocation fails and an error message is displayed.

Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade Node Manager

Note: Before proceeding, make sure you have performed the prerequisite steps described in "Prepare to Upgrade" on page 2-5.

To start the WebLogic Upgrade Wizard in graphical mode and upgrade Node Manager:

1. Verify that the WebLogic domain is not running.

2. Open an MS-DOS command prompt window (on Windows) or a command shell (on UNIX) and set up the environment as described in "Step 6: Set Up the Environment" on page 2-8.

3. If the Node Manager directory resides in the pre-10.0 installation directory, for example, in the default location, *WL_HOME*/common/nodemanager (where *WL_HOME* specifies the root directory of the WebLogic Server installation), copy the contents of the Node Manager directory to the 10.0 installation directory.

   In this case, you will need to upgrade the *copy* Node Manager in the 10.0 installation directory.

   Note: Make sure you maintain the current directory structure. You do not need to copy the log (.log) files to the new location.

   If the Node Manager directory resides outside of the pre-10.0 installation directory, you can skip this step.

4. At a command prompt, enter the following command:

   ```
   java weblogic.Upgrade -type nodemanager [-out file]
   ```

The -out argument is optional. It allows you to designate a file in which you want all standard output (stdout) and error messages to be written. By default, these messages are written to the command window and a summary of them is displayed at the end of the upgrade process.

After you run the command, the WebLogic Upgrade Wizard opens, as shown in the following figure.

Figure 4-1 WebLogic Upgrade Wizard for Node Manager
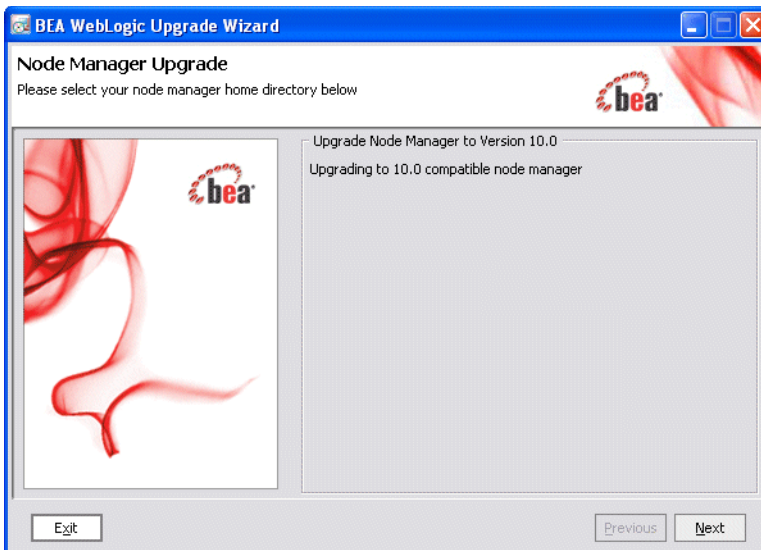
5. Click **Next** to proceed to the next window.

Procedure for Upgrading Node Manager

The following table summarizes the steps in the procedure to upgrade Node Manager using the WebLogic Upgrade Wizard.

Table 4-1 Procedure for Upgrading Node Manager

In this step ...	You ...
Node Manager Home Directory Selection	Select a home directory for the instance of Node Manager to be upgraded by navigating the local directory hierarchy. The Node Manager home directory contains the `nodemanager.properties` file, logs, and other related files. The exact set of files generated by Node Manager varies between releases and service packs. By default, the Node Manager home directory is `WL_HOME`/common/nodemanager, where `WL_HOME` specifies the root directory of the WebLogic Server installation, as shown in the following example. **Figure 4-2 Node Manager Home Directory Selection** Click **Next** to proceed to the next window.
Upgrade Your Node Manager Home	Review progress of the wizard as it saves the upgraded configuration and deletes any temporary files that were created during the upgrade process. Progress messages are displayed in the window. After the process is complete, click **Next** to proceed to the next window.
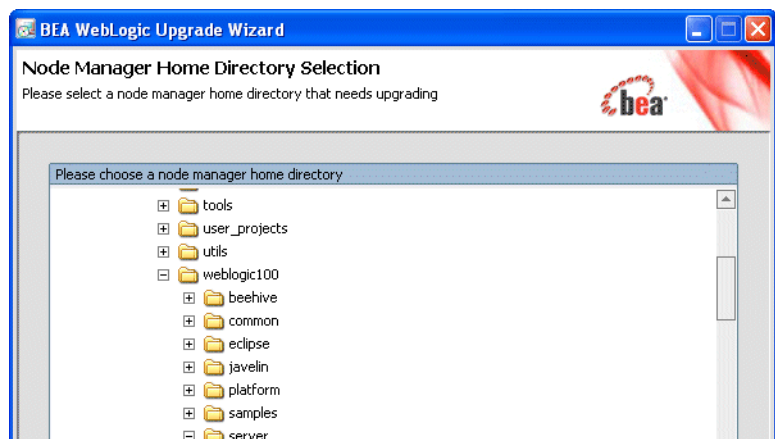
Table 4-1 Procedure for Upgrading Node Manager (Continued)

In this step ...	You ...
Upgrade Complete	Review the upgrade results, including any important messages that require further consideration.
	Click **Done** to close the WebLogic Upgrade Wizard.
	Note: Before using Node Manager, you must enroll the machine, as described in "Step 4: Enroll the Machine with Node Manager" on page 2-15. This step should be performed after you complete the WebLogic domain upgrade process.

Upgrading Node Manager in Silent Mode

In some circumstances, for example, when Node Manager resides on a remote machine, it is not practical to use the WebLogic Upgrade Wizard in graphical mode. In such situations, you can use the wizard in silent mode to upgrade Node Manager.

Note: Before proceeding, make sure you have performed the prerequisite steps described in "Prepare to Upgrade" on page 2-5.

To start the WebLogic Upgrade Wizard in silent mode and upgrade Node Manager:

1. Verify that Node Manager and all instances of WebLogic Server in the domain are not running.

2. Open an MS-DOS command prompt window (on Windows) or a command shell (on UNIX) and set up the environment as described in "Step 6: Set Up the Environment" on page 2-8.

3. If the Node Manager directory resides in the pre-10.0 installation directory, for example, in the default location, `WL_HOME`/common/nodemanager (where `WL_HOME` specifies the root directory of the WebLogic Server installation), copy the contents of the Node Manager directory to the 10.0 installation directory.

   In this case, you will need to upgrade the *copy* of Node Manager in the 10.0 installation directory.

   Note: Make sure you maintain the current directory structure. You do not need to copy the log (`.log`) files to the new location.

   If the Node Manager directory resides outside of the pre-10.0 installation directory, you can skip this step.

4. (Optional) Create an XML script to define the upgrade requirements. For more information, see "Silent Upgrade XML Script Reference" on page E-1.

5. Navigate to the Node Manager directory that you want to upgrade.

6. At a command prompt, enter the following command:

```
java weblogic.Upgrade -mode silent -type nodemanager [-responses xmlfile]
[-out file]
```

Two arguments are optional: `-responses` and `-out`. Include these arguments if you want to override the default values for the following:

– The location of an XML file that defines the upgrade requirements. If you do not specify a file with the `-responses` option, the wizard uses the default values during the upgrade process. For more information about the format of the XML file and the default values, see "Silent Upgrade XML Script Reference" on page E-1.

– The output file in which all standard output (`stdout`) and error messages are written. If you do not specify a file with the `-out` argument, these messages are written to the command window.

Note: Before using Node Manager, you must enroll the machine, as described in Before using Node Manager, you must enroll the machine, as described in "Step 4: Enroll the Machine with Node Manager" on page 2-15. This step should be performed after you complete the WebLogic domain upgrade process.

Upgrading Node Manager

Upgrading a WebLogic Domain

The 10.0 WebLogic Upgrade Wizard allows you to upgrade domains created in WebLogic Server 7.0 or 8.1.

You can also use the WebLogic Upgrade Wizard to upgrade a WebLogic domain created in WebLogic Server 9.0, 9.1, or 9.2 to 10.0, but this is optional. This type of domain runs under WebLogic Server 10.0 without modification. See Appendix A, "Upgrading WebLogic Server 9.0, 9.1, or 9.2 Application Environments to 10.0."

The following sections describe how to use the WebLogic Upgrade Wizard for this purpose:

- What Happens During a WebLogic Domain Upgrade?
- Important Notes About the Domain Upgrade Process
- Upgrading a Domain

What Happens During a WebLogic Domain Upgrade?

During a WebLogic domain upgrade, you specify the domain that you wish to upgrade and respond to a set of prompts. The WebLogic Upgrade Wizard performs the following tasks:

1. Optionally, the wizard backs up the original domain directory.

   If a backup is requested, the wizard backs up the domain directory only, and it does not preserve file permissions. BEA recommends that you back up the domain, any external applications, and application database resources in a separate process, as described in "Step 3: Back Up the Application Environment" on page 2-6.

Note: Backup files created by the wizard need to be protected by the user as they may contain confidential information.

2. Recreates scripts, such as startup and shutdown scripts, and renames any original scripts as `orig-scriptname`.bak, where `orig-scriptname` specifies the original script name and extension.

   Note: The wizard does not copy any customizations in the original startup scripts to the new scripts. For example, if you specified a non-default value for the `JAVA_OPTIONS` environment variable in the original script, the specified value will not be preserved in the new script.

3. Restructures the original domain, creating a new directory structure and moving domain components to new locations.

   During the restructuring, if a required directory already exists, the wizard simply keeps that directory and maintains the files and subdirectories that reside in it.

   Existing server log files are copied to the `servers/server_name/logs/pre-10.0-logs` directory in the domain, where `server_name` specifies the name of the server.

   To review changes to the domain directory structure, see "WebLogic Domain Directory Structure Enhancements" on page C-1.

4. Upgrades the persisted configuration information stored in the configuration file (`config.xml`) to the `config` directory.

   If the wizard encounters duplicate resources when upgrading the configuration file (`config.xml`), a message is logged in the progress window. In this case, the last resource definition encountered is used during the conversion.

5. Upgrades persisted data, such as JMS file stores, JMS JDBC stores, and transaction stores.

   Note: If JMS JDBC stores are used in the domain, see "Step 6: Set Up the Environment" on page 2-8.

   After the JMS JDBC stores are upgraded, the original JMS JDBC stores are not deleted. You should take this fact into account when performing capacity planning. You can delete the original JMS JDBC store tables once the upgrade is successful. Original JMS JDBC store tables are named `PrefixNameJMSSTORE` and `PrefixNameJMSSTATE`, where `PrefixName` is the value of the `Prefix Name` attribute for the JMS JDBC store.

   If you do not want to upgrade persisted JMS messages, you can delete the JMS file store or JMS JDBC store tables before running the upgrade. When you do so, only JMS messages are lost; the configuration is not changed. For information about managing JDBC store tables, see "Managing JDBC Store Tables" in *Configuring WebLogic Server Environments*

at
http://e-docs.bea.com/wls/docs100/config_wls/store.html#managing_jdbc_store_tables.

The wizard does not upgrade a JMS JDBC or file store if it detects that an upgrade has already been performed. If you need to perform multiple upgrades of a domain in which the same persistent stores are use (for example, in a test scenario), you must revert the data in the JMS store each time you repeat the upgrade process, as follows:

– For a JMS JDBC store, the upgrade process creates a new table named *PrefixNameWLSTORE*, where *PrefixName* is the value of the `Prefix Name` attribute for the JMS JDBC store. Before re-running the upgrade process on a domain that uses the JMS JDBC store is used, to drop this table.

– If you need to re-run the upgrade, make sure you first restore the backed up version of the JMS file store.

6. Saves the configuration.

   Note: When upgrading remote Managed Servers, the wizard does not persist the configuration information.

7. Reports any issues with the domain upgrade that require further consideration.

Important Notes About the Domain Upgrade Process

Please note the following important notes about the upgrade process:

- WebLogic Server applications do not need to be undeployed. In most cases, WebLogic Server applications can be run without modifications in the new WebLogic Server 10.0 application environment. Review the compatibility information in "WebLogic Server 10.0 Compatibility with Previous Releases" on page B-1 to determine whether any features changes affect the applications in your environment. Note that if APIs that have been deprecated or removed are used in the application, then you may encounter warnings or exceptions at run time.

- At a minimum, the domain directory must contain the following files:

  – config.xml

  – Security-related files, including `SerializedSystemIni.dat`, `DefaultAuthenticatorInit.ldift`, `DefaultAuthorizerInit.ldift`, and `DefaultRoleMapperInit.ldift`

If the security-related files are not available, the server fails to start and an authentication error message is logged.

– Any transaction log (`.tlog`) files that reside in the domain. For more information, see "Transaction Log Files" in *Programming WebLogic JTA* at http://e-docs.bea.com/wls/docs100/jta/trxcon.html#tlog.

- All contents of the domain directory on the target machine are updated during this process.

- You must upgrade the domain on every machine in the application environment.

- The wizard does not upgrade applications during a WebLogic domain upgrade.

- Domains that contain resources for WebLogic Liquid Data, or AquaLogic Data Services Platform cannot be upgraded to WebLogic 10.0.

- As of 9.0, the `.wlnotdelete` directory is no longer used in the WebLogic Server environment.

- During the upgrade process, file permissions are not preserved. All non-default file permissions need to be verified and reset.

- On a UNIX system, ownership and permissions for any new files created during the upgrade process are assigned to the user performing the upgrade. For example, if the upgrade is performed by root, then root is assigned ownership of any new files. As a result, any user who subsequently wants to update these files in the domain must have root privileges. You may want to review and/or modify the permissions on files created during the upgrade process.

- In WebLogic Server 10.0, you may need to specify a username and password when starting the server. As a result, you may notice the following behavioral changes:

  – When upgrading the WebLogic product samples that were installed automatically for WebLogic Server 7.0 releases (located in `WL_HOME`\samples\server\config), you may notice a change in behavior when starting the Administration Server in the upgraded 10.0 domain. With 10.0, you are prompted for a username and password. With 7.0, the username and password are passed as a command-line argument to the `startServer` command, as follows:

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

- When you upgrade a domain on a remote Managed Server, a message similar to the following may be displayed to indicate that the referenced application path does not reside on the system:

```
<Aug 1, 2005 6:42:06 PM EDT> <INFO> <Upgrade> <BEA-800000> <An invalid
path, 'C:\bea\wlserver_10.0\user_projects\mydomain\medrecEar.ear', was
specified for application, 'medrecEar'.>
```

This message can be ignored.

- If you upgraded the Avitek Medical Records application from 8.1 to 10.0 on a Solaris machine (only), before starting the server, you must edit the `setDomainEnv.sh` file to remove `-Xverify:none` from the start command by setting `JAVA_OPTIONS=""` after the following line:

  `. ${WL_HOME}/common/bin/commEnv.sh`

  Otherwise, the server start will fail with a JVM error.

Upgrading a Domain

The wizard supports the following upgrade modes:

- **Graphical**—For upgrading a domain interactively, the Domain Upgrade Wizard using a graphical user interface.

- **Silent**—You can upgrade a WebLogic Server domain silently by specifying upgrade requirements in a file.

Note: You can also use implicit mode to upgrade a WebLogic Server domain automatically when the Administration Server is started. For more information, see "Upgrading a Domain at Administration Server Startup (Implicit Mode)" on page F-1.

You must upgrade the domains on every machine in the domain. For information about preparing remote managed server domain directories, see "Step 5: Prepare the Remote Managed Server Domain Directories" on page 2-8.

Note: Before proceeding, make sure you have:

- Performed the prerequisite steps described in "Prepare to Upgrade" on page 2-5.

- Reviewed the important notes described in "Important Notes About the Domain Upgrade Process" on page 5-3.

- Ensured that the WebLogic Domain is not running.

- Backed-up the JMS Store.

The following sections provide instructions for:

- Upgrading a Domain in Graphical Mode

● Upgrading a Domain in Silent Mode

Upgrading a Domain in Graphical Mode

The following sections describe how to upgrade a WebLogic domain using the WebLogic Upgrade Wizard in graphical mode:

● Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade a Domain

● Procedure for Upgrading a WebLogic Domain

Note: The console from which you are running the Upgrade Wizard in graphical mode must support a Java-based GUI. If you attempt to start the Upgrade Wizard in graphical mode on a system that cannot support a graphical display, the invocation fails and an error message is displayed.

Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade a Domain

To start the WebLogic Upgrade Wizard in graphical mode and upgrade a WebLogic domain on a Windows platform, choose the Domain Upgrade Wizard option from the BEA program group in the Windows Start Menu:

Start −> Programs −> BEA Products−> Tools −> Domain Upgrade Wizard

Note: You can only use this option if you **do not** have to customize the environment to specify JDBC driver classes, as described in step 3 of "Step 6: Set Up the Environment" on page 2-8.

To start the WebLogic Upgrade Wizard in graphical mode and upgrade a WebLogic domain from a Windows command prompt or on a UNIX platform:

1. Verify that the WebLogic domain is not running.

2. Backup the JMS Store, if applicable.

3. Open a command prompt window (on Windows) or a command shell (on UNIX) and set up the environment as described in "Step 6: Set Up the Environment" on page 2-8.

4. Execute the following script to upgrade your domains.

   – On Windows: *WL_HOME*\common\bin\upgrade.cmd

   – On Unix: *WL_HOME*/common/bin/upgrade.sh

   The log file will be available in the *BEA_HOME*/user_projects/upgrade_logs directory.

The following command can also be used to upgrade a WebLogic Server domain.

```
java weblogic.Upgrade [-type domain] [-out file]
```

Two arguments are optional: `-type` and `-out`. Include these arguments if you want to override the default values for the following:

– The type of upgrade to be performed. If you do not specify a type with the `-type` option, a domain upgrade is performed.

– The output file in which all standard output (`stdout`) and error messages are written. If you do not specify a file with the `-out` option, such messages are written to the command window, and a summary of messages is displayed at the end of the upgrade process.

The `-out` argument is optional. It allows you to designate a file in which you want all standard output (`stdout`) and error messages to be written. By default, these messages are written to the command window and a summary of them is displayed at the end of the upgrade process.

After you run the command, the WebLogic Upgrade Wizard opens, as shown in the following figure.
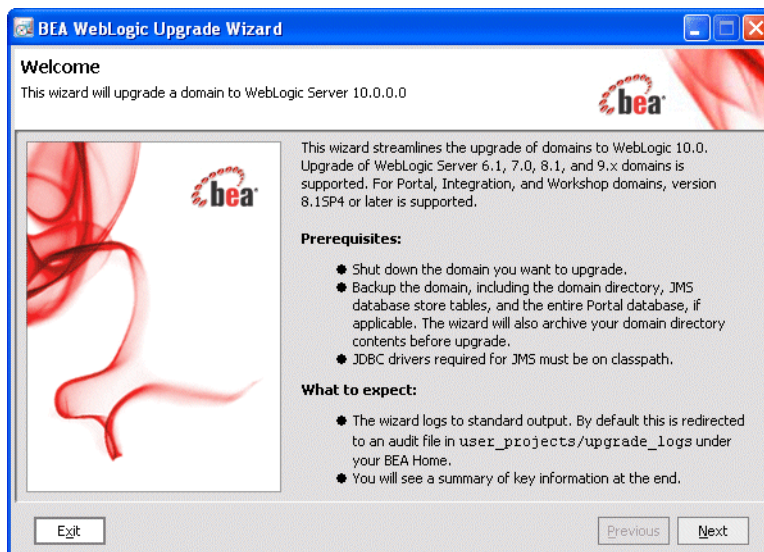
5. If JMS JDBC stores are used, ensure the corresponding database is running. Note that PointBase databases are automatically started and shut down by the Domain Upgrade Wizard.

6. Click **Next** to proceed to the next window.

Procedure for Upgrading a WebLogic Domain

The following table summarizes the steps in the procedure to upgrade a domain using the WebLogic Upgrade Wizard.

Note: The screens provided in this section are only indicative of what you might see. The actual screens depend on the combination of resources used in your domain.

Table 5-1 Procedure for Upgrading a WebLogic Domain

In this step ...	You ...
Select WebLogic Version	Select the WebLogic version of the domain that you are upgrading.
	Click **Next** to proceed to the next window.
Select a Domain to Upgrade	Select the directory that contains the WebLogic domain to be upgraded by navigating the local directory hierarchy.
	Click **Next** to proceed to the next window.
Inspect Domain	Review progress of the wizard as it inspects the domain. Progress messages are displayed in the window.
	If you attempt to upgrade a domain in which custom security providers are used, without first upgrading those security providers, an error message is displayed and the wizard exits.
	If you receive this error message, upgrade the customer security providers, as described in "Upgrading a Security Provider" on page 3-1, and start the domain upgrade procedure again.
	Once the inspection is complete (and if no error is encountered), the wizard advances to the next window automatically.
Select Administration Server	Select a server to function as the Administration Server in the new domain.
	Note: If there is only one server defined in the domain, this window is skipped. This window is displayed only if the domain you are upgrading has multiple servers.
	Click **Next** to proceed to the next window.
Enter Node Manager Credentials	Enter the username and password (and password confirmation) for Node Manager authorization.
	For WebLogic Server 10.0, Node Manager requires user and password credentials to be specified for each domain. By default, the username and password are set to weblogic. If you do not use Node Manager, leave the default values unchanged.
	Click **Next** to proceed to the next window.

Table 5-1 Procedure for Upgrading a WebLogic Domain (Continued)

In this step ...	You ...
Select Upgrade Options	• Back up current domain (recommended)—If selected, the wizard backs up the original domain directory and stores it in a zip file. This option is selected by default.
	Note: The wizard backs up the domain directory only and does not preserve file permissions. BEA recommends that you back up the domain and any external application and application database resources in a separate process, as described in Step 3: Back Up the Application Environment.
	• Add log files to backup zip—If selected, log files will be included in the backup zip file. The number and size of log files can be large and you may want to disable this option to exclude them from the backup file. By default, log files are included in the backup file.
	• Do not set backwards compatibility flags—As of WebLogic Server 9.0, some previously supported behavior has changed to comply with J2EE 1.4. By default, the wizard sets flags to enable the previous behavior in the new domain. If you select this option, these flags are not set for backward compatibility. For more information about the backward compatibility flags, see Backward Compatibility Flags.
Directory Selection Analysis and Optional Tasks	Review progress as the wizard processes the domain information and options provided. Progress messages are displayed in the window. Once processing is complete, the wizard advances automatically to the next window.
Domain Backup	Review progress of the wizard as it prepares to back up the domain. Progress messages are displayed in the window. Once processing is complete, the wizard advances automatically to the next window.
Select Directory for Domain Backup	**Note:** If you did not select the **Back up current domain** upgrade option, as described in "Select Upgrade Options" on page 5-9, skip to "Restructure Domain Directory" on page 5-10. In this window, set values for the following: • Backup directory — Navigate the local hierarchy and select the directory in which you want to save the backup zip file. By default, the original domain directory is used. • Backup filename—Enter the name of the backup file in the text box. The default filename is `weblogic-domain-backup-domain.zip`, where *domain* specifies the name of the domain. Click **Next** to proceed to the next window.

Table 5-1 Procedure for Upgrading a WebLogic Domain (Continued)

In this step ...	You ...
Backup Domain	Review progress as the wizard backs-up the domain. A progress bar displays the percentage of the backup process that is complete, and progress messages are displayed in the window. **Note:** Backup files created by the wizard need to be protected by the user as they may contain confidential information. Once the backup process is complete, the wizard advances automatically to the next window.
Restructure Domain Directory	Review progress as the wizard restructures the domain directory. Progress messages are displayed in the window. Once the process is complete, the wizard automatically advances to the next window.
Upgrade Configuration Settings	Review progress as the wizard upgrades the configuration settings. Progress messages are displayed in the window. The configuration information is not persisted until a later step. Once the configuration upgrade is complete, the wizard advances automatically to the next window.
Upgrade Persisted Messages and Transaction Log Formats	Review progress as the wizard upgrades the persisted messages (JMS file and JDBC stores) and transaction (JTA) logs that exist in the domain. A progress bar displays the percentage complete and progress messages are displayed in the window. Once the persisted message and transaction log upgrade process is complete, the wizard advances to the next window automatically.
Upgrade RDBMS Authenticator Security Provider	Specify whether or not the deprecated RDBMSAuthenticator should be replaced by the SQLAuthenticator. **Note:** This window appears only when an RDBMSAuthenticator Security Provider exists in the domain you are upgrading. Click **Next** to proceed to the next window.
Persist Upgraded Configuration	Review progress of the wizard as it saves the upgraded configuration and deletes any temporary files that were created during the upgrade process. Progress messages are displayed in the window. **Note:** When upgrading remote Managed Servers, the wizard does not persist the configuration information. Once this process is complete, click **Next** to proceed to the next window.

Table 5-1 Procedure for Upgrading a WebLogic Domain (Continued)

In this step ...	You ...
Upgrade Complete	Review the results of the upgrade, including any important messages that require further consideration.
	Click **Done** to close the WebLogic Upgrade Wizard.

Upgrading a Domain in Silent Mode

Note: Only WebLogic Server domains can be upgraded in the Silent mode.

In some circumstances, for example, when the domain resides on a remote machine, it is not practical to use the WebLogic Upgrade Wizard in graphical mode. In such situations, you can use the wizard in silent mode to upgrade the WebLogic domain.

Note: Before proceeding, make sure you have performed the prerequisite steps described in "Prepare to Upgrade" on page 2-5.

To start the WebLogic Upgrade Wizard in silent mode and upgrade a WebLogic domain:

1. Verify that the WebLogic domain is not running.

2. Open an MS-DOS command prompt window (on Windows) or a command shell (on UNIX) and set up the environment as described in "Step 6: Set Up the Environment" on page 2-8.

3. (Optional) Create an XML script to define the upgrade requirements. For more information, see "Silent Upgrade XML Script Reference" on page E-1.

4. Navigate to the directory that contains WebLogic domain that you want to upgrade. For example:

   ```
   cd c:\bea\user_projects\domains\domain
   ```

   where *domain* specifies the name of the domain.

5. At a command prompt, enter the following command:

   ```
   java weblogic.Upgrade -mode silent -type domain [-responses xmlfile]
   [-out file]
   ```

   The following arguments are optional: `-responses` and `-out`. Include these arguments if you want to override the default values for the following:

   – The location of an XML file that defines the upgrade requirements. If you do not specify a file with the `-responses` option, the wizard uses the default values during

the upgrade process. For more information about the format of the XML file and the default values, see "Silent Upgrade XML Script Reference" on page E-1.

– The output file in which all standard output (stdout) and error messages are written. If you do not specify a file with the -out argument, these messages are written to the command window.

Upgrading WebLogic Server 9.0, 9.1, or 9.2 Application Environments to 10.0

The following sections describe the procedures for upgrading application environments from WebLogic Server 9.0, 9.1, or 9.2 to the WebLogic Server 10.0 release:

- "Creating a New Domain"

- "Updating an Existing Domain"

- "Upgrading Beehive Applications"

Note: BEA does not recommend upgrading an application environment that is currently deployed in production. Instead, you should upgrade your application environment while it is under development or test and execute standard procedures for quality assurance and performance tuning before promoting the upgraded environment to production.

Creating a New Domain

This option is useful if the process of creating and customizing a domain has been automated. The steps to create a new domain are as follows:

1. Install WebLogic Server 10.0 software.

2. To create your domain in 9.*x*, use a default domain template provided in 10.0. Alternatively, if you used a custom 9.*x* template to create your domain in 9.*x*, use the same to create a new 10.0 domain.

   This step can be accomplished using the Configuration Wizard or using automated scripts, built using WebLogic scripting tools such as WLST.

You may need to update your automated scripts, for example, to reference the 10.0 domain template and/or to implement new features provided in the 10.0 release.

For example, starting in WebLogic 9.1, the default security providers are XACML-based, as described in Security for BEA WebLogic Server 10.0. You may choose to add support for the security providers prior to WebLogic 9.1 or make the appropriate changes to use the new XACML-based security providers.

3. Deploy existing WebLogic Server 9.*x* applications to the new 10.0 domain.

   Note: If you used a custom 9.*x* template in step 2, the 9.*x* applications may already be deployed.

Updating an Existing Domain

Updating an existing domain is useful for maintaining customizations within your test domain if generation of the domain has not been automated. There are two options for upgrading an existing domain, as follows:

- Option One: Manual Update

   a. Install WebLogic Server 10.0 software.

   b. Back up the existing application environment, including the domain directory, application and application data that is external to the domain, and log files (if necessary).

   c. Update the script files in the domain to point to the installation of WebLogic Server 10.0. For example, set BEA_HOME, BEA_JAVA_HOME, JAVA_HOME, and WL_HOME to the appropriate values.

   d. Update the CLASSPATH to remove path information that is no longer required, such as patch file information that applies to a pre-10.0 release.

- Option two: Automated Update

   a. Use the Domain Upgrade Wizard, as described in Chapter 5, "Upgrading a WebLogic Domain."

Upgrading Beehive Applications

If you developed Beehive applications in 9.0, 9.1 or 9.2, you must upgrade the applications as described in Upgrade Paths in *Beehive Integration in WebLogic Server 10.0*.

WebLogic Server 10.0 Compatibility with Previous Releases

This section describes important compatibility information that you should consider before upgrading to WebLogic Server 10.0.

 Also see, *Compatibility Statement for BEA WebLogic Server 10.0.*

Compatibility considerations are provided in the following categories:

- JMX 1.2 Implementation

- Dynamic Configuration Management

- Modular Configuration and Deployment of JDBC Resources

- JDBC Debugging Enhancements

- Modular Configuration and Deployment of JMS Resources

- JMS Message ID Format

- Improved Message Paging

- Thread Management

- JTA Transaction Log Migration

- Security

- Web Services

- Web Applications, JSPs, and Servlets

- XML Implementation

- XMLBeans and XQuery Implementation

- WebLogic Administration and Configuration Scripts

- Deployment Descriptor Validation and Conversion

- Deprecated Startup and Shutdown Classes

- Administration Console Extension Architecture

- Resource Adapters

- WLEC

- SNMP MIB Refresh Interval and Server Status Check Interval No Longer Used

- Backward Compatibility Flags

- Deprecated and Removed APIs

JMX 1.2 Implementation

As of WebLogic Server 9.0, WebLogic Server uses the Java Management Extensions (JMX) 1.2 implementation that is included in JDK 5.0. Prior to 9.0, WebLogic Server used its own JMX implementation based on the JMX 1.0 specification.

The JMX 1.2 reference implementation introduces serialization incompatibilities. Despite these incompatibilities in the reference implementation, JMX clients created for WebLogic Server 8.1 can be used with 9.2 and 10.0 as follows:

- If your JMX client accesses only WebLogic Server MBeans and uses only `weblogic.management.MBeanHome`, it can be run in a WebLogic Server 9.2 or 10.0 instance without being upgraded.

- A JMX client in which WebLogic Server 8.1 classes are used can interact with 9.2 or 10.0 JMX agents if all of the following are true:

  – The client accesses only WebLogic Server MBeans.

  – The client uses only `weblogic.management.MBeanHome`; it does not use the JDK `MBeanServer` interface.

  – The WebLogic Server classes are from 8.1 SP4 or 7.0 SP6 with any appropriate patches applied.

- If the standard JMX MBeanServer interface is used in your JMX client, either to interact with WebLogic Server MBeans or to create and access custom MBeans, you must include the following JDK startup option for the WebLogic Server 9.2 or 10.0 instance: `-Djmx.serial.form=1.0`

  This startup option causes the JVM to use JMX 1.0 class descriptions when it is serializing objects. The option is required when JMX 1.0 clients communicate with JMX 1.2 agents using the standard JDK classes.

- If your JMX client interacts with security provider MBeans, see "Security MBeans" on page B-12.

BEA recommends that you update your JMX clients to be compliant with WebLogic Server 10.0. Prior to 9.0, WebLogic Server supported a typed API layer over its JMX layer. It was possible for your JMX application classes to import type-safe interfaces for WebLogic Server MBeans, retrieve a reference to the MBeans through the `weblogic.management.MBeanHome` interface, and invoke the MBean methods directly.

As of 9.0, the `MBeanHome` interface is deprecated. Instead of using this API-like programming model, all JMX applications should use the standard JMX programming model, in which clients use the `javax.management.MBeanServerConnection` interface to discover MBeans, attributes, and attribute types at runtime. In this JMX model, clients interact indirectly with MBeans through the `MBeanServerConnection` interface.

If any of your classes import the type-safe interfaces (available under `weblogic.management`), BEA recommends that you update them to use the standard JMX programming model. For more information, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities with JMX* at
http://e-docs.bea.com/wls/docs100/jmx/understandWLS.html.

Dynamic Configuration Management

Configuration attributes are classified as *dynamic* or *non-dynamic*.

- Changes to dynamic configuration attributes are available as soon as they are activated, without restarting the affected server or system resource. These changes are made available to the server and runtime hierarchies once they are activated.

- Changes to non-dynamic configuration attributes are not immediately available. When a non-dynamic configuration attribute is changed, the server or system resource must be restarted to make the change effective.

WebLogic Server 9.0 introduced a change management process to provide a secure, predictable means for applying configuration changes in a domain. A batch change mechanism changes the way dynamic changes are applied when they are mixed with non-dynamic changes. Specifically, when a configured server or system resource is affected by a change to a non-dynamic attribute, no other changes (even dynamic changes) will take effect, in current or future batches, until after the server or system resource is restarted. In this case, BEA recommends that you restart the entity as soon as possible after the batch change is completed to ensure the system is in a consistent state and to allow future changes to be accepted.

You should test your configuration scripts to determine whether a non-dynamic change has been applied, and if so, restart the server. To determine whether a change is non-dynamic and requires a server restart:

- Prior to activating a change you can:
  - View the change listed in the Change Center in the Administration Console, as described in "Dynamic and Non-Dynamic Changes" in "Using the Change Center" in "Overview of the Administration Console" in *Introduction to WebLogic Server* at http://e-docs.bea.com/wls/docs100/intro/console.html.
  - Use the following WLST commands: `isRestartRequired` or `showChanges`. For more information, see "WLST Command and Variable Reference" in *WebLogic Scripting Tool* at http://e-docs.bea.com/wls/docs100/config_scripting/reference.html.

- After you activate a change you can:
  - Review the server log to identify whether the change is categorized as non-dynamic.
  - Check the value of the `RestartRequired` or `PendingRestartSystemResources` attribute that is associated with the changed object, if applicable.

To determine which security attributes are dynamic or non-dynamic, see "Security Configuration MBeans" in *Securing WebLogic Server* at http://e-docs.bea.com/wls/docs100/secmanage/mbeans.html.

For more information, see "Managing Configuration Changes" in *Understanding Domain Configuration* at http://e-docs.bea.com/wls/docs100/domain_config/changes.html.

Modular Configuration and Deployment of JDBC Resources

As of WebLogic Server 9.0, the number of JDBC resource types was reduced to simplify JDBC configuration and to reduce the likelihood of configuration errors. Now, instead of configuring a JDBC connection pool and then configuring a data source or tx data source to point to the connection pool and bind to the JNDI tree, you can configure a data source that encompasses a connection pool. For more information about simplified JDBC resource configuration, see "Simplified JDBC Resource Configuration" in *Configuring and Managing WebLogic JDBC* at `http://e-docs.bea.com/wls/docs100/jdbc_admin/jdbc_intro.html#simple_res_config`.

The WebLogic Upgrade Wizard automatically converts JDBC data sources, connection pools, MultiPools, and data source factories to their new counterparts in WebLogic Server 10.0, as described in the following sections.

Note: Each upgraded JDBC module contains an internal properties section. WebLogic Server uses internal properties to manage the data sources for backward compatibility. Also, some legacy attributes are preserved as properties in the Properties attribute of the JDBC data source file. Do not manually edit any internal properties.

For information about deprecated JDBC features, methods, interfaces, and MBeans, see "Deprecated JDBC Features, Methods, Interfaces, and MBeans" in *WebLogic Server and WebLogic Express Release Notes* at `http://e-docs.bea.com/wls/docs100/notes/new.html#deprecated_jdbc_features`.

JDBC Data Sources and Connection Pools

The Upgrade Wizard converts legacy JDBC data source/connection pool pairs to two data source system resource modules, one for the data source and one for the connection pool:

- The data source that replaces the existing data source or tx data source defines the data source parameters and refers to the second data source for its connection pool and related attributes.

- The data source that replaces the connection pool contains the JDBC driver parameters, the connection pool parameters, and the XA parameters.

  Note: Only data sources that are converted as part of a domain upgrade can refer to another data source for its connection pool. In all other cases, each data source contains its own pool of database connections.

During an upgrade, the Upgrade Wizard sets the `GlobalTransactionsProtocol` parameter for a data source based on the type of data source being converted (tx or non-tx) and the type of driver used in the related connection pool, as noted in the following table.

Table B-1 Parameter Settings for Global Transaction Protocol Parameter Setting

Legacy Data Source Type	Driver Type	Emulate Two-Phase Commit	GlobalTransactionProtocol
Tx Data Source	XA	N/A	`TwoPhaseCommit`
Tx Data Source	Non-XA	False	`OnePhaseCommit` (by default; not explicitly set)
Tx Data Source	Non-XA	True	`EmulateTwoPhaseCommit`[1]
Data Source	Non-XA	N/A	None

1. Depending on your environment, you may want to consider using the `LoggingLastResource` (LLR) transaction protocol in place of the `EmulateTwoPhaseCommit` protocol for transaction processing because of its performance benefits. For more information see "Understanding the Logging Last Resource Transaction Option" in *Configuring and Managing WebLogic JDBC* at http://e-docs.bea.com/wls/docs100/jdbc_admin/jdbc_datasources.html#llr.

MultiPools

The Upgrade Wizard converts a MultiPool to a multi-data source, which is another instance of a data source object that provides load balancing and/or failover between data sources.

Data Source Factories

Data source factories are deprecated in this release and are included for backward compatibility only. No conversion of data source factories is required.

JDBC Debugging Enhancements

The JDBC subsystem uses the new system-wide WebLogic Diagnostic Service for centralized debugging access and logging.

See Using the WebLogic Diagnostic Framework Console Extension.

Modular Configuration and Deployment of JMS Resources

As of WebLogic Server 9.0, JMS configurations are stored as modules, defined by XML documents that conform to the new `weblogic-jmsmd.xsd` schema. With modular deployment of JMS resources, you can promote your application and the JMS configuration from one environment to another. For example, you can promote your application and the required JMS configuration from a testing environment to a production environment, without opening an EAR file and without extensive manual JMS re-configuration.

For more information, see:

- "New and Changed JMS Features in This Release" in *Configuring and Managing WebLogic JMS* at
  `http://e-docs.bea.com/wls/docs100/jms_admin/intro.html#WhatsNewJMS`.

- "Understanding JMS Resource Configuration" in *Configuring and Managing WebLogic JMS* at `http://e-docs.bea.com/wls/docs100/jms_admin/overview.html`.

- "Deploying JDBC, JMS, and WLDF Application Modules" in *Deploying Applications to WebLogic Server* at
  `http://e-docs.bea.com/wls/docs100/deployment/deploy.html#deploy_resourc es`.

The WebLogic Upgrade Wizard automatically converts pre-10.0 JMS resources to a JMS Interop module file named `interop-jms.xml`, which is copied to the domain's `config\jms` directory. For more information, see "JMS Interop Modules" in *Configuring and Managing WebLogic JMS* at
`http://e-docs.bea.com/wls/docs100/jms_admin/overview.html#jms_interop_modu les`.

Please note the following JMS configuration changes:

- When generating new JMS resources, you must define all attributes in the JMS module (that is, not using the pre-10.0 configuration file).

- The `Allow Persistent Downgrade` option enables you to specify whether JMS clients receive an exception when they send persistent messages to a destination targeted to a JMS server that does not have a persistent store configured. This option is provided for backward compatibility with previous releases.

  By default, the option is set to `false` specifying that clients will receive an exception when they send persistent messages to a JMS server for which no store is configured. When the option is set to `true`, persistent messages are downgraded to non-persistent, but, the send

operations are allowed to continue. This parameter is effective only when the `Store Enabled` parameter is disabled (that is, when it is set to `false`).

For more information, see "AllowsPersistentDowngrade" in "JMSServerBean" in *WebLogic Server MBean Reference* at `http://e-docs.bea.com/wls/docs100/wlsmbeanref/mbeans/JMSServerMBean.html#AllowsPersistentDowngrade`.

- A `Temporary Template` is created, by default, for JMS Servers. In previous releases, no default template was provided. You can also configure a temporary template, using the JMS server's `Temporary Template` attribute.

  You can control whether the JMS Server can host a temporary destination by setting the `Hosts Temporary Destinations` attribute. In previous releases, a JMS Server was enabled to host temporary destinations if and only if the `TemporaryTemplate` attribute was set.

- JMS templates specified for distributed destinations are no longer supported as of WebLogic Server 9.0, and they will be ignored. As of WebLogic Server 9.0, this functionality is replaced by uniform distributed destinations. For more information, see "Creating Uniform Distributed Destinations" in "Configuring Distributed Destinations" in *Configuring and Managing WebLogic JMS* at `http://e-docs.bea.com/wls/docs100/jms_admin/advance_config.html#jms_distributed_destination_create`.

- The `AllowCloseInOnMessage` attribute for JMS Connection Factories is enabled by default. For more information, see `ClientParamsBean` in *WebLogic Server MBean Reference* at `http://e-docs.bea.com/wls/docs100/wlsmbeanref/mbeans/ClientParamsBean.html`.

- The `getExpirationLoggingPolicy` attribute in the `DeliveryFailureParamsBean` has been deprecated. BEA recommends that you update your applications to use the Message Life Cycle Logging feature described in "Message Life Cycle Logging" in *Configuring and Managing WebLogic JMS* at `http://e-docs.bea.com/wls/docs100/jms_admin/troubleshoot.html#message_life_cycle_logging`. It should also be noted that the `getExpirationLoggingPolicy` attribute now removes any leading and trailing white space that may have been embedded in an application.

JMS Message ID Format

As of WebLogic Server 9.0, the format of the JMS message ID has changed. BEA will continue to support the pre-9.0 format for existing consumers, producers, and servers. For example, existing JMS consumers may continue to view messages in the pre-9.0 format, even when received from a new JMS producer and JMS server.

Improved Message Paging

The message paging feature for freeing up virtual memory during peak message load situations is always enabled on JMS servers. Additionally, administrators no longer need to create a dedicated message paging store since paged out messages can be stored in a directory on your file system. However, for the best performance you should specify that messages be paged to a directory other than the one used by the JMS server's persistent store.

See "Paging Out Messages To Free Up Memory" in Configuring and Managing WebLogic JMS.

Thread Management

BEA recommends using Work Manager concepts to manage threads, as execute queues are no longer the default method used as of WebLogic Server 9.0. You define the rules and constraints for your application by defining a Work Manager and applying it either globally to a WebLogic Server domain or specifically to an application component. For more information, see: "Using Work Managers to Optimize Scheduled Work" in *Configuring WebLogic Server Environments* at `http://e-docs.bea.com/wls/docs100/config_wls/self_tuned.html`.

In WebLogic Server 8.1, processing was performed in multiple execute queues. If you had been using execute queues to improve performance in 8.1, you may continue to use them after you upgrade your application domains. BEA provides a `Use81StyleExecuteQueues` flag that enables you to disable the self-tuning execute pool and provide backward compatibility for upgraded applications to continue to use user-defined execute queues. For information about enabling the backward compatibility flag, and configuring and monitoring execute queues, see "Using User-defined Execute Queues" in *WebLogic Server Performance and Tuning* at `http://e-docs.bea.com/wls/docs100/perform/appb_queues.html`.

JTA Transaction Log Migration

All JTA domain configuration options are persisted from the legacy configuration file. The only changes are at the server level. As of WebLogic Server 9.0, the Transaction Manager uses the

default WebLogic persistent store to store transaction log records. During the upgrade, the Upgrade Wizard copies transaction log records to the default store. The transaction log file prefix from the existing server configuration is used only to locate the transaction log (`.tlog`) files during an upgrade; it is not preserved after the upgrade.

If the entire domain resides on a single machine, the Upgrade Wizard handles the upgrade (and copies transaction log records to the default store) for all Managed Servers during the initial domain upgrade. If Managed Servers reside on separate machines, you must upgrade each Managed Server individually, as described in "Upgrade Your Application Environment" on page 2-8.

Please note the following:

- When an explicit upgrade is performed (see "Upgrading a Domain" on page 5-5), transaction recovery does not run during the upgrade process, but it starts running when you start the server(s).

- When an implicit upgrade is performed (see "Upgrading a Domain at Administration Server Startup (Implicit Mode)" on page F-1), transaction recovery runs during the server boot process.

If you have put your transaction log files in network storage in preparation for Transaction Recovery Service migration, the log file location is not preserved after the upgrade. In this release, the WebLogic Server Transaction Manager uses the WebLogic default persistent store to store transaction log files. You can achieve the same result by moving the location of the WebLogic default persistent store to a network location. Note that you must manually copy the DAT file from the default location of the current default store to the new location of the default store.

If transactions will span multiple domains, you must configure your domain to enable inter-domain transactions. For more information, see "Configuring Domains for Inter-Domain Transactions" in *Programming WebLogic JTA* at
`http://e-docs.bea.com/wls/docs100/jta/trxcon.html#interop`.

Security

The following sections identify changes to security.

- Windows NT Authentication Provider Deprecated

- XACML Security Providers

- SAML V2 Providers

- Security MBeans

- Password Encryption

- Security for HTTP Requests

- Secure Access to MBeanHome

- Message-Level Security in Web Services

Windows NT Authentication Provider Deprecated

The Windows NT Authentication provider is deprecated as of WebLogic Server 10.0. Use one or more of the other supported Authentication providers instead.

XACML Security Providers

As of 9.2, WebLogic Server includes two new security providers, the XACML Authorization provider and the XACML Role Mapping provider. Previous releases of WebLogic Server used an authorization provider and a role mapping provider based on a proprietary security policy language. These XACML security providers support the eXtensible Access Control Markup Language (XACML) 2.0 standard from OASIS. These providers can import, export, persist, and execute policy expressed using all standard XACML 2.0 functions, attributes, and schema elements.

WebLogic domains created using WebLogic Server 10.0 include the XACML providers by default. The new XACML providers are fully compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). Existing WebLogic domains that you upgrade to 10.0 will continue to use the authorization and role mapping providers currently specified, such as third-party partner providers or the original WebLogic Authorization and Role Mapping providers. If you wish, you can migrate existing domains from using WebLogic Server proprietary providers to the XACML providers, including performing bulk imports of existing policies. For more information, see *Security for BEA WebLogic Server 10.0* at http://e-docs.bea.com/wls/docs100/security.html.

SAML V2 Providers

New versions of the SAML Credential Mapping provider and SAML Identity Assertion provider were added in WebLogic Server 9.2. The SAML Credential Mapping V1 provider and SAML

Identity Assertion V1 provider are deprecated; you should use the V2 versions of the SAML Credential Mapping and SAML Identity Assertion providers.

Although the version number of the providers has been incremented to V2, the new SAML security providers implement the SAML 1.1 standard, as did the V1 providers.

Security MBeans

The following table lists the changes to security MBeans as of WebLogic Server 9.0.

Table B-2 Changes to Security MBeans as of WebLogic Server 9.0

Type of Security MBean	Description
All security MBeans	In WebLogic Server 8.1, when you updated a security MBean attribute, the values were available to the security configuration and management hierarchy immediately, and to the security runtime hierarchy following a server reboot.
	As of WebLogic Server 9.0, whether a security MBean attribute change is effective and available to the configuration, management, and runtime hierarchies immediately or upon server reboot is controlled by setting that attribute as dynamic or non-dynamic. For more information, see "Dynamic Configuration Management" on page B-3.

Table B-2 Changes to Security MBeans as of WebLogic Server 9.0 (Continued)

Type of Security MBean	Description
`RealmMBean`, `UserLockoutManagerMBean`, and all security provider MBeans	• The `wls_getDisplay` method is deprecated. In its place, you should use the new `getName` method. In addition, the following security methods have been removed: `wls_getAttributeTag` `wls_getConstructorTag` `wls_getMBeanTag` `wls_getNotificationTag` `wls_getOperationTag`
	• The `weblogic.Admin` tools and pre-9.2 JMX security APIs can no longer be used to configure security MBeans. These utilities and APIs can still be used, however, to view and invoke methods on the security MBeans.
	For security provider MBeans (only):
	• When adding or removing a security provider, you must reboot the server before any changes will take effect.
	• When modifying an existing security provider, if you modify any non-dynamic attributes, the server must be rebooted before *any* (that is, non-dynamic or dynamic) changes will take effect. For more information, see "Dynamic Configuration Management" on page B-3.
All *custom* security provider MBeans	• By default, all custom security provider MBeans attributes are non-dynamic. For more information, see "Dynamic Configuration Management" on page B-3.
	• You can set an MBean attribute to be dynamic by setting `Dynamic="true"` for the attribute within the MDF file. For example: `<MBeanAttribute` `Name        = "Foo"` `Type        = "java.lang.String"` **`Dynamic = "true"`** `Description = "Specifies that this attribute is a dummy."` `/>`

Password Encryption

To prevent unauthorized access to sensitive data such as passwords, some attributes in configuration MBeans are encrypted. The attributes persist their values in the domain configuration files as an encrypted string. For further security, the in-memory value is stored in

the form of an encrypted byte array to help reduce the risk of the password being snooped from memory.

In pre-9.0 releases, you could edit the `config.xml` file to specify an encrypted attribute, such as a password, in clear-text or encrypted format. In this case, when booted, the WebLogic Server will encrypt the information the next time it writes to the file.

As of WebLogic Server 9.0, when operating in production mode, the password of an encrypted attribute must be encrypted in the configuration files. In development mode, the password of an encrypted attribute can be either encrypted or clear-text.

You can use the `weblogic.security.Encrypt` command-line utility to encrypt the passwords, as follows:

```
java weblogic.security.Encrypt
```

You are prompted to enter a password, and the command returns the encrypted version. Then, copy the encrypted password returned into the appropriate file.

This utility is not just used for passwords in the configuration files. It can also be used to encrypt passwords in descriptor files (for example, a JDBC or JMS descriptor) and in deployment plans. For more information, see "encrypt" in "Using the WebLogic Server Java Utilities" in *WebLogic Server Command Reference* at
http://e-docs.bea.com/wls/docs100/admin_ref/utils.html.

Security for HTTP Requests

By default, when an instance of WebLogic Server 10.0 responds to an HTTP request, its HTTP response header does not include the WebLogic Server name and version number. This behavior is different from releases prior to WebLogic Server 9.0.

To have the name and version number included in the HTTP response header when responding to an HTTP request, enable the Send Server Header attribute for the WebLogic Server instance in the Administration Console. The attribute is located on the Server −> *ServerName* −> Protocols −>HTTP tab under the Advanced Options section. Note that enabling this feature may creates a security risk if a possible attacker knows about a vulnerability in the specified version of WebLogic Server.

For more information about ensuring security, see "Securing the WebLogic Security Service" in "Ensuring the Security of Your Production Environment" in *Securing a Production Environment* at http://e-docs.bea.com/wls/docs100/lockdown/practices.html.

Secure Access to MBeanHome

In pre-9.0 releases of WebLogic Server, anonymous access to `MBeanHome` was enabled by default. With the security enhancements delivered as of WebLogic Server 9.0, anonymous access to `MBeanHome` is no longer allowed.

Although doing so is not recommended, you can re-enable anonymous access by specifying the following flag when starting the server:

```
-Dweblogic.management.anonymousAdminLookupEnabled
```

Message-Level Security in Web Services

As of WebLogic Server 9.0, message-level security in Web Services was enhanced to use the standards-based Web Services Policy Framework (WS-Policy). WS-Policy provides a flexible and extensible grammar for expressing the capabilities, requirements, and general characteristics of entities in an XML Web Services-based system. For more information about WS-Policy, see "Using WS-Policy Files for Message-Level Security Configuration" in *Programming Web Services for WebLogic Server* at

http://e-docs.bea.com/wls/docs100/webserv/security.html#ws_policy.

In 8.1, the implementation was based on an OASIS implementation of the Web Services Security (WSS) standard. This implementation is supported for backward compatibility, but is deprecated as of 9.0. For more information, see

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

Web Services

WebLogic Server 8.1 Web Services can be run in 10.0, although the 8.1 Web Services run-time engine has been deprecated as of 9.0.

No web service upgrade is required going from 9.2 to 10.0.

The WebLogic Server 7.0 Web Services need to be upgraded to at least 8.1 in order to run in 10.0. For more information, see "Upgrading WebLogic Web Services" in *Programming WebLogic Web Services* at http://e-docs.bea.com/wls/docs81/webserv/migrate.html.

BEA strongly recommends that you upgrade, to 10.0, all of your 8.1 Web Services, including any 7.0 Web Services that have been upgraded to 8.1. For information about upgrading your existing 8.1 Web Services, see Upgrading WebLogic Web Services From Previous Releases to 10.0 in *WebLogic Web Services: Getting Started*.

Note: See also "Message-Level Security in Web Services" on page B-15.

Web Applications, JSPs, and Servlets

The following sections provide important compatibility information for Web applications, JSPs, and Servlets in WebLogic Server 10.0:

- "Deprecated and Obsolete Web Application Features" on page B-16
- "Backwards Compatibility Flags" on page B-17
- "Servlet Path Mapping" on page B-18

Deprecated and Obsolete Web Application Features

For a list of Web application features that are deprecated or are not supported as of WebLogic Server 10.0, see WebLogic Server and WebLogic Express Release Notes.

BASIC Authentication with Unsecured Resources

For WebLogic Server versions 9.2 and later, client requests that use HTTP BASIC authentication must pass WebLogic Server authentication, even if access control is not enabled on the target resource.

The setting of the Security Configuration MBean flag `enforce-valid-basic-auth-credentials` determines this behavior. (The DomainMBean can return the new Security Configuration MBean for the domain.) It specifies whether or not the system should allow requests with invalid HTTP BASIC authentication credentials to access unsecured resources.

Note: The Security Configuration MBean provides domain-wide security configuration information. The `enforce-valid-basic-auth-credentials` flag effects the entire domain.

The `enforce-valid-basic-auth-credentials` flag is true by default, and WebLogic Server authentication is performed. If authentication fails, the request is rejected. WebLogic Server must therefore have knowledge of the user and password.

See Understanding BASIC Authentication with Unsecured Resources in *Programming WebLogic Security* for complete information.

Backwards Compatibility Flags

For WebLogic Server 10.0, backward compatibility for WebLogic Server 9.2 or earlier is supported via the `backward-compatible` element within the `jsp-descriptor` element, as described in this section and in `jsp-descriptor` in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.

JSP 2.1 Support and Compatibility With JSP 2.0 Web Applications

JSP 2.1 is supported as of WebLogic Server 10.0. Depending on the version of the Web application (version 2.4 or 2.5) and the setting of the `backward-compatible` element, Weblogic Server 10.0 also supports JSP 2.0.

See Backwards Compatibility Flags in *Developing Web Applications, Servlets, and JSPs for WebLogic Server* for important information about the buffer suffix setting and implicit servlet 2.5 package imports.

Support for JSP 2.0

JSP 2.0 was supported as of WebLogic Server 9.0, and continues to be supported as described in "Backwards Compatibility Flags" on page B-17. Please note the following changes to the JSP behavior as required in support of JSP 2.0:

- If a JSP does not participate in a session (or if the session in which a JSP participates is invalid), an `IllegalStateException` is thrown when the following command is executed:

  ```
  PageContext.getAttribute(name, PageContext.SESSION_SCOPE)
  ```

  If you are not concerned about this type of error, you can catch and ignore it.

- `JspWriterImpl` now replaces `\n` with `System.getProperty("line.separator")` for each `printline` function. This replacement causes problems with JSPs that:

  – Contain multiple page directives that appear on new lines. For example:

  ```
  <%@ page import="com.foo.bar.*" %>
  <%@ page import="com.foo.xyz.*" %>
  ...
  ```

  – Generate output in XML format.

  – Generate an XML declaration following the page directives.

  – Are served by Windows systems. In this case, `\r\n` is output for each page directive.

– Are viewed using Internet Explorer.

When viewed in Internet Explorer, each page directive outputs an empty `\r\n` and the XML declaration (`<?xml version="1.0" encoding="iso-8859-1"?>`) appears after every new line. Internet Explorer displays an error message indicating that it cannot locate the declaration and that the page cannot be viewed, even though it can be compiled.

To work around any issues caused by changes to `JspWriterImpl`, you can perform one or both of the following tasks:

– Define the XML declaration at the top of the page.

– Group the page directives into a single declaration, for example:

```
<%@ page import="com.foo.bar.*, com.foo.baz.*"
contentType="text/html" pageEncoding="UTF-8" errorPage="Error.jsp" %>
```

- The JSP `<param name>` tag no longer allows run-time expression values. For example:

```
<jsp:param name="<%= AdminActions.RETURN_LINK %>" value="<%= returnlink
%>" />
```

You can continue to support this feature by disabling the **Do not set backwards compatibility flags** upgrade option during the domain upgrade, as described in "Select Upgrade Options" on page 5-9, or enabling the `backwardCompatible` flag in the `weblogic.xml` file, as follows:

```
<jsp-descriptor>
<jsp-param>
<param-name>backwardCompatible</param-name>
<param-value>true</param-value>
</jsp-param>
</jsp-descriptor>
```

Servlet Path Mapping

As of the Servlet 2.3 Specification from Sun Microsystems, which is downloadable at http://java.sun.com/products/servlet/download.html#specs, the following syntax is used to define mappings:

- A servlet path string that contains only the `/` (forward slash) character indicates the default servlet of the application. The servlet path resolves to the request URI minus the context path; in this case, the path resolves to `null`.

- A String that begins with an `*` (asterisk) specifies an extension mapping.

These changes introduce a change in behavior with the following `HttpServletRequest` methods:

- `getPathInfo`

- `getServletPath`

To better illustrate the change in behavior, consider a request `/abc/def.html` that resolves to ServletA:

- If `/` maps to ServletA, then `servletPath="abc/def.html"` and `pathInfo=null`.

- If `/*` maps to ServletA, then `servletPath=""` and `pathInfo="abc/def.html"`.

To ensure that the path info returned is non-null, replace all occurrences of the `/` (forward slash) servlet mapping string with `/*`.

XML Implementation

Please note the following changes to XML support as of WebLogic Server 9.0:

- The default XML parser is the XML parser shipped with the Sun Java 2 JDK 5.0. The previous default XML parser, the Apache Xerces parser (`weblogic.apache.xerces.*`), is deprecated as of 9.0.

  You can modify the XML parser that is used by default using the Administration Console. For information about configuring the XML parser, see "Difference In Default Parsers Between Versions 8.1 and 9.0 of WebLogic Server" in *Programming WebLogic XML* at

  `http://e-docs.bea.com/wls/docs100/xml/overview.html#deprecated_xerces`.

- As of 9.0, WebLogic Server supports Streaming API for XML (StAX), a standard specification from the Java Community Process that provides an easy and intuitive means of parsing and generating XML documents. StAX gives you more control over XML parsing than the WebLogic XML Streaming API, which is deprecated as of 9.0. For information about using StAX, see "Using the Streaming API for XML (StAX)" in *Programming WebLogic XML* at

  `http://e-docs.bea.com/wls/docs100/xml/stax.html`.

- You can no longer parse XML documents from within a servlet using the `setAttribute` and `getAttribute` methods without some preliminary setup. Specifically, as of 9.0, you must configure a WebLogic Server servlet filter called `weblogic.servlet.XMLParsingHelper` (deployed, by default, on all WebLogic Server instances) as part of your Web application. For more information, see "Parsing XML

Documents in a Servlet" in *Programming WebLogic XML* at
http://e-docs.bea.com/wls/docs100/xml/programming.html#apps005.

XMLBeans and XQuery Implementation

As of 9.0, the XMLBean implementation in WebLogic Server has been moved from an internal BEA library (`com.bea.xml`) to the Apache open source project (`org.apache.xmlbeans`).

If you used XMLBeans in your WebLogic Server 8.1 applications, you must perform the following steps:

1. Update the package name used by XMLBeans from `com.bea.xml` to `org.apache.xmlbeans`.

2. Recompile your XMLBean schemas to update the schema metadata (`.xsb`) files and generated code.

As of 9.0, the XMLQuery (XQuery) implementation conforms to the following specifications:

- *XQuery 1.0 and XPath 2.0 Data Model—W3C Working Draft 23 July 2004* available from the W3C Web site at http://www.w3.org/TR/2004/WD-xpath-datamodel-20040723.

- *XQuery 1.0: An XML Query Language—W3C Working Draft 23 July 2004* available from the W3C Web site at http://www.w3.org/TR/2004/WD-xquery-20040723.

In WebLogic Server 8.1, the XQuery implementation conformed to *XQuery 1.0 and XPath 2.0 Functions and Operators—W3C Working Draft 16 August 2002*, available from the W3C Web site at http://www.w3.org/TR/2002/WD-xquery-operators-20020816. The 2002 XQuery implementation is deprecated as of 9.0.

In most cases, simple XQuery and XPath operations in pre-9.0 code will behave the same in 10.0. To ensure that the XQuery and XPath operations produce the expected results, you can review and/or update the existing `XMLObject.selectPath()` and `XMLObject.execQuery()` method calls using one of the following methods:

- To guarantee 8.1-style behavior, update the existing method calls to include a new parameter that specifies that the 2002 XQuery engine will be used instead of the new 2004 XQuery engine. For example:

```
import org.apache.xmlbeans.impl.store.Path;
XmlObject xo = ?
xo.selectPath(".//c", (new
XmlOptions()).put(Path._forceXqrl2002ForXpathXQuery));
```

Note: The 2002 XQuery engine is deprecated as of WebLogic Server 9.0, and is available for backward compatibility. It is only used if you specify this parameter. Otherwise, the 2004 XQuery engine is used, by default.

● To guarantee conformance with the 2004 XQuery engine, review your pre-9.0 scripts to identify any changes that may be required with the syntax and/or semantics of the XQuery strings that are passed to the method calls and update methods accordingly.

As of 9.0, the behavior of `XMLCursor.moveXML()` has changed. In 8.1, a cursor that was inside a moved fragment remained on the original document. As of 9.0, cursors move with fragments.

WebLogic Administration and Configuration Scripts

Due to changes with the MBean hierarchy, BEA does not guarantee that existing configuration and administration scripts (such as WLST, `wlconfig`, `weblogic.Admin`, Ant, and so on) will run in 10.0. BEA recommends that you update your scripts to take advantage of the new features provided as of WebLogic Server 10.0. For more information about new features and changes in the MBean hierarchy, see "What's New in WebLogic Server 10.0" in the *WebLogic Server and WebLogic Express Release Notes* at

http://e-docs.bea.com/wls/docs100/notes/new.html.

For additional information about upgrading your application infrastructure and the scripting tools that have been deprecated, see "Step 1: Upgrade Your Application Infrastructure" on page 2-11.

Deployment Descriptor Validation and Conversion

This section describes changes in the use of deployment descriptors in a WebLogic Server environment, as of release 9.0:

● Deployment descriptor validation is more strict as of the 9.0 release of the EJBGen and ejbc tools. For example, an error is returned if a `cmr-field` is defined in `@ejbgen:relation`, but there are no methods tagged with `@ejbgen:cmr-field` in the Bean class.

Note: ejbc is deprecated as of WebLogic Server 9.0; you should use `appc` instead. For more information, see "appc Reference" in *Programming WebLogic Enterprise JavaBeans* at http://e-docs.bea.com/wls/docs100/ejb/appc_ejbc.html.

● In pre-9.0 versions of WebLogic Server, applications that define multiple modules, as illustrated in the following excerpt from a configuration file, are deployed successfully regardless of whether a `META-INF\application.xml` deployment descriptor is defined as part of the application:

```
<Application Deployed="true" Name="SessionBeanLifeCycleBean"
Path="C:\bea\weblogic70\tools\deployment\ejb" TwoPhase="false">
<EJBComponent Name="CMFinderTestBean" Targets="myserver"
URI="CMFinderTestBean.jar"/>
<EJBComponent Name="SessionBeanLifeCycleBean" Targets="myserver"
URI="SessionBeanLifeCycleBean.jar"/>
</Application>
```

As of 9.0, the `META-INF\application.xml` deployment descriptor is *required* if a deployed application defines multiple modules. If this type of deployment descriptor is not provided, the upgrade fails with an error similar to the following:

```
[J2EE Deployment SPI:260089]Unable to determine type of application at
path 'C:\bea\weblogic70\tools\deployment\ejb' and upgrade will not
succeed.
```

When upgrading a domain, make sure that the deployed applications adhere to the proper Java EE application format. For example, if required by the application, make sure that the applications define the `META-INF\application.xml` and/or `META-INF\weblogic-application.xml` deployment descriptors.

For more information about the deployment descriptors, see "Enterprise Application Deployment Descriptor Elements" in *Developing Applications with WebLogic Server* at http://e-docs.bea.com/wls/docs100/programming/app_xml.html.

- So that your applications can take advantage of the features in the current Java EE specification and release of WebLogic Server, BEA recommends that you upgrade deployment descriptors when you upgrade applications to a new release of WebLogic Server. For more information, see "Upgrading Deployment Descriptors From Previous Releases of Java EE and WebLogic Server" in *Developing Applications with WebLogic Server* at http://e-docs.bea.com/wls/docs100/programming/overview.html#ddconverter .

Deprecated Startup and Shutdown Classes

As of 9.0, application-scoped startup and shutdown classes were deprecated in WebLogic Server, in favor of applications that respond to application lifecycle events. BEA recommends that you update your application environment to use the lifecycle events in place of application-scoped and domain-level startup and shutdown classes. For more information, see "Programming Application Lifecycle Events" in *Developing Applications with WebLogic Server* at http://e-docs.bea.com/wls/docs100/programming/lifecycle.html.

Administration Console Extension Architecture

In WebLogic Server version 9.0, the Administration Console was built on the WebLogic Portal Framework, which makes it more open and more readily extensible. The architecture necessitated new procedures for extending the Administration Console. WebLogic Administration Console extensions built for releases of WebLogic Server prior to 9.0 will not function with the new Console infrastructure.

For more information about extending the Administration Console, see *Extending the Administration Console* at
http://e-docs.bea.com/wls/docs100/console_ext/index.html.

Important Console-Extension Information for Version 9.2

Version 9.2 of WebLogic Server introduced the following changes to console extensions:

- Prior to this release, Administration Console extensions could import a set of third-party JSP tag libraries by specifying a pathname to the tag library file. For example,

```
<%@ taglib uri="/WEB-INF/beehive-netui-tags-template.tld"
prefix="beehive-template" %>
```

As of this release, Administration Console extensions that use these third-party JSP tag libraries from the WebLogic Server installation must use pre-defined, absolute URIs to specify the tag libraries. For example:

```
<%@ taglib uri="http://beehive.apache.org/netui/tags-template-1.0"
prefix="beehive-template" %>
```

The Administration Console's `web.xml` file maps these URIs to tag libraries within the WebLogic Server installation. This mapping facility enables BEA to reorganize its installation directory without requiring you to change your JSPs.

Any Administration Console extensions that use the old pathname syntax to import Apache Struts, Apache Beehive, or the JSTL tag libraries must update all pathnames to the new URIs.

The URI for the WebLogic Server Console Extension tag library (console-html.tld) remains unchanged: `/WEB-INF/console-html.tld`.

For more information, see JSP Templates and Tag Libraries.

- By convention, portal include files (.pinc) files are now called portal book files (.book).

WebLogic Portal Skeleton URI References Should be Fully Qualified

WebLogic Portal requires that any explicit Skeleton URI references be fully qualified relative to the webapp. However, the documentation and some of the console extension examples have sometimes used relative references to these skeletons. Consider the following incorrect example:

```
<netuix:singleLevelMenu markupType="Menu" markupName="singleLevelMenu"
skeletonUri="singlelevelmenu_children2.jsp"/>
```

This example should have been correctly specified as:

```
<netuix:singleLevelMenu markupType="Menu" markupName="singleLevelMenu"
skeletonUri="/framework/skeletons/default/singlelevelmenu_children2.jsp"/>
```

For this release, relative skeleton URI references will continue to work. However, any console extensions that you have written should be updated to use fully qualified skeleton URIs, as these relative references may no longer function correctly in a future release.

Resource Adapters

The following table lists the configuration settings for resource adapters that are deprecated or no longer supported. For more information about new features and changes, see "New and Changed Features in This Release" in *Programming WebLogic Resource Adapters* at http://e-docs.bea.com/wls/docs100/resadapter/intro.html#features.

Table B-3 Deprecated or Unsupported Resource Adapter Configuration Settings

This element ...	As of WebLogic Server 9.0 ...
Link-Ref Mechanism	This element has been deprecated and replaced by the new Java EE libraries feature. For more information about Java EE libraries, see "Creating Shared J2EE Libraries and Optional Packages" in *Developing Applications with WebLogic Server* at `http://e-docs.bea.com/wls/docs100/programmin g/libraries.html`.
	The Link-Ref mechanism is still supported in this release for resource adapters developed under the J2CA 1.0 Specification. For more information about using the Link-Ref mechanism with 1.0 resource adapters, see "(Deprecated) Configuring the Link-Ref Mechanism" in "Configuring the weblogic-ra.xml File" in *Programming WebLogic Resource Adapters* at `http://e-docs.bea.com/wls/docs100/resadapter /creating.html#weblogic-ra-xml`.
`<shrink-period-minutes>`	This element has been deprecated and replaced by `<shrink-frequency-seconds>`, which allows you to specify the shrink period in increments of seconds, instead of minutes.
	The `<shrink-frequency-seconds>` element overrides the `<shrink-period-minutes>` element if both are set.
`<connection-maxidle-time>`	This element has been deprecated and is replaced by `<inactive-connection-timeout-seconds>`, which allows you to specify the connection timeout in increments of seconds.
	The `<inactive-connection-timeout-seconds>` element overrides the `<connection-maxidle-time>` element if both are set.
`<security-principal-map>`	This element is no longer supported; the security principal map is configured using the Administration Console.
	You should remove the `<security-principal-map>` definition from the `weblogic-ra.xml` file. Otherwise, deployment of the resource adapter fails.

Table B-3 Deprecated or Unsupported Resource Adapter Configuration Settings (Continued)

This element ...	As of WebLogic Server 9.0 ...
`<connection-cleanup-frequency>`	This element is no longer supported and is ignored during deployment.
`<connection-duration-time>`	This element is no longer supported and is ignored during deployment.

WLEC

WLEC was deprecated in WebLogic Server 8.1. WLEC users should move applications to the WebLogic Tuxedo Connector, as described in *WLEC to WebLogic Tuxedo Connector Migration Guide* at `http://e-docs.bea.com/wls/docs100/wlec_migration/index.html`.

SNMP MIB Refresh Interval and Server Status Check Interval No Longer Used

The SNMPAgentMBean MBean MibDataRefreshInterval and ServerStatusCheckIntervalFactor attributes are deprecated and are ignored in WebLogic Server 10.0.

Backward Compatibility Flags

The following configuration flags are available to support backward compatibility when you upgrade a domain. By default, these flags are set to support backward compatibility, unless you disable them by selecting the **Do not set backwards compatibility flags** option during an upgrade, as described in "Upgrading a Domain in Graphical Mode" on page 5-6.

Table B-4 Backward Compatibility Flags

Category	Backward Compatibility Flag	For More Information
Security	• `EnforceStrictURLPattern`—Specifies whether the server should enforce strict adherance of URL patterns to the Servlet 2.4 specification. During an upgrade, this flag is set to `false` for backward compatibility. • `WebAppFilesCaseInsensitive`—Specifies whether the URL-pattern matching behavior is case-insensitive for security constraints, servlets, filters, virtual hosts, and so on, in the Webapp container and external security policies. During an upgrade, this flag is set to `os`, which sets URL-pattern matching as case-sensitive on all platforms except Windows, for compatibility with pre-9.0 releases. As of WebLogic Server 9.0, URL-pattern matching is strictly enforced across operating systems	`SecurityConfiguration MBean` at `http://e-docs.bea.com /wls/docs100/wlsmbean ref/mbeans/SecurityCo nfigurationMBean.html`

Table B-4 Backward Compatibility Flags (Continued)

Category	Backward Compatibility Flag	For More Information
Web Application	• `backward-compatible`—Specifies which JSP version is supported. Depending on the version of the Web application (version 2.4 or 2.5) and the setting of the backward-compatible element, Weblogic Server 10.0 supports JSP 2.1 or JSP 2.0. `backward-compatible` is located within the `jsp-descriptor` element, as described in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*. • `AllowAllRoles`—Specifies that a wildcard (*) character can be used to set the role name to enable all users/roles in the *realm* to have access to the resource collection. As of WebLogic Server 9.0, if you specify the wildcard (*) character as the role name, all users/roles in the *Web application* will have access to the resource collection. • `FilterDispatchedRequestsEnabled`—Specifies whether to apply filters to dispatched requests. As of WebLogic Server 9.0, the new Dispatcher element makes this behavior explicit. • `JSPCompilerBackwardsCompatible`—Specifies whether to allow JSPs that do not comply with the JSP 2.0 specification. • `ReloginEnabled`—Specifies whether to allow users multiple attempts to log in to a Web page if the original credentials were not supported. As of WebLogic Server 9.0, the FORM/BASIC authentication behavior has been modified to return the 403 (FORBIDDEN) Web page. • `RtexprvalueJspParamName`—Specifies whether to allow run-time expression values for the JSP `<param name>` tag. As of WebLogic Server 9.0, the JSP Compiler no longer allows run-time expression values.	`WebAppContainerMBean` at `http://e-docs.bea.com/wls/docs100/wlsmbeanref/mbeans/WebAppContainerMBean.html`

Deprecated and Removed APIs

This section summarizes APIs that have been deprecated or removed in WebLogic Server 9.0.

Note: If you want to rebuild the WebLogic Server 8.1 MedRec application in the 10.0 environment, you must replace references to the `weblogic.webservice.tools.wsdlp` package with the new package name, `weblogic.webservice.wsdl`, in two Java files:

medrec/src/common/web/com/bea/medrec/utils/MedRecWebAppUtils.java
and
medrec/src/clients/com/bea/medrec/webservices/swing/EditProfileFram
e.java.

The 8.1, 9.0, and 9.2 packages, `weblogic.webservice.tools.wsdlp` and
`weblogic.webservice.wsdl`, respectively, are not publicly available.

See also "Web Services" on page B-15.

Deprecated APIs

For a list of **deprecated** APIs, see
http://e-docs.bea.com/wls/docs100/javadocs/deprecated-list.html.

Removed APIs

This section describes the APIs that have been removed in WebLogic Server 9.0, including:

- "Deprecated APIs in WebLogic Server 10.0" on page B-29

- "Removed APIs, Deprecated in WebLogic Server 8.1" on page B-30

- "Removed APIs, Deprecated in WebLogic Server 7.0" on page B-30

- "Removed APIs in 9.0 (Not Deprecated)" on page B-35

Deprecated APIs in WebLogic Server 10.0

Table B-5 provides a list of APIs that have been deprecated in WebLogic Server 10.0.

Table B-5 Deprecated APIs in WebLogic Server 10.0

Package Name	Interface/Class/Exception	Method/Variable
weblogic.jws	WLHttpsTransport	*
weblogic.logging	ConsoleFormatter	ConsoleFormatter(weblogic.management.configuration.LogMBean)
weblogic.logging.log4j	WLLog4jLogEvent	setThrowableInfo(weblogic.logging.ThrowableInfo)

Table B-5 Deprecated APIs in WebLogic Server 10.0

`weblogic.logging.lo g4j`	`WLLog4jLogEvent`	`getThrowableInfo()`
`weblogic.wsee.jws`	`JwsContext`	`getLogger(java.lang.String)`
`weblogic.wsee.jws`	`SoapFaultException`	
`weblogic.wsee.jws.u til`	`Logger`	`*`

Removed APIs, Deprecated in WebLogic Server 8.1

The following table provides a list of APIs that have been removed in WebLogic Server 10.0 and were deprecated in WebLogic Server 8.1.

Table B-6 Removed APIs, Deprecated in WebLogic Server 8.1

Package Name	Interface/Class/Exc eption	Method/Variable
`weblogic.jms`	`ServerSessionPoolFa ctory`	`createServerSessionPool(javax.jm s.Connection, int, int, boolean, java.lang.String,java.io.Seriali zable)`
`weblogic.jms`	`ServerSessionPoolFa ctory`	`getServerSessionPool(javax.jms.Q ueueConnection, int, boolean, int, java.lang.String)`
`weblogic.jms`	`ServerSessionPoolFa ctory`	`getServerSessionPool(javax.jms.T opicConnection, int, boolean, int, java.lang.String)`
`weblogic.time.commo n`	`ScheduledTriggerDef`	`setDispatchPolicy(java.lang.Stri ng)`

Removed APIs, Deprecated in WebLogic Server 7.0

Table B-7 provides a list of APIs that have been removed in WebLogic Server 10.0 and were deprecated in WebLogic Server 7.0.

Table B-7 Removed APIs—Deprecated in WebLogic Server 7.0

Package Name	Interface/Class/Exception	Method/Variable
weblogic.common	WLSerializable	*
weblogic.io.common	T3File	*
weblogic.io.common	T3FileInputStream	*
weblogic.io.common	T3FileOutputStream	*
weblogic.io.common	T3FileSystem	*
weblogic.security	AuthenticationExcep tion	
weblogic.security	Certificate	*
weblogic.security	Cipher	*
weblogic.security	Cipher3EDE	*
weblogic.security	CipherCBC	*
weblogic.security	CipherException	
weblogic.security	Coder	*
weblogic.security	CoderException	
weblogic.security	CoderInputStream	*
weblogic.security	CoderOutputStream	*
weblogic.security	DES	*
weblogic.security	DiffieHellman	*
weblogic.security	DigestInputStream	*
weblogic.security	DigestOutputStream	*
weblogic.security	DigestSignature	*
weblogic.security	Entity	*
weblogic.security	HexOutputStream	*

Table B-7 Removed APIs—Deprecated in WebLogic Server 7.0 (Continued)

weblogic.security	JDK11Certificate	*
weblogic.security	KeyManagementException	*
weblogic.security	MD5RandomBitsSource	*
weblogic.security	NullCipher	*
weblogic.security	NullOutputStream	*
weblogic.security	PKCS5	*
weblogic.security	Padding	*
weblogic.security	RC4	*
weblogic.security	RSA	*
weblogic.security	RSAKey	*
weblogic.security	RSAMDSignature	*
weblogic.security	RSAPrivateKey	*
weblogic.security	RSAPrivateKeyPKCS8	*
weblogic.security	RSAPublicKey	*
weblogic.security	RSApkcs1	*
weblogic.security	RandomBitsSource	*
weblogic.security.SSL	BadMACException	
weblogic.security.SSL	Biguint	*
weblogic.security.SSL	CipherSpec	*
weblogic.security.SSL	RecordInputStream	*
weblogic.security.SSL	SSLCertificate	*

Table B-7 Removed APIs—Deprecated in WebLogic Server 7.0 (Continued)

`weblogic.security.S SL`	`SSLParams`	`*`
`weblogic.security.S SL`	`SSLSocket`	`*`
`weblogic.security.S SL`	`SessionParams`	`*`
`weblogic.security.S SL`	`TrustManager`	`certificateCallback(weblogic.sec urity.X509[],int)`
`weblogic.security`	`Signature`	`*`
`weblogic.security`	`SimpleRandomBitsSou rce`	`*`
`weblogic.security`	`SpinnerRandomBitsSo urce`	`*`
`weblogic.security`	`StreamCipher`	`*`
`weblogic.security`	`Streamable`	`*`
`weblogic.security`	`SymmetricCipher`	`*`
`weblogic.security`	`TeeOutputStream`	`*`
`weblogic.security`	`X500Name`	`*`
`weblogic.security`	`X509`	`*`
`weblogic.security.a cl`	`CertAuthentication`	`authenticate(java.lang.String,we blogic.security.Certificate[],bo olean)`
`weblogic.security.a cl`	`CertAuthenticator`	`authenticate(java.lang.String,we blogic.security.Certificate[],bo olean)`
`weblogic.security.a cl`	`SSLUserInfo`	`setSSLCertificate(weblogic.secur ity.X509)`

Table B-7 Removed APIs—Deprecated in WebLogic Server 7.0 (Continued)

weblogic.security.a cl	Security	receiveSecurityMessage(weblogic. rjvm.JVMID, weblogic.security.acl SecurityMessage)
weblogic.security.a cl	SecurityMessage	execute(weblogic.rjvm.JVMID)
weblogic.security.a udit	Audit	certificateInvalid(java.lang.Str ing, java.lang.Object,weblogic.securi ty.X509)
weblogic.security.a udit	Audit	rootCAInvalid(java.lang.String,j ava.lang.Object, weblogic.security.X509)
weblogic.security.a udit	AuditProvider	certificateInvalid(java.lang.Str ing, java.lang.Object,weblogic.securi ty.X509)
weblogic.security.a udit	AuditProvider	rootCAInvalid(java.lang.String, java.lang.Object,weblogic.securi ty.X509)
weblogic.security.s ervice	InvocableResource	toString()
weblogic.security.s ervice	MBeanResource	*
weblogic.security.s ervice	RealmAdapterAclReso urce	*
weblogic.servlet.se curity	AuthFilter	doSuccessAuth(javax.servlet.Serv letRequest, javax.servlet.ServletResponse)
weblogic.servlet.se curity	AuthFilter	doFailAuth(javax.servlet.Servlet Request, javax.servlet.ServletResponse)
weblogic.servlet.se curity	AuthFilter	doPreAuth(javax.servlet.ServletR equest, javax.servlet.ServletResponse)

Table B-7 Removed APIs—Deprecated in WebLogic Server 7.0 (Continued)

weblogic.servlet.se curity	AuthFilter	service(javax.servlet.ServletReq uest, javax.servlet.ServletResponse)
weblogic.servlet.se curity	ServletAuthenticati on	authObject(java.lang.String, java.lang.Object,javax.servlet.h ttp.HttpSession, javax.servlet.http.HttpServletRe quest)
weblogic.servlet.se curity	ServletAuthenticati on	authObject(java.lang.String, java.lang.Object,javax.servlet.h ttp.HttpSession, javax.servlet.http.HttpServletRe quest)
weblogic.servlet.se curity	ServletAuthenticati on	weak(java.lang.String, java.lang.String,javax.servlet.h ttp.HttpSession)
weblogic.xml.sax	XMLInputSource	*

Removed APIs in 9.0 (Not Deprecated)

The following provides a list of APIs that were removed in WebLogic Server 9.0, without being deprecated. Except where indicated, the APIs were removed because the related functionality was no longer supported.

Note: In the following table, * (asterisk) represents a wildcard character.

Table B-8 Removed APIs (Not Deprecated)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.apache	*	*
weblogic.common	CallbackDispatcher	CallbackDispatcher(weblogic. common.ClientCallback, boolean)
	T3ResourceDef T3ResourceFactory	*

Table B-8 Removed APIs (Not Deprecated) (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.jdbc.co mmon		*OracleBlob.* *OracleConnection.* *OracleCallableStatement.* *OracleLobCloser.* *OracleClob.*
weblogic.jms		*VirtualDestination.*
weblogic.jms.ext ensions	JMSHelper	*getRepositoryForDomain.*
weblogic.jndi	Environment	*getProviderRJVM.* *getSecuritySubject.* *getUseIIOPServiceProvider.* *setProviderRJVM.* *setSecuritySubject.* *setUseIIOPServiceProvider.*
	WLContext	*PROVIDER_RJVM.* *USE_IIOP_SERVICE_PROVIDER.*
weblogic.logging	ConsoleHandler	*isLoggable.*[1]
	FileStreamHandler	*isLoggable.*[1]
	WLLevel	*WLLevel.*
	WLLogRecord	*setId.* *setMachineName.* *setServerName.* *setThreadName.* *setTransactionId.* *setUserId.*

Table B-8 Removed APIs (Not Deprecated) (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.managem ent		*WebServiceLegalHelper.*
	AdminServerAdmin	*
	DeploymentException	getApplication()
	DistributedManagementExcepti on	MAX_EXCEPTIONS
	LocalAdminServer	*
	ManagedServerAdmin	*
	RemoteMBeanServer	java.lang.String getSharedCommoMBeanRepositor yIdentifier(javax.management .ObjectName) java.util.List getCommoMBeans(javax.managem ent.ObjectName)
	tools	*
	WebLogicObjectName	WebLogicObjectName(java.lang .String, java.lang.String)
	UndeploymentException	getApplication()
	utils	InvalidPasswordException InvalidPredicateException
weblogic.managem ent.configuratio n	*	*
weblogic.managem ent.runtime	*	*
weblogic.managem ent.security	BaseMBeanImpl	*getRequiredModelMBean.*
	ProviderMBean	*setRealm*
	RealmMBean	*setAdjudicator* *setUserLockoutManager*

Table B-8 Removed APIs (Not Deprecated) (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.managem ent.security.aut hentication	UserLockoutManagerMBean	*setRealm*
weblogic.managem ent.timer	Timer	void addNotification(weblogic.tim e.common.internal.TimerNotif ication) java.lang.String getDispatchPolicy() void removeNotification(weblogic. time.common.internal.TimerNo tification) void removeNotification(weblogic. time.common.internal.TimerNo tification, boolean) void run() void setDispatchPolicy(java.lang. String) void start(java.lang.ThreadGroup, java.lang.String,int)
weblogic.net.htt p	HttpClient	
	HttpsClient	
	HttpUnauthorizedException	
	HttpURLConnection	*
	HttpsURLConnection	*
	KeepAliveCache	
	KeepAliveStream	
	MessageHeader	

Table B-8 Removed APIs (Not Deprecated) (Continued)

Package Name	Interface/Class/Exception	Method/Variable
`weblogic.rmi`	`Naming`	`void bind(java.lang.String, weblogic.rmi.Remote)` `weblogic.rmi.Remote lookup(java.lang.String)` `void rebind(java.lang.String, weblogic.rmi.Remote)`
`weblogic.security`	`HMAC`	`*`
	`Key`	`*`
`weblogic.security.providers.authentication`	`*`	`*`
`weblogic.security.providers.authorization`	`*`	`*`
`weblogic.security.service`	`Auditor`	`*`
	`SecurityManager`	`*`
	`WebResource`	`*`
	`WebServiceResource`	`void initialize(java.lang.String) void main(java.lang.String[]) WebServiceResource WebServiceResource(weblogic.security.service.WebResource, java.lang.String, java.lang.String[], java.lang.String[])`
`weblogic.security.spi`	`KeyStoreProvider`	

Table B-8 Removed APIs (Not Deprecated) (Continued)

Package Name	Interface/Class/Exception	Method/Variable
`weblogic.securit y.SSL`	`HostnameVerification`	
	`HostnameVerifier`	`*verify(java.lang.String, java.lang.String)` `*verify(java.net.InetAddress , weblogic.security.X509)`
	`SSLClientInfo`	
	`SSLSocketFactory`	`*SSLSocketFactory()`
`weblogic.servlet .proxy`	`GenericProxyServlet`	
	`GenericProxyServlet`	`ProxyConnection` `ProxyConnectionPool`
	`HttpClusterServlet`	`RequestInfo` `Server` `ServerList`
	`HttpProxyServlet`	
`weblogic.webserv ice.async`	`KernelFeederImpl`	`ExecuteTask void execute(weblogic.kernel.Exec uteThread)`
`weblogic.webserv ice.client`	`BaseWLSSLAdapter`	`void main(java.lang.String)` `void _setStrictChecking(boolean)`
	`JSSEAdapter`	`javax.net.ssl.SSLSocketFacto ry getSocketFactory()` `void setSocketFactory(javax.net.s sl.SSLSocketFactory)`
	`SSLAdapterFactory`	`void main(java.lang.String)`
`weblogic.xml`	`xpath`	`*XMLNodeXPath.*`
`weblogic.xml.sec urity.specs`	`SecuritySpec`	`java.lang.String getName()`

Table B-8 Removed APIs (Not Deprecated) (Continued)

Package Name	Interface/Class/Exception	Method/Variable
`weblogic.xml.sec urity.wsse`	`SecurityElementFactory`	`weblogic.xml.security.wsse.T oken createToken(java.lang.String , java.lang.String, javax.xml.namespace.QName)`
	`UsernameToken`	`String getPasswordType()`

1. API functionality is supported by the `java.util.logging.Handler` class.

WebLogic Domain Directory Structure Enhancements

As of 9.0, WebLogic Server offers the following enhancements to the structure of the WebLogic domain directory:

- To improve configuration management and promote XML file validation, WebLogic Server supports the specification of domain configuration data in multiple files, including `config.xml`, in the new *domain_name*/`config` directory. (Here *domain_name* specifies the domain directory.) In previous releases, the `config.xml` file was the repository for all configuration information. Now, new subdirectories of `config` maintain configuration modules for diagnostic, JDBC, JMS, Node Manager, and security subsystems. Each configuration file adheres to an XML Schema definition.

- Startup and shutdown scripts are maintained in the *domain_name*/`bin` directory. In previous releases, they were stored in the root directory of the domain.

In addition to the structural enhancements to the domain directory, WebLogic Server supports utilities for managing changes to server configuration. These tools enable you to implement a secure, predictable means for distributing configuration changes in a domain. For more information, see "Managing Configuration Changes" in *Understanding Domain Configuration* at `http://e-docs.bea.com/wls/docs100/domain_config/changes.html`.

For more details about the WebLogic Server domain directory structure, see "Domain Configuration Files" in *Understanding Domain Configuration* at `http://e-docs.bea.com/wls/docs100/domain_config/config_files.html`.

Figure C-1 compares the domain directory structures for WebLogic Server 10.0 and WebLogic Server 8.1.

Figure C-1 WebLogic Server 10.0 and WebLogic Server 8.1 Directory Structures
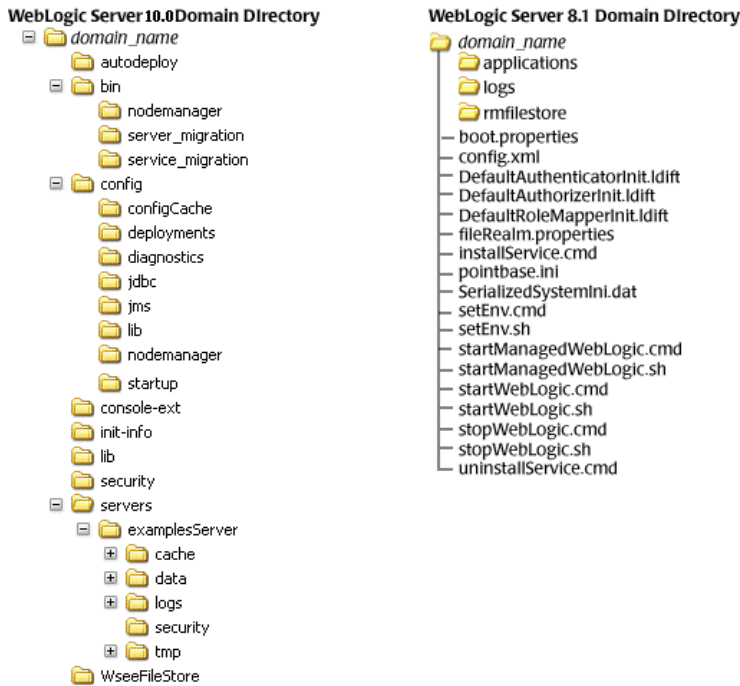
The following sections describe the domain directory structures for WebLogic Server 8.1 and 7.0.

WebLogic Server 8.1 Domain Directory Structure

In WebLogic Server 8.1 environments, the domain directory structure created by the Configuration Wizard contains:

- A domain root directory with the same name as the domain, such as `mydomain` or `petstore`. This directory contains the following:

  - `config.xml` file for the domain

  - Scripts used to start server instances and establish the environment

  - Subdirectory for storing applications for the domain, typically named `applications`.

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

- Files containing security information

- `logs` directory for storing domain-level logs

- For each server running in the domain, a directory for storing server logs and HTTP access logs

For more details about the WebLogic Server 8.1 domain directory structure, see "Overview of WebLogic Server Domains" in *Configuring and Managing WebLogic Server* at http://e-docs.bea.com/wls/docs81/adminguide/overview_domain.html. For a summary of the directory structure contents for the default configuration templates, see "Template Reference" in *Creating and Configuring WebLogic Domains Using the Configuration Wizard* at http://e-docs.bea.com/platform/docs81/confgwiz/tempref.html.

WebLogic Server 7.0 Domain Directory Structure

In WebLogic Server 7.0 environments, the root of the domain directory structure is a directory with the same name as the domain, such as `mydomain` or `petstore`. This directory should contain the following:

- `config.xml` file for the domain

- Scripts you use to start server instances and establish your environment

- For storing applications for the domain, a directory, typically named `applications`.

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

- `data` directory for storing security information

- `logs` directory for storing domain-level logs

- For each server running in the domain, a *server_name* directory for storing server-level logs

- `temp` directory for storing temporary files

The following sample tree shows where the directories are created in a domain directory called `mydomain`.
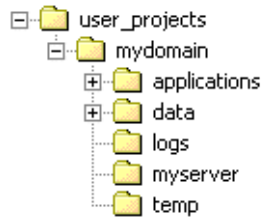
For more information about the WebLogic Server 7.0 domain directory structure, see "Domain Directories Structure" in "Domain Restrictions" in "Overview of WebLogic Server Domains" in *Creating and Configuring WebLogic Server Domains* at http://e-docs.bea.com/wls/docs70/admin_domain/overview.html.

Upgrade Wizard Command-Line Reference

To invoke the WebLogic Upgrade Wizard, enter the following command:

```
java weblogic.Upgrade [-help | -h | -? | -usage] [-type type] [-mode mode]
[-responses xmlfile] [-out file]
```

The following table describes each of the arguments included in this command line.

Table D-1 Command-Line Arguments for the WebLogic Upgrade Wizard

Argument	Description
-help | -h | -? | -usage	Displays help information.
-type *type*	Specifies one of the following types of upgrade: domain, nodemanager, or securityproviders. The default value is domain.
	Note: The -type domain option can only be used to upgrade a WebLogic Server domain in silent mode. For more information, see "Upgrading a Domain in Silent Mode" on page 5-11.
-mode *mode*	Specifies one of the following modes of upgrade: gui or silent. The default value is gui.

Table D-1 Command-Line Arguments for the WebLogic Upgrade Wizard (Continued)

Argument	Description
`-responses` *xmlfile*	Specifies the location of the XML file that contains the upgrade requirements to be used during a silent upgrade. This argument is valid only when `-mode` is set to `silent`. For more information about the format of the responses file, see "Silent Upgrade XML Script Reference" on page E-1.
`-out` *file*	Specifies the file in which standard output and errors are stored. By default, log output is sent to `stdout`.

Examples

- The following command upgrades a domain in interactive mode:

  ```
  java weblogic.Upgrade
  ```

- The following command upgrades a domain in silent mode:

  ```
  java weblogic.Upgrade -mode silent -type domain
  ```

Silent Upgrade XML Script Reference

Before using the WebLogic Upgrade Wizard in silent mode, you have the option of creating an XML script that defines your upgrade requirements, and passing that script to the wizard on the command line.

When run in silent mode, the Upgrade Wizard searches the domain's root directory for an XML script with a name that indicates the type of upgrade task to be performed, as listed in Table E-1. If the wizard does not locate an XML script, it uses the default system values, as defined in Table E-2.

Note: Only WebLogic Server domains can be upgraded using silent mode.

Table E-1 XML Scripts for Silent Upgrade

If you are upgrading ...	The wizard looks for an XML script named ...

Security Provider	`weblogic-upgrade-securityproviders-responses.xml`
	Sample:
	`http://e-docs.bea.com/common/docs100/upgrade/scripts/`
	`weblogic-upgrade-securityproviders-responses.xml`

Table E-1 XML Scripts for Silent Upgrade (Continued)

If you are upgrading ...	The wizard looks for an XML script named ...
Node Manager	`weblogic-upgrade-nodemanager-responses.xml` **Sample:** http://e-docs.bea.com/common/docs100/upgrade/scripts/ weblogic-upgrade-nodemanager-responses.xml
Domain	`weblogic-upgrade-domain-responses.xml` **Sample:** http://e-docs.bea.com/common/docs100/upgrade/scripts/ weblogic-upgrade-domain-responses.xml

To create an XML script for a silent mode upgrade:

1. In a supported browser, click on one of the sample XML scripts provided in Table E-1, based on the type of upgrade task you wish to perform.

2. Save the sample script, as an XML file of the same name, in the root directory of the domain that you want to upgrade.

3. In the XML script that you saved to your system, edit the values for the keywords shown in Table E-2 to match your requirements. If you do not specify a value for a keyword, the wizard uses the default value.

   Note: When editing the file, please note the following:

   - The XML definition (`<?xml version="1.0" encoding="UTF-8"?>`) must be at the very beginning of the XML script. No spaces or line breaks are allowed before the XML definition.

   - You must follow XML guidelines for characters when modifying values. That is, you cannot use characters reserved for use in XML, such as `<`, `>`, `[`, and `]`.

Table E-2 Silent-mode XML Script Values

If you are upgrading...	For this keyword...	Set the value to the...	This keyword defaults to the...
Security Providers	INPUT_DIRECTORY	Path of the directory that contains the security provider JARs to be upgraded. By default, security providers are located in `WL_HOME\server\lib\mbeantypes`, where `WL_HOME` specifies the root directory of the pre-10.0 installation.	Path of directory from which the Upgrade Wizard is run, such as: `c:\bea\wlserver_10.0\server\lib\mbeantypes`
	OUTPUT_DIRECTORY	Path of the directory in which you want to save the new security provider JAR files.	`WL_HOME\server\lib\mbeantypes`, where `WL_HOME` specifies the root directory of the WebLogic Server 10.0 installation.
Node Manager	NODE_MANAGER_HOME	Path of the directory of Node Manager to be upgraded by navigating the local directory hierarchy. By default, the Node Manager home directory is located in `WL_HOME\common\nodemanager`, where `WL_HOME` specifies the root directory of the pre-10.0 installation.	Directory from which the Upgrade Wizard is run. For example: `c:\bea\wlserver_10.0\common\nodemanager`

Table E-2 Silent-mode XML Script Values (Continued)

If you are upgrading...	For this keyword...	Set the value to the...	This keyword defaults to the...
Domains	WEBLOGIC_VERSION	Version of WebLogic Server.	Version of WebLogic Server identified in the configuration file (config.xml). For example, 8.1.4.0. If the software version number is not specified in the domain configuration file, the wizard displays the version number as 8.1.0.0, by default. In this case, there is no impact if the default value does not match the actual version number of the pre-10.0 domain.
	DOMAIN_DIR	Path of the directory that contains the WebLogic domain to be upgraded.	Directory from which the Upgrade Wizard is run, such as: c:\bea\user_project s\domains\mydomain
Domains (Continued)	NODE_MANAGER_USERNA ME	Username for Node Manager.	weblogic
	NODE_MANAGER_PASSWO RD	Password for Node Manager.	weblogic
	NODE_MANAGER_PASSWO RD	Confirmation password for Node Manager.	weblogic

Table E-2 Silent-mode XML Script Values (Continued)

If you are upgrading...	For this keyword...	Set the value to the...	This keyword defaults to the...
Domains (Continued)	OPTIONAL_ACTION_1	One or more of the following options: • `DOMAIN_DIRECTORY_BACKUP_SELECTED_VALUE` If this option is selected, the wizard backs up the original domain directory and stores it in a zip file. • `DOMAIN_DIRECTORY_BACKUP_LOG_FILES_INCLUDED_SELECTED_VALUE` If this option is selected, log files are included in the backup zip file. The number and size of log files can be large, so you may want to exclude them from the backup file by disabling this option. By default, log files are included in the backup file. • `SKIP_BACKWARDS_COMPATIBILITY_FLAGS_SELECTED_VALUE` Some behavior supported in pre-9.0 releases of WebLogic Server has been changed as of 9.0 to comply with J2EE 1.4. By default, the wizard sets flags to enable the previous behavior in the new domain. If you select this option, these flags or backward compatibility are not set.	`DOMAIN_DIRECTORY_BACKUP_SELECTED_VALUE`, `DOMAIN_DIRECTORY_BACKUP_LOG_FILES_INCLUDED_SELECTED_VALUE`

Table E-2 Silent-mode XML Script Values (Continued)

If you are upgrading...	For this keyword...	Set the value to the...	This keyword defaults to the...
Domains (Continued)	BACKUP_DIR	Path of the directory in which you want to save the backup zip file.	Directory from which the Upgrade Wizard is run, such as: c:\bea\user_project s\domains\mydomain
	BACKUP_FILE_NAME	Name of the backup zip file.	weblogic-domain-bac kup-mydomain.zip

Upgrading a Domain at Administration Server Startup (Implicit Mode)

The implicit mode of upgrade enables you to upgrade a WebLogic Server domain automatically at server startup. This mode should be used only in a development environment, and it is valid only for the machine on which the Administration Server resides. The implicit mode of upgrade is not recommended for use in a production environment.

Note: The implicit mode of upgrade can only be used to upgrade a WebLogic Server domain.

Before proceeding, make sure that you have performed the prerequisite steps described in "Prepare to Upgrade" on page 2-5.

To start the WebLogic Upgrade Wizard in implicit mode:

1. Verify that the WebLogic domain is not running.

2. Open an MS-DOS command prompt window (on Windows) or a command shell (on UNIX) and set up the environment as described in "Step 6: Set Up the Environment" on page 2-8.

3. Update the startup scripts in the domain to reference the WebLogic Server 10.0 installation.

   For example, make sure `WL_HOME` is set to the WebLogic Server 10.0 installation, as in the following example: `c:\bea\wlserver_10.0`.

   By default, the WebLogic Upgrade Wizard prompts you for confirmation before starting the upgrade process. You can disable the prompt by specifying, in the startup script, the following command argument for starting the server:
   `-Dweblogic.ForceImplicitUpgradeIfNeeded=true`

4. Start up the Administration Server in your domain.

   Respond to the upgrade confirmation prompt, if necessary.

The WebLogic Upgrade Wizard checks whether the domain directory was generated using a valid version of WebLogic Server (7.0 or 8.1), and, if the result is true, it automatically upgrades the domain. This process occurs only once: the first time you start the Administration Server.

Note: Implicit upgrade is not available for WebLogic Server version 9.*x* to 10.0 because upgrade is optional between these two versions. You can run a 9.*x* domain under 10.0 without modifications.

WebLogic Server Rolling Upgrade

What is Rolling Upgrade?

Rolling Upgrade is the process of upgrading a running WebLogic Server cluster with a patch, maintenance pack, or minor release without shutting down the entire cluster or domain. During the rolling upgrade of a cluster, each server in the cluster is individually upgraded and restarted while the other servers in the cluster continue to host your application.

Prior to the WebLogic Server 9.2 release, to upgrade a cluster, you had to either shutdown the entire domain, or install the upgraded server version on a parallel domain on additional hardware and use a load balancer to transfer load from the old domain to the new domain. The rolling upgrade process minimizes downtime and allows you to install a patch, maintenance pack, or minor release while the domain is still running.

This document provides information about the following topics:

- Tasks you need to do Before you Begin
- Rolling Upgrade Process
- Rolling Uninstall
- Limitations

Scope of the Rolling Upgrade Process

Support for rolling upgrade was available starting with WebLogic Server 9.2. The scope of rolling upgrade support encompasses installation of patches and maintenance packs as they are made available for WebLogic Server 10.0.

Rolling upgrade is most applicable to a cluster of WebLogic Server instances, however, you can also install updates to a domain of Managed Servers that are not in a cluster. This documentation focuses on installing upgrades in a cluster.

You can also uninstall a patch, maintenance pack, or minor release in a rolling fashion.

You can still upgrade to the latest patch using other strategies including bringing applications down in order to perform WebLogic Server upgrades, and restarting applications after WebLogic Server upgrades are complete. The rolling upgrade process provides you with an option of upgrading a running WebLogic Server cluster without shutting down the entire cluster or domain.

Before you Begin

Before you begin the upgrade process, make sure that you take any necessary precautions such as:

- Back up your applications, database schema, other application data, and domains.

- If required, obtain latest third party plug-ins or components that are compatible with required non-BEA components. For example, Apache Web Server libraries.

Rolling Upgrade Process

The rolling upgrade process includes stopping the Administration Server, installing the upgrade, restarting the Administration Server, and then doing the same for each managed server in the cluster. See the following sections for more information:

1. Quiescing and Stopping Servers

2. Installing a Patch, Maintenance Pack, or Minor Version

3. Restarting Servers

Quiescing and Stopping Servers

Before you install the WebLogic Server update on a server, you must first shut down the server. Prior to server shutdown, and depending on your environment, you may need to first stop load

balancers or Web Servers from sending requests or traffic to the server, complete any pending processes and then gracefully shut down the server.

Shutdown Servers in a Cluster

You can shut down a Managed Server from the command line, in a WLST script, or from the Administration Console.

From the command line, you can gracefully shut down the server using the `Graceful Shutdown` command. This command waits for all the in-process work to be completed before shutting down the server or cluster.

For information about using the `Graceful Shutdown` command, see shutdown in WebLogic Scripting Tool (WLST). For information about how to shut down the Managed Server from the Console, see Shutdown servers in a cluster in *Administration Console Online Help*.

Notes:

- When using the WLST `shutdown()` command, make sure that you are connected to the Managed server instance that you want to stop. For information about using WLST to shut down servers, see Managing Servers and Server Life Cycle in *WebLogic Scripting Tool.*

- Make sure that you stop all server instances that use the same installed WebLogic Server files.

Installing a Patch, Maintenance Pack, or Minor Version

After you stop the running server on your machine, install the maintenance upgrade. For more information, see Downloading and Applying Patches in *Installing Maintenance Updates and Service Packs*.

There are multiple methods for installing the patch, maintenance pack, or minor release. The following sections provide information about each of these options.

Installing Using Smart Update

You can use the Smart Update feature to periodically check for available software updates. When you start Smart Update, it checks the version numbers of the products installed (associated with the current BEA Home directory), and then connects to the BEA Web site to check for available service packs. For more information about installing using Smart Update, see *Installing Maintenance Updates and Service Packs*.

Command Line Interface

In most cases, the maintenance upgrade can be distributed and installed using a script. You can create a mechanism for replicating a specific maintenance level of a BEA product that is installed on multiple machines. This capability is especially valuable in production environments, in which the distribution of software updates to machines must be implemented in a controlled, reliable, and reproducible manner.

You can use the Smart Update bsu command, which you can use to apply patches, interactively or via script, that have been downloaded into a patch download directory. For more information, see Using the Command-Line Interface in *Installing Maintenance Updates and Service Packs*.

Silent Installation

Silent-mode installation is a way of setting installation configurations only once and then using those configurations to duplicate the installation on many machines. During installation in silent mode, the installation program reads the settings for your configuration from an XML file that you create prior to beginning the installation. The installation program does not display any configuration options during the installation process. Silent installation applies only to maintenance packs and minor version installation. For more information, see Running the Installation Program in Silent Mode in *BEA Products Installation Guide.*

Additional Information

- For complete information about installing BEA Products, see *BEA Products Installation Guide*.

- For sample scenarios and best practices during upgrade, see Best Practices for Distributing Maintenance Updates in *Installing Maintenance Updates and Service Packs.*

Restarting Servers

After you install the maintenance upgrade, you may need to modify start scripts before you restart your server. For more information see, Inserting Patches into the System Path in *Installing Maintenance Updates and Service Packs*. In addition, other post install tasks that you need to do will vary depending on your environment and the type of maintenance that you installed.

For an overview of methods for starting and stopping server instances, see Starting and Stopping Servers in *Managing Server Startup and Shutdown.*

If required, you may need to reconfigure your Web server or load balancer after the server is started so that requests are again sent to the server.

Steps in Rolling Upgrade Process for a Patch and Maintenance Pack

A patch is a file containing a fix or a small number of fixes that you can install using Smart Update. A patch is typically created to fix a software defect. A patch can be installed using Smart Update (recommended) or by referencing the patch at the beginning of the classpath.

A maintenance pack is a group of fixes combined into one file. Maintenance packs can be installed using Smart Update or by downloading an installer from the support Web site. Both maintenance packs and patches are applied to an existing installation.

While the WebLogic Server domain is running, follow these steps:

1. Shutdown the Administration Server. For more information, see "Quiescing and Stopping Servers" on page G-2.

2. On the machine that hosts the Administration Server, install the WebLogic Server patch. For more information, see "Installing a Patch, Maintenance Pack, or Minor Version" on page G-3.

3. Restart the Administration Server. For more information, see "Restarting Servers" on page G-4.

4. For each managed server, you need to do the following tasks:

   a. Quiescing and Stopping Servers

   b. Installing a Patch, Maintenance Pack, or Minor Version

   c. Restarting Servers

      Note: You can also modify the start scripts using Smart Update to automate the restart process. For more information, see Modifying Start Scripts in *Installing Maintenance Updates and Service Packs*.

You have now completed the upgrade process.

Steps in Rolling Upgrade Process for Minor WebLogic Server Releases

A minor version is a new release that includes fixes and new features. A minor release is installed in a completely new directory. Existing domains from the same release family can be run using the new minor release. WebLogic Server WebLogic Server 9.1 and WebLogic Server 9.2 are existing examples of minor versions of WebLogic Server 9.*x*.

While the WebLogic Server domain is running, follow these steps:

1. Shut down the Administration Server. For more information, see "Quiescing and Stopping Servers" on page G-2.

2. On the machine that hosts the Administration Server, install the new version of WebLogic Server. For more information about installing BEA Products, see *BEA Products Installation Guide*.

3. Update your start scripts for the domain so that the new files are in your classpath.

4. Update your environment variables such as, JAVA_HOME, BEA_HOME, and WLS_HOME, to point to the new version of WebLogic Server.

5. Restart the Administration Server. For more information, see "Restarting Servers" on page G-4.

6. For each managed server, you need to do the following tasks:

    a. Quiescing and Stopping Servers

    b. Installing a Patch, Maintenance Pack, or Minor Version

    c. Updating your start scripts for the domain so that the new files are in your classpath.

    d. Updating your environment variables such as, JAVA_HOME, BEA_HOME, and WLS_HOME, to point to the new version of WebLogic Server.

    e. Restarting Servers

Rolling Uninstall

You can uninstall a patch, maintenance pack, or minor release without shutting down the entire cluster or domain. The following sections provide more information.

Uninstalling a Patch or Maintenance Pack

At times you may want to uninstall a maintenance upgrade. You can use Smart Update to revert your system installation to an earlier version of the release. For more information, see Uninstalling Service Packs and Patches in *Installing Maintenance Updates and Service Packs*.

The procedure for uninstalling a patch or maintenance pack is in the opposite order of the rolling upgrade:

1. On each managed server, you need to do the following tasks:

    a. Quiescing and Stopping Servers.

    b. Uninstalling the patch or maintenance pack.

       For more information, see Uninstalling Service Packs and Patches in *Installing Maintenance Updates and Service Packs.*

    c. Restarting Servers.

2. After all managed servers have been downgraded, do the following tasks on the Administration Server:

    a. Quiescing and Stopping Servers.

    b. Uninstalling the patch or maintenance pack.

    c. Restarting Servers.

The applied patch or maintenance pack is now uninstalled.

Uninstalling a Minor Release

For information about uninstalling a minor release, see Uninstalling the Software in *BEA Products Installation Guide*.

The procedure is in the opposite order of the rolling upgrade:

1. On each managed server, do the following tasks:

    a. Quiescing and Stopping Servers.

    b. Uninstalling the patch or maintenance pack.

       For more information, see Uninstalling the Software in *BEA Products Installation Guide.*

    c. Updating the start scripts so that the previous version of WebLogic Server is in the classpath and all environment variables refer to the previous version of WebLogic Server.

    d. Restarting Servers.

2. After all managed servers have been downgraded, do the same steps on the Administration Server:

    a. Quiescing and Stopping Servers.

    b. Uninstalling the patch or maintenance pack.

    c. Updating the start scripts so that the previous version of WebLogic Server is in the classpath and all environment variables refer to the previous version of WebLogic Server.

    d. Restarting Servers.

The new version of WebLogic Server is now uninstalled.

Limitations

- Rolling upgrade applies only to upgrades within a product family. For example, you can upgrade from 9.*x* to 9.*y* but cannot upgrade from 9.*x* to 10.*x*.

- When WebLogic Server is installed on a machine and multiple Managed Servers are run from this same installation, you must shut down all Managed Servers that use the same installation before you can upgrade.

  During the upgrade process, certain installed files may need to be replaced. However, because the WebLogic Server instance uses these installed files while it is running, the files can be replaced during the upgrade process only after you shutdown the server instance. If multiple managed servers share the same installation of WebLogic Server, all instances that use the set of files must be shut down and upgraded at the same time. In support of rolling upgrade, BEA recommends that each managed server have its own installation of WebLogic Server.

- During the upgrade, you can use new features only after the entire domain has been upgraded.

  You should not make configuration changes during the upgrade process until all the servers in the cluster have been upgraded. This is especially true for new configuration options. Servers will silently ignore settings that they do not understand, and the local configuration file may not be updated properly. Also, using new configuration options may prohibit the capability of uninstalling a maintenance upgrade in a rolling fashion.

- For a minor release, during the rolling upgrade, there must be two entirely separate installation directories. That is, the location of the old installation and the location of the new installation must be two different directories.