



BEA Products

Upgrading WebLogic Application Environments

Copyright

Copyright © 1995-2005 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software is protected by copyright, and may be protected by patent laws. No copying or other use of this software is permitted unless you have entered into a license agreement with BEA authorizing such use. This document is protected by copyright and may not be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form, in whole or in part, without prior consent, in writing, from BEA Systems, Inc.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA SYSTEMS DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE DOCUMENT IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks and Service Marks

Copyright © 1995-2005 BEA Systems, Inc. All Rights Reserved. BEA, BEA JRockit, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop, Built on BEA, Jolt, JoltBeans, SteelThread, Top End, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA AquaLogic, BEA AquaLogic Data Services Platform, BEA AquaLogic Enterprise Security, BEA AquaLogic Service Bus, BEA AquaLogic Service Registry, BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA MessageQ, BEA WebLogic Commerce Server, BEA WebLogic Communications Platform, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Java Adapter for Mainframe, BEA WebLogic JDriver, BEA WebLogic Log Central, BEA WebLogic Network Gatekeeper, BEA WebLogic Personalization Server, BEA WebLogic Personal Messaging API, BEA WebLogic Platform, BEA WebLogic Portlets for Groupware Integration, BEA WebLogic Server Process Edition, BEA WebLogic SIP Server, BEA WebLogic WorkGroup Edition, Dev2Dev, Liquid Computing, and Think Liquid are trademarks of BEA Systems, Inc. BEA Mission Critical Support, BEA Mission Critical Support Continuum, and BEA SOA Self Assessment are service marks of BEA Systems, Inc.

All other names and marks are property of their respective owners.

Contents

1. Introduction

Overview of the Upgrade Process	1-2
How the Upgrade Wizard Simplifies the Upgrade Process	1-4
View a Demo: Upgrading an 8.1 WebLogic Application Environment to 9.1	1-4
Interoperability and Compatibility with Previous Releases.	1-5

2. Roadmap for Upgrading Your Application Environment

Plan the Upgrade	2-1
Step 1: Inventory the Application Environment	2-2
Step 2: Verify Supported Configuration Information	2-3
Step 3: Review the Compatibility Information.	2-4
Step 4: Create an Upgrade Plan	2-5
Prepare to Upgrade.	2-5
Step 1: Shut Down Servers in the Application Environment	2-5
Step 2: Back Up the Application Environment.	2-6
Step 3: Install WebLogic Server 9.1	2-7
Step 4: Prepare the Remote Managed Server Domain Directories.	2-7
Step 5: Set Up the Environment	2-8
Upgrade Your Application Environment.	2-9
Complete Post-Upgrade Procedure	2-11
Step 1: Upgrade Your Application Infrastructure	2-11
Deprecated Scripting Tools	2-12

Using the WebLogic Scripting Tool	2-12
Upgrading Your Custom Configuration Templates	2-13
Using SNMP to Monitor WebLogic Server	2-13
Step 2: Customize Startup Scripts	2-14
Default Startup Scripts	2-14
Custom Startup Scripts	2-14
Step 3: Verify File Permissions	2-15
Step 4: Enroll the Machine with Node Manager	2-15
Step 5: Verify Remote Server Startup Options	2-17
Step 6: Promote the Application Environment to Production	2-17
What to Do If the Upgrade Process Fails	2-18

3. Upgrading a Security Provider

What Happens During a Security Provider Upgrade	3-2
Upgrading a Security Provider	3-2
Upgrading a Security Provider in Graphical Mode	3-3
Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade a Security Provider	3-4
Procedure for Upgrading a Security Provider.	3-5
Upgrading a Security Provider in Silent Mode	3-8

4. Upgrading Node Manager

What Happens During a Node Manager Upgrade	4-1
Upgrading Node Manager.	4-2
Upgrading Node Manager in Graphical Mode	4-3
Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade Node Manager.	4-3
Procedure for Upgrading Node Manager	4-4
Upgrading Node Manager in Silent Mode.	4-6

5. Upgrading a WebLogic Domain

What Happens During a WebLogic Domain Upgrade?	5-1
Important Notes About the Domain Upgrade Process	5-3
Upgrading a Domain	5-5
Upgrading a Domain in Graphical Mode	5-6
Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade a Domain. 5-6	
Procedure for Upgrading a WebLogic Domain.	5-8
Upgrading a Domain in Silent Mode	5-19

A. Compatibility with Previous Releases

JMX 1.2 Implementation	A-2
Dynamic Configuration Management	A-3
Modular Configuration and Deployment of JDBC Resources	A-4
JDBC Data Sources and Connection Pools	A-5
MultiPools	A-6
Data Source Factories	A-6
JDBC Debugging Enhancements	A-6
Modular Configuration and Deployment of JMS Resources	A-6
JMS Message ID Format	A-8
Thread Management	A-9
JTA Transaction Log Migration	A-9
Security	A-10
XACML Security Providers	A-10
New SAML V2 Providers	A-11
Security MBeans	A-11
Password Encryption	A-12
Security for HTTP Requests	A-13

Secure Access to MBeanHome	A-14
Message-Level Security in Web Services	A-14
Web Services	A-14
Web Applications, JSPs, and Servlets	A-15
Deprecated and Obsolete Web Application Features	A-15
Support for JSP 2.0	A-15
Servlet Path Mapping	A-16
XML Implementation	A-17
XMLBeans and XQuery Implementation	A-18
WebLogic Administration and Configuration Scripts	A-19
Deprecated Deployment Features	A-19
Deployment Descriptor Validation and Conversion	A-19
Deprecated Startup and Shutdown Classes	A-20
Administration Console Extension Architecture	A-21
Resource Adapters	A-21
WLEC	A-23
Backward Compatibility Flags	A-23
Deprecated and Removed APIs	A-24
Deprecated APIs	A-25
Removed APIs	A-25
Removed APIs, Deprecated in WebLogic Server 8.1	A-25
Removed APIs, Deprecated in WebLogic Server 7.0	A-26
Removed APIs (Not Deprecated)	A-31

B. WebLogic Domain Directory Structure Enhancements

WebLogic Server 8.1 Domain Directory Structure	B-3
WebLogic Server 7.0 Domain Directory Structure	B-3
WebLogic Server 6.1 Domain Directory Structure	B-4

- C. Upgrade Wizard Command-Line Reference
- D. Silent Upgrade XML Script Reference
- E. Upgrading a Domain at Administration Server Startup (Implicit Mode)

Introduction

This document describes how to upgrade your application environment to WebLogic Server 9.1 from: WebLogic Server 6.1, 7.0, or 8.1.

Note: If you are upgrading from WebLogic Server 9.0 to WebLogic Server 9.1, see *Roadmap for Upgrading WebLogic and AquaLogic Application Environments* at <http://e-docs.bea.com/common/docs91/interm/upgrade.html>.

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

- [Overview of the Upgrade Process](#)
- [How the Upgrade Wizard Simplifies the Upgrade Process](#)
- [View a Demo: Upgrading an 8.1 WebLogic Application Environment to 9.1](#)
- [Interoperability and Compatibility with Previous Releases](#)

Important Terminology

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

- **Upgrade**—To update BEA WebLogic Server 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.

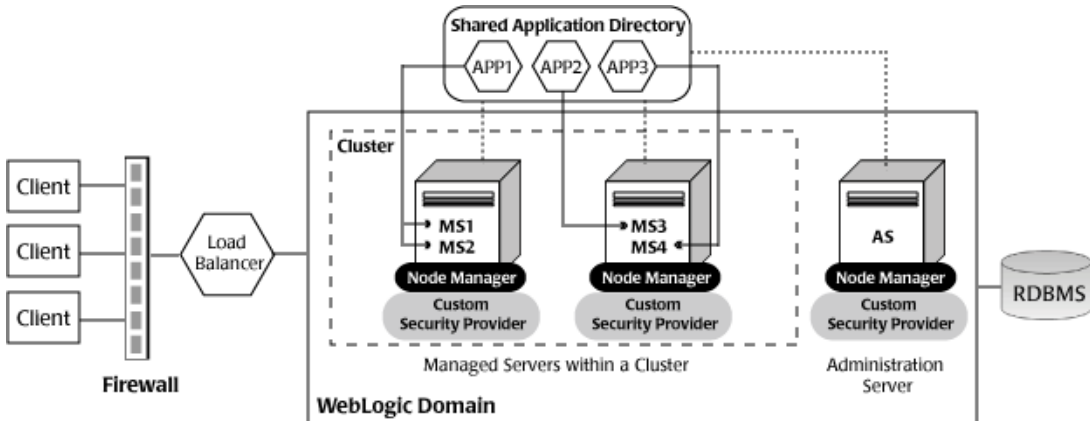
- **Interoperability**—(1) The ability of an application deployed in one release or service pack to communicate with another application that is deployed in a different release or service pack. (2) The ability of WebLogic Server components to communicate with third-party software via standard protocols.
- **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 <i>Understanding Domain Configuration</i> at http://e-docs.bea.com/wls/docs91/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 <i>Developing Security Providers for WebLogic Server</i> at http://e-docs.bea.com/wls/docs91/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 <i>Managing Server Startup and Shutdown</i> at http://e-docs.bea.com/wls/docs91/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 J2EE 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 9.1 application environment. For more information, see “Interoperability and Compatibility with Previous Releases” on page 1-5.
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 9.1.

Each step in the upgrade process is described in more detail in this document.

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 6.1, 7.0, or 8.1 such that it runs in a WebLogic Server 9.1 application environment. As part of the upgrade process, you must upgrade any custom security providers and Node Managers used in the domain.

You can step through the upgrade process interactively, using the graphical user interface (GUI), or “silently,” by creating an upgrade script and running it.

View a Demo: Upgrading an 8.1 WebLogic Application Environment to 9.1

View an interactive demo, [Upgrading an 8.1 WebLogic Application Environment to 9.1](#), that shows you how to upgrade a WebLogic Server application environment, built using WebLogic Server 8.1, so that it runs on WebLogic Server 9.1.

The 8.1 Avitek Medical Records sample application is used in this demo to illustrate the upgrade process. For more information about this sample application, see the *Avitek Medical Records Development Tutorials* at

http://e-docs.bea.com/wls/docs81/medrec_tutorials/index.html.

Interoperability and Compatibility with Previous Releases

Application environments that run with WebLogic Server 9.1 can interact with application environments built on WebLogic Server 7.0 or 8.1.

Most existing WebLogic Server applications can be run without modification in the new WebLogic Server 9.1 application environment. You should review the compatibility information described in [“Compatibility with Previous Releases” on page A-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](#)

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 ...	
<input type="checkbox"/>	Administration Server and the machine on which it resides
<input type="checkbox"/>	Managed Servers and the machine(s) on which they reside
<input type="checkbox"/>	Custom security providers used in the domain
<input type="checkbox"/>	Custom Node Managers used in the domain
<input type="checkbox"/>	Location of the applications (including all external client applications)
<input type="checkbox"/>	External resources, for example: <ul style="list-style-type: none">• Databases used to store persisted and application data• Firewalls• Load balancers
<input type="checkbox"/>	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	<p>“List of Supported Operating System Configurations” in “WebLogic Server 9.1 Supported Configurations” in <i>Supported Configurations for WebLogic Server 9.1</i> at http://e-docs.bea.com/platform/suppconfigs/configs91/91_over/overview.html.</p>
Database support	<p>“Supported Database Configurations” in <i>Supported Configurations for WebLogic Server 9.1</i> at http://e-docs.bea.com/platform/suppconfigs/configs91/91_over/supported_db.html.</p> <p>Please note the following:</p> <ul style="list-style-type: none"> • WebLogic Server 9.1 supports PointBase 5.1. <p>If you are using a pre-5.1 version of PointBase and you want to continue using that version, <i>after</i> you upgrade your domain 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 CLASSPATH statement in your setDomainEnv files.</p> <p>If you are using a pre-5.1 version of PointBase and you want to update your environment to use PointBase 5.1, <i>before</i> you upgrade your domain you must perform the following steps:</p> <p>Note: In the following description, <i>WL_HOME</i> represents the root directory of the WebLogic Server 9.1 installation, for example, <i>c:\bea\weblogic91</i> (on Windows).</p> <ol style="list-style-type: none"> 1) Update references to database names within configuration files. 2) Update the <i>WL_HOME\common\eval\pointbase\tools\pointbase.ini</i> file to include the following license information: <pre>pbembedded.lic=WL_HOME\common\eval\pointbase\lib\pbembedded.lic</pre> 3) 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: <pre>WL_HOME\common\eval\pointbase\lib\pbupgrade51.jar</pre> 4) Update your applications to use PointBase 5.1. Note that the PointBase 5.1 database software is installed as part of the 9.1 installation. See “Step 3: Install WebLogic Server 9.1” on page 2-7. • If you are upgrading a domain from WebLogic Server 6.1, please note that Cloudscape is no longer supported, and you will need to upgrade to a database that is supported for the 9.1 release.

Table 2-2 Verify Supported Configuration Information (Continued)

To verify ...	See ...
Database support (Continued)	<ul style="list-style-type: none"> • 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 <i>Supported Configurations for WebLogic Server 9.1</i> at http://e-docs.bea.com/platform/suppconfigs/configs91/91_over/supported_db.html. 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 <i>WebLogic Type 4 JDBC Drivers</i> at http://e-docs.bea.com/wls/docs91/jdbc_drivers/oracle.html and “Third-Party JDBC Drivers Installed with WebLogic Server” in <i>Configuring and Managing WebLogic JDBC</i> at http://e-docs.bea.com/wls/docs91/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 <code>server</code> property (as illustrated below) from the generated JDBC module. For example: <pre style="margin-left: 20px;"> <property> <name>server</name> <value>servername</value> </property> </pre>
Web servers, browsers, and firewalls	<p>“Supported Web Servers, Browsers, and Firewalls” in <i>Supported Configurations for WebLogic Server 9.1</i> at http://e-docs.bea.com/platform/suppconfigs/configs91/91_over/add-ons.html.</p>

Step 3: Review the Compatibility Information

Most existing WebLogic Server applications can be run without modification in the new WebLogic Server 9.1 application environment. However, you should review “[Compatibility with Previous Releases](#)” on [page A-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, for example, 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/docs91/ejb/message_beans.html.

Prepare to Upgrade

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

- [Step 1: Shut Down Servers in the Application Environment](#)
- [Step 2: Back Up the Application Environment](#)
- [Step 3: Install WebLogic Server 9.1](#)
- [Step 4: Prepare the Remote Managed Server Domain Directories](#)
- [Step 5: Set Up the Environment](#)

Step 1: Shut Down Servers in the Application Environment

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

Step 2: 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 5-15](#). 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	<p>Back up the Administration Server and any remote Managed Server domain directories that are defined in the application environment.</p> <p>By default, the Configuration Wizard creates a domain directory in the <i>BEA_HOME\user_projects</i> directory, where <i>BEA_HOME</i> specifies the directory in which you installed WebLogic Server.</p>
Applications and application-persisted data	<p>Back up any applications and data that reside outside the domain directory.</p> <p>By default, applications are created in the <i>BEA_HOME\user_projects\applications</i> directory, where <i>BEA_HOME</i> specifies the directory in which you installed WebLogic Server.</p>
Custom security providers	<p>Back up any custom security providers that you are using in your application environment.</p> <p>By default, security providers are located in <i>WL_HOME\server\bin\mbeantypes</i>, where <i>WL_HOME</i> specifies the root directory of the WebLogic Server installation.</p>

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

Component	Recommendations
Node Manager directory and scripts	<p>Back up any Node Manager directories and scripts that you are using to manage your servers in a clustered environment.</p> <p>The names of the directory and script depend on your operating system:</p> <ul style="list-style-type: none"> • Windows: <ul style="list-style-type: none"> <code>WL_HOME\common\nodemanager</code> <code>WL_HOME\server\bin\startNodeManager.cmd</code> • UNIX: <ul style="list-style-type: none"> <code>WL_HOME/common/nodemanager</code> <code>WL_HOME/server/bin/startNodeManager.sh</code> <p>In the preceding pathnames, <i>WL_HOME</i> represents the root directory of the WebLogic Server installation, for example, <code>c:\bea\weblogic81</code>.</p>
Log files	<p>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.</p>

Step 3: Install WebLogic Server 9.1

Before upgrading your application environment, you must install WebLogic Server 9.1 on each machine in the domain. For more information about installing WebLogic Server 9.1, see *Installation Guide* at <http://e-docs.bea.com/common/docs91/install/index.html>.

Note: If you are using Node Manager in your pre-9.1 installation, when installing the 9.1 product, you should set the Node Manager listen port to match the port number used in the pre-9.1 installation, if possible. The default value for the listen port for Node Manager is 5556.

Step 4: 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 9.1, 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 2: Verify Supported Configuration Information” on page 2-3](#).

Step 5: Set Up the Environment

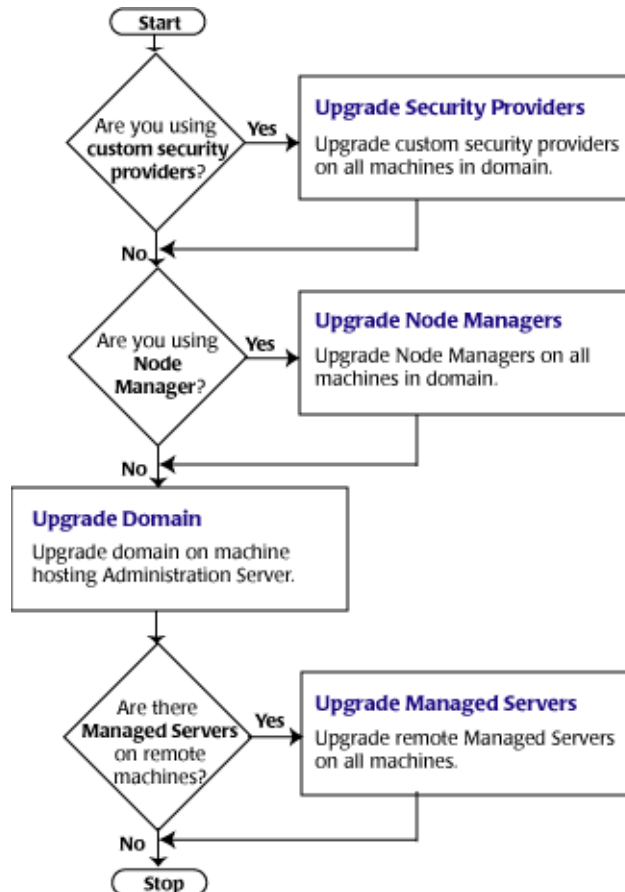
To set up the environment for an upgrade:

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
<p>1. Upgrade custom security providers</p>	<p>If are using custom security providers in your current application environment and you wish to continue using them in the new environment, upgrade them:</p> <ul style="list-style-type: none"> • On all machines in the environment. • Before you upgrade the WebLogic domain. <p>Note: If you are installing WebLogic Server 9.1 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, <i>WL_HOME</i>\server\lib\mbeatypes, where <i>WL_HOME</i> specifies the root directory of the pre-9.1 installation, are upgraded automatically. (Custom security providers were not supported in 6.1 environments.) 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.</p>
<p>2. Upgrade Node Managers</p>	<p>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:</p> <ul style="list-style-type: none"> • On all machines in the environment • Before you start the servers in an upgraded WebLogic domain
<p>3. Upgrade WebLogic domain (Administration Server)</p>	<p>Upgrade the WebLogic domain on the machine that hosts the Application Server.</p> <p>Note: BEA recommends that you upgrade the Administration Server for a domain before the Managed Servers.</p>
<p>4. Upgrade WebLogic domain (remote Managed Servers)</p>	<p>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 4: Prepare the Remote Managed Server Domain Directories” on page 2-7.</p> <p>Note: Managed Servers that reside on the same machine as the Administration Server do not require additional upgrade steps.</p>

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: Customize 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 9.1 environments. BEA recommends that you update your scripts to take advantage of the new features provided with WebLogic Server 9.1. For more information about new WebLogic Server features and changes in the MBean hierarchy, see “What’s New in WebLogic Server 9.1” in the *WebLogic Server and WebLogic Express Release Notes* at <http://e-docs.bea.com/wls/docs91/notes/new.html>.

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

- [“Deprecated Scripting Tools” on page 2-12](#)
- [“Using the WebLogic Scripting Tool” on page 2-12](#)
- [“Upgrading Your Custom Configuration Templates” on page 2-13](#)
- [“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:

- Silent-mode operation of the Configuration Wizard. For more information, see http://e-docs.bea.com/common/docs91/silent_cw/index.html.
- `weblogic.Admin` utility. For more information, see http://e-docs.bea.com/wls/docs91/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 9.1, the online and offline versions of WLST are delivered as a single tool. WLST fully supports the administrative and configuration features offered by 9.1. For more information about WLST, see *WebLogic Scripting Tool* at http://e-docs.bea.com/wls/docs91/config_scripting/index.html.

Note: WLST online can be run on WebLogic Platform 7.0 and 8.1, and WebLogic Server 9.0 and 9.1. WLST offline can be run on WebLogic Platform 8.1 SP5 and WebLogic Server 9.0 and 9.1.

As with the other pre-9.1 tools, BEA does not guarantee that existing WLST scripts will run in 9.1 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 9.1. For more information about new WebLogic Server features and changes in the MBean hierarchy, see "What's New in WebLogic Server 9.1" in the *WebLogic Server and WebLogic Express Release Notes* at <http://e-docs.bea.com/wls/docs91/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 2-9.
2. Modify the domain configuration to leverage new features, as appropriate.	See “What’s New in WebLogic Server 9.1” in the <i>WebLogic Server and WebLogic Express Release Notes</i> at http://e-docs.bea.com/wls/docs91/notes/new.html .
3. Use the Template Builder or <code>pack</code> command to create a 9.1 domain and/or extension template.	See: <ul style="list-style-type: none"> • <i>Creating Templates Using the Domain Template Builder</i> at http://e-docs.bea.com/common/docs91/tempbuild/index.html. • <i>Creating Domains and Templates Using the pack and unpack Commands</i> at http://e-docs.bea.com/common/docs91/pack/index.html.

Using SNMP to Monitor WebLogic Server

If you use an SNMP manager to monitor WebLogic Server:

1. Load the WebLogic Server 9.1 MIB into your SNMP manager.

The MIB is located at `BEA_HOME/weblogic91/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/docs91/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/docs91/issues/index.html>.

Step 2: Customize Startup Scripts

The following sections describe how to customize the default and 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.

Note: If you are upgrading your domain to WebLogic Server 9.1 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.

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 9.1 classes to the beginning of the variable.
 - Remove all *unused* pre-9.1 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/docs91/server_start/nodemgr.html#ReviewNMDomains.

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/docs91/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/docs91/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/docs91/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 9.1 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/docs91/ConsoleHelp/taskhelp/startstop/ConfigureStartupArgumentsForManagedServers.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[B]I)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 9.1 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 2: 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 2-9](#).

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 9.1 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 9.1 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 9.1* at <http://e-docs.bea.com/wls/docs91/security.html>.

Custom security providers were not supported in WebLogic Server 6.1.

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

- [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/docs91/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 9.1 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/docs91/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-9.1 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 9.1 environment.

Note: If you are installing WebLogic Server 9.1 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-9.1 installation, are upgraded automatically. (Custom security providers were not supported in 6.1 environments.) 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 9.1 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.

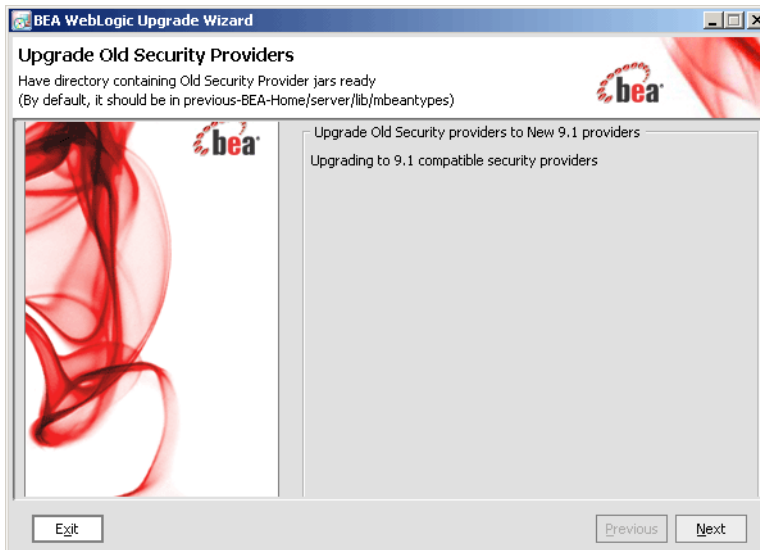
To start the WebLogic Upgrade Wizard in graphical mode and upgrade the security provider:

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 5: 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.



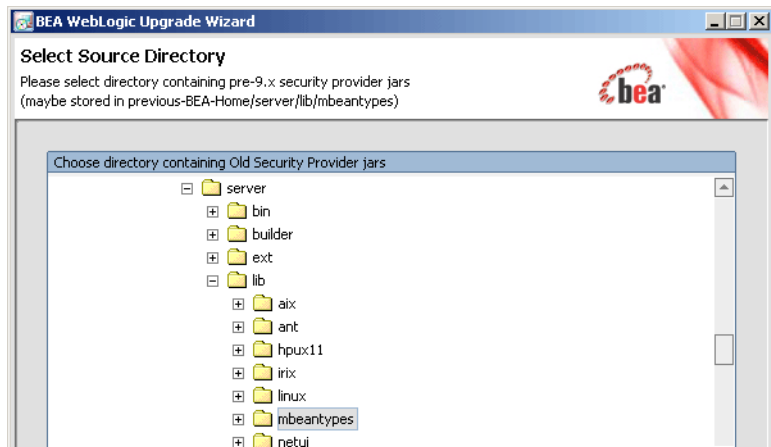
4. Click **Next** to proceed to the next window.

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	<p>Select the directory that contains the security provider JARs that need to be upgraded. By default, the selected directory is the current directory.</p> <p>By default, security providers are located in <code>WL_HOME\server\lib\mbeantypes</code>, where <code>WL_HOME</code> specifies the root directory of the pre-9.1 installation of WebLogic Server, as shown in the following example.</p>

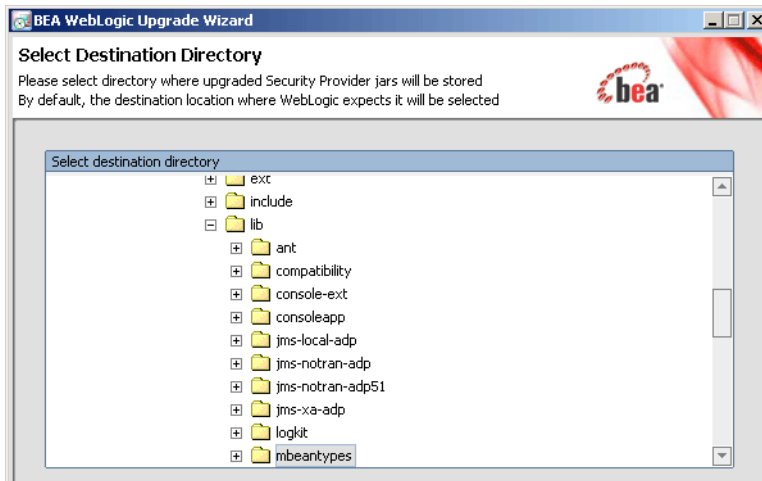


Note: The security provider JARs must contain the MBean Definition File (MDF) for the associated MBean. For more information about creating MDFs, see *Developing Security Providers for WebLogic Server* at <http://e-docs.bea.com/wls/docs91/dvspisec/index.html>. If JAR file does not contain an MDF, the upgrade process fails for the associated security provider.

Click **Next** to proceed to the next 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 <code>WL_HOME\server\lib\mbeantypes</code> , where <code>WL_HOME</code> specifies the root directory of the WebLogic Server 9.1 installation, as shown in the following example.



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.

Table 3-1 Procedure for Upgrading a Security Provider (Continued)

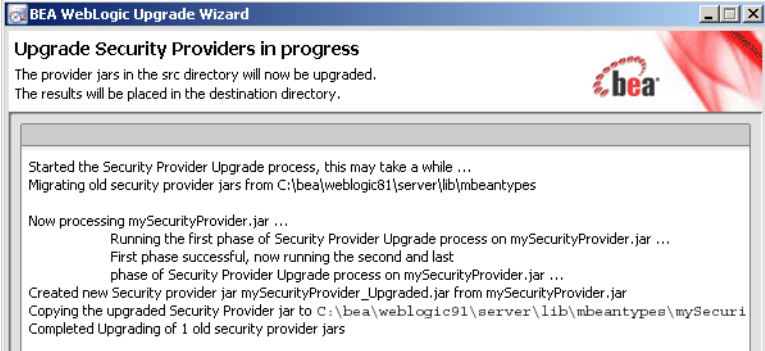
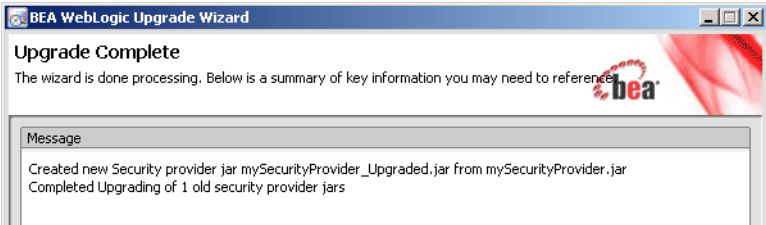
In this step ...	You ...
Upgrade Security Providers in Progress	Review progress of the wizard as it saves the upgraded JARs and deletes any temporary files that were created during the upgrade process. Progress messages are displayed in the window, as illustrated in the following illustration.
	
<p>The security provider JAR must contain the MBean Definition File (MDF) for the associated MBean. For more information about creating MDFs, see <i>Developing Security Providers for WebLogic Server</i> at</p>	
<p>http://e-docs.bea.com/wls/docs91/dvspisec/index.html. If a JAR file does not contain an MDF, the upgrade process fails for the associated security provider. For example:</p>	
<pre>Now processing mySecurityProviderToo.jar ...</pre>	
<pre>No MDFs (.xmls) found in the old security provider jar with name mySecurityProviderToo.jar</pre>	
<p>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.</p>	
<p>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:</p>	
<pre>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)</pre>	
<p>Click Next to proceed to the next window.</p>	

Table 3-1 Procedure for Upgrading a Security Provider (Continued)

In this step ...	You ...
Upgrade Complete	<p>Review the upgrade results, including any important messages that require further consideration, such as the messages shown in the following example.</p> <div data-bbox="364 465 1130 690" style="border: 1px solid gray; padding: 5px;">  </div> <p>Click Done to close the wizard.</p>

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 5: 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 D-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 D-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 a Security Provider

Upgrading Node Manager

If you are using a customized version of Node Manager in a pre-9.1 environment, you can use the WebLogic Upgrade Wizard to upgrade Node Manager for use in a WebLogic Server 9.1 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/docs91/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 9.1 application environment. The Upgrade Wizard upgrades the `nodemanager.properties` file and converts the `NodeManagerSerializedSystemIni.dat` to `nm_data.properties`.

- 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 5-11](#). The listen port number used by Node Manager in the pre-9.1 installation is maintained during the upgrade process.

Note: When installing the 9.1 product, you should, if possible, set the Node Manager listen port to the same port number used in the pre-9.1 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 9.1 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 5: Set Up the Environment” on page 2-8](#).
3. If the Node Manager directory resides in the pre-9.1 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 9.1 installation directory.

In this case, you will need to upgrade the *copy* Node Manager in the 9.1 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-9.1 installation directory, you can skip this step.

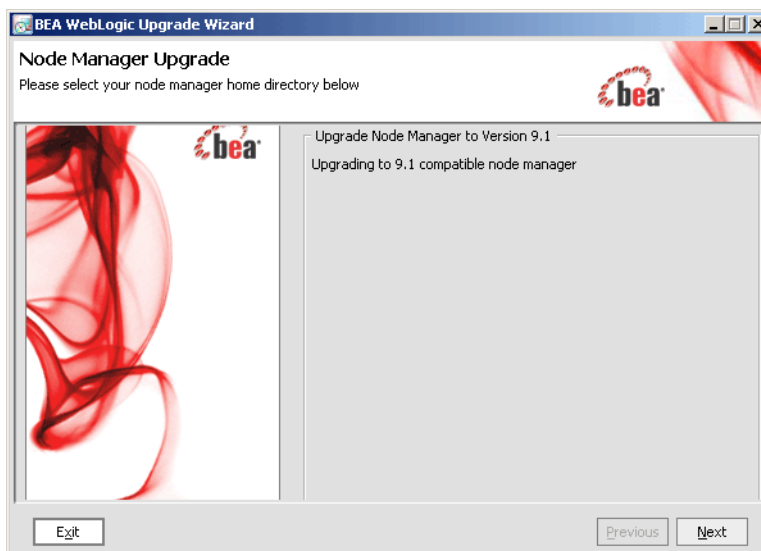
4. At a command prompt, enter the following command:

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

Upgrading Node Manager

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.



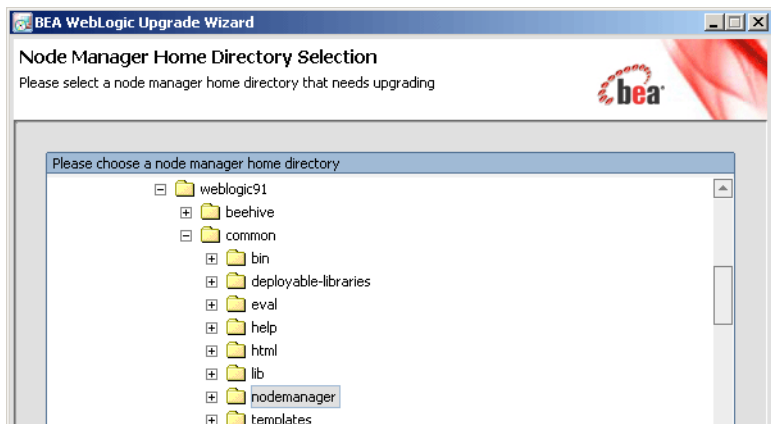
5. Click **Next** to proceed to the next window.

Procedure for Upgrading Node Manager

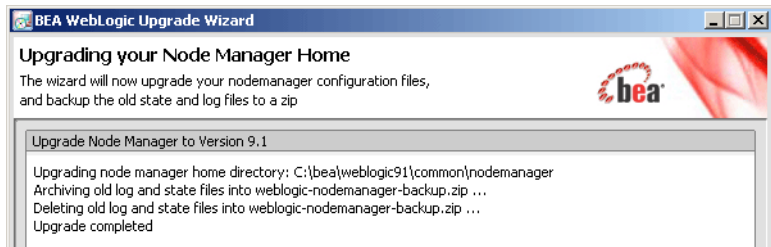
The following table summarizes the steps in the procedure to upgrade Node Manager using the WebLogic Upgrade Wizard.

Table 4-1 Procedure for Upgrading Node Manager

In this step ...	You ...
<p>Node Manager Home Directory Selection</p>	<p>Select a home directory for the instance of Node Manager to be upgraded by navigating the local directory hierarchy.</p> <p>The Node Manager home directory contains the <code>nodemanager.properties</code> file, logs, and other related files. The exact set of files generated by Node Manager varies between releases and service packs.</p> <p>By default, the Node Manager home directory is <code>WL_HOME/common/nodemanager</code>, where <code>WL_HOME</code> specifies the root directory of the WebLogic Server installation, as shown in the following example.</p>
<p>Upgrade Your Node Manager Home</p>	<p>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, as shown in the following example.</p>

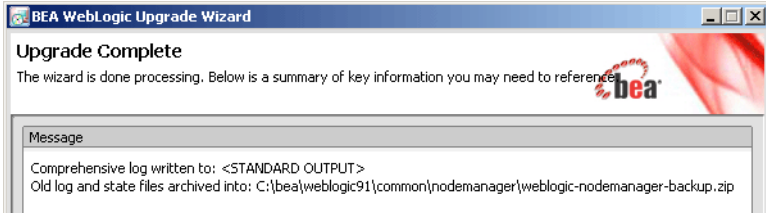


Click **Next** to proceed to the next window.



Once this process is complete, click **Next** to proceed to the next window.

Table 4-1 Procedure for Upgrading Node Manager (Continued)

In this step ...	You ...
<p>Upgrade Complete</p>	<p>Review the upgrade results, including any important messages that require further consideration, such as those shown in the following example.</p>
	
	<p>Click Done to close the WebLogic Upgrade Wizard.</p>
	<p>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.</p>

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 5: Set Up the Environment”](#) on page 2-8.
3. If the Node Manager directory resides in the pre-9.1 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 9.1 installation directory.

In this case, you will need to upgrade the *copy* of Node Manager in the 9.1 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-9.1 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 D-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 D-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 that is hosting the WebLogic domain with Node Manager, 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

If you have created a WebLogic domain that is compatible with WebLogic Server 6.1, 7.0, or 8.1, you can use the WebLogic Upgrade Wizard to upgrade the domain for a WebLogic Server 9.1 application environment.

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 2: 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-90-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 B-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 5: 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/docs91/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 used (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 *PrefixName*WLSTORE, 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, remove or clear 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:

- At a minimum, the domain directory must contain the following files:
 - `config.xml`
 - Security-related files, including `SerializedSystemIni.dat`, `DefaultAuthenticatorInit.ldif`, `DefaultAuthorizerInit.ldif`, and `DefaultRoleMapperInit.ldif`

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/docs91/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.
- When upgrading a WebLogic Server 6.1 domain, please note the following:
 - The `WL_HOME\config` directory must contain only one domain. If it contains more than one domain, the upgrade will fail.

- When a domain contains single-module applications (.war, .jar) and an application.xml file is not defined for the domain, make sure that the pathname for the application does not include URL for the module. For example, if the path is specified as c:\bea\wlserver6.1\config\petstore\applications\tour.war, the tour.war filename must be removed before you start the upgrade. Otherwise, the upgrade process returns an error indicating that the path is invalid.

If an error is encountered during the upgrade process and the application module is not deployed, the module is autodeployed as long as the server is running in development mode. Otherwise, you need to deploy the application module using the WebLogic administration tools.

- The wizard does not upgrade applications during a WebLogic domain upgrade. In most cases, WebLogic Server applications can be run without modifications in the new WebLogic Server 9.1 application environment. Review the compatibility information in [“Compatibility with Previous Releases” on page A-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.
- 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 9.1, you may need to specify a username and password when starting the server. As a result, you may notice the following behavioral changes:
 - In 6.1, the username defaulted to system, and you were required to specify a password only.
 - 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 9.1 domain. With 9.1, 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\weblogic81\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 9.1 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, using the graphical user interface.
- **Silent**—For upgrading a domain silently, by specifying upgrade requirements in a file.

Note: You can also use implicit mode, to upgrade a domain automatically when the Administration Server is started. For more information, see [“Upgrading a Domain at Administration Server Startup \(Implicit Mode\)”](#) on page E-1.

You must upgrade the domains on every machine in the domain. For information about preparing remote managed server domain directories, see [“Step 4: Prepare the Remote Managed Server Domain Directories”](#) on page 2-7.

The following sections provide instructions for both modes:

- [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 5: 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:

Note: Before proceeding, make sure you have performed the prerequisite steps described in [“Prepare to Upgrade” on page 2-5](#) and reviewed the important notes described in [“Important Notes About the Domain Upgrade Process” on page 5-3](#).

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 5: Set Up the Environment” on page 2-8](#).
3. At the command prompt, enter the following command:

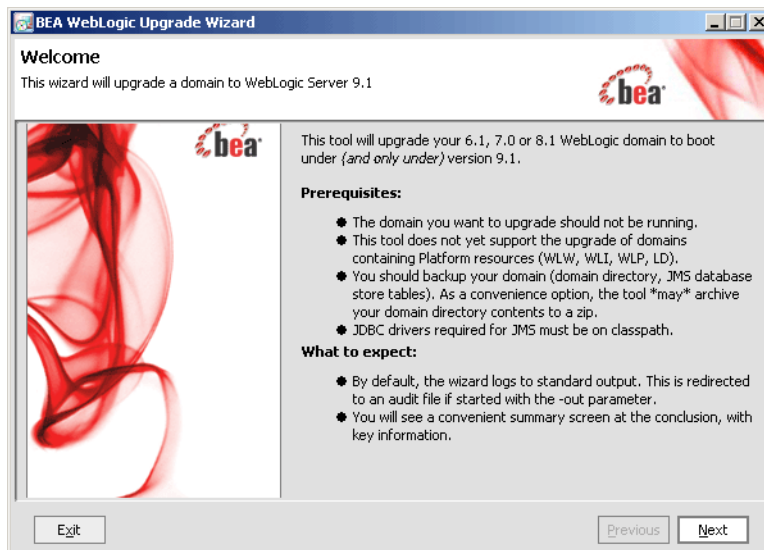
```
java weblogic.Upgrade [-type domain] [-out file]
```

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.



4. If JMS JDBC stores are used in the domain, you should start the corresponding databases.
5. 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.

Table 5-1 Procedure for Upgrading a WebLogic Domain

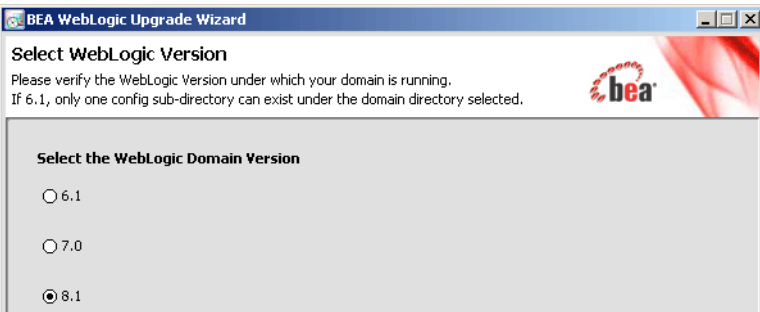
In this step ...	You ...
Select WebLogic Version	<p>Select the WebLogic version of the domain that you are upgrading, as shown in the following example.</p>  <p>Note: When upgrading a WebLogic Server 6.1 domain, the <code>WL_HOME\config</code> directory must contain only one domain. If it contains more than one domain, the upgrade will fail.</p> <p>Click Next to proceed to the next window.</p>

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

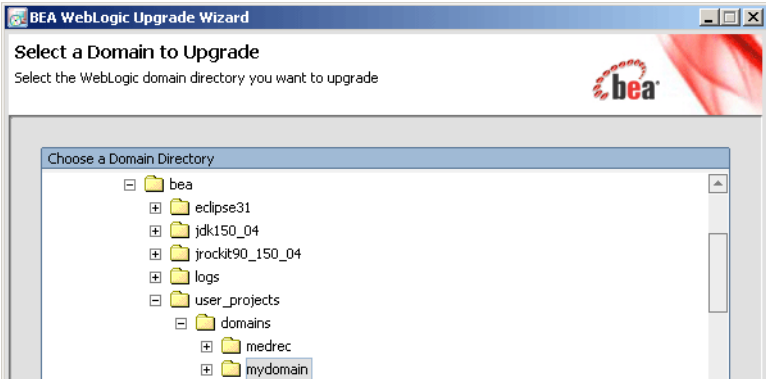
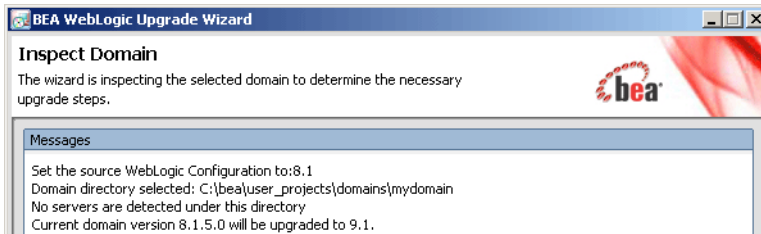
In this step ...	You ...
Select a Domain to Upgrade	Select the directory that contains the WebLogic domain to be upgraded by navigating the local directory hierarchy, as shown in the following example.
	 <p>The screenshot shows a window titled "BEA WebLogic Upgrade Wizard" with the subtitle "Select a Domain to Upgrade". Below the subtitle is the instruction "Select the WebLogic domain directory you want to upgrade". The main area is a "Choose a Domain Directory" dialog box showing a tree view of a file system. The tree structure is as follows:</p> <ul style="list-style-type: none"> bea <ul style="list-style-type: none"> eclipse31 jdk150_04 jrockit90_150_04 logs user_projects <ul style="list-style-type: none"> domains <ul style="list-style-type: none"> medrec mydomain (selected)
	<p>Note: When upgrading a WebLogic Server 6.1 domain, you should select the directory that contains the <code>config</code> directory. For example, if your domain resides in <code>c:/bea/user_projects/config/mydomain</code>, you should select <code>c:/bea/user_projects</code> as the directory of the WebLogic domain. If you select another directory, such as <code>c:/bea/user_projects/config/mydomain</code>, you will get unexpected results.</p> <p>Click Next to proceed to the next window.</p>

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

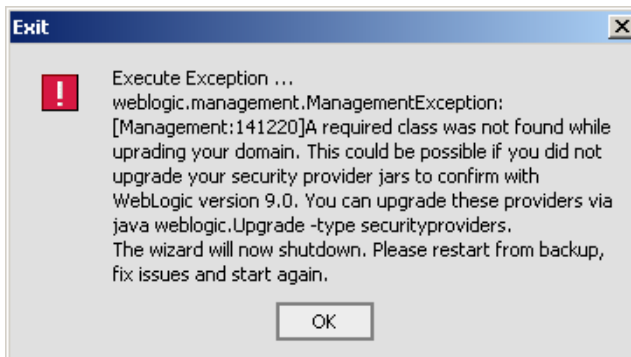
In this step ...	You ...
------------------	---------

Inspect Domain

Review progress of the wizard as it inspects the domain. Progress messages are displayed in the window, as shown in the following example.



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. The error message appears as follows.



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.

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

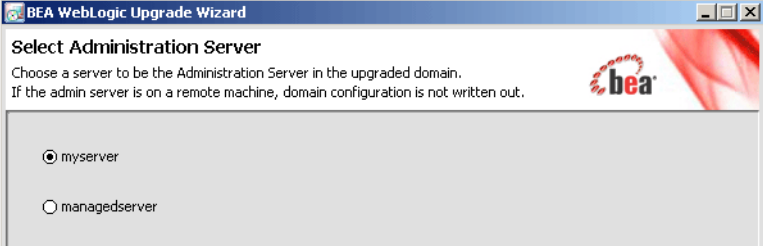
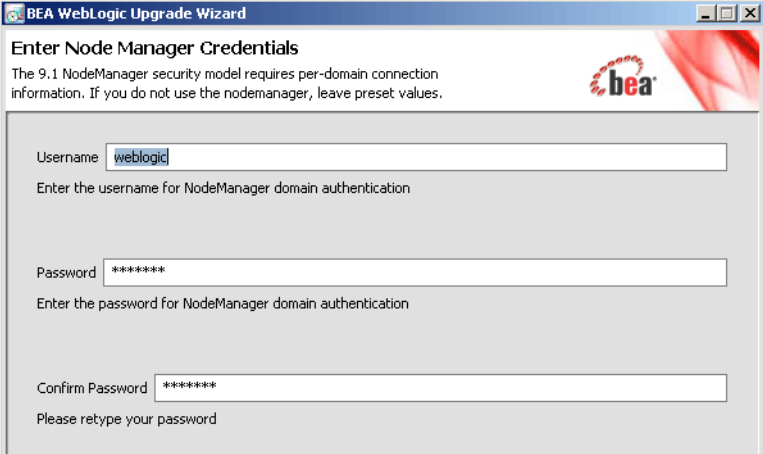
In this step ...	You ...
Select Administration Server	<p>Select a server to function as the Administration Server in the new domain, as shown in the following example.</p>  <p>Note: If there is only one server defined in the domain, this window is skipped. Click Next to proceed to the next window.</p>
Enter Node Manager Credentials	<p>Enter the username and password (and password confirmation) for Node Manager authorization, as shown in the following example.</p>  <p>For WebLogic Server 9.1, Node Manager requires user and password credentials to be specified for each domain. By default, the username and password are set to <code>weblogic</code>. If you do not use Node Manager, leave the default values unchanged. Click Next to proceed to the next window.</p>

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

In this step ...	You ...
<p>Select Upgrade Options</p>	<p>Select one or more of the following upgrade options:</p> <ul style="list-style-type: none"> <p>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.</p> <p>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 2: Back Up the Application Environment” on page 2-6.</p> <p>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.</p> <p>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” on page A-23.</p>
<p>The screenshot shows a window titled "BEA WebLogic Upgrade Wizard" with a sub-header "Select Upgrade Options". Below the sub-header, it says "These deal with optional actions the wizard may perform. Make your selections below." There are three options listed with checkboxes: <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Back up current domain (recommended). Below it, text reads: "The domain directory will be upgraded in place. You will be prompted for a location for the backed up files." <input checked="" type="checkbox"/> Add log files to backup zip. Below it, text reads: "The log files in the domain directory may be extremely large. If the backup option is selected, you can disable adding the log files to the zip, by unchecking this." <input type="checkbox"/> Do not set backwards compatibility flags. Below it, text reads: "Do not set flags which cause previous default behaviour to be used." </p>	
<p>Click Next to proceed to the next window.</p>	

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

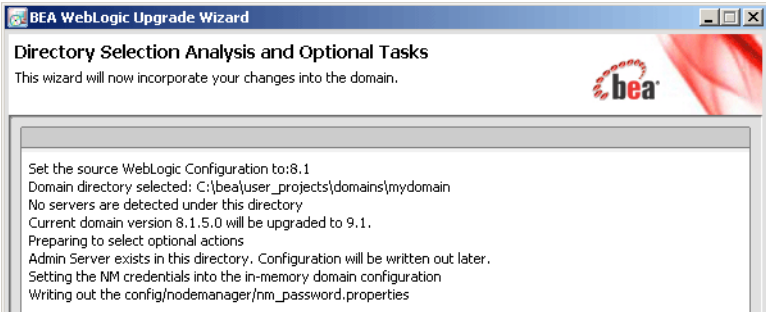
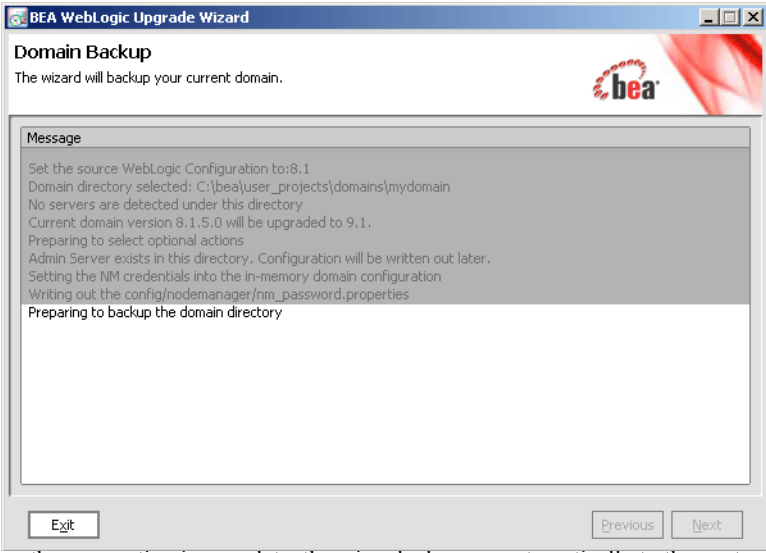
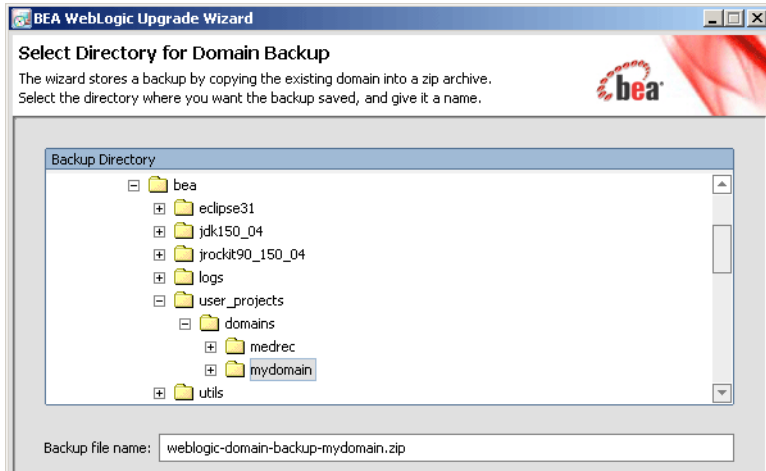
In this step ...	You ...
<p>Directory Selection Analysis and Optional Tasks</p>	<p>Review progress as the wizard processes the domain information and options provided. Progress messages are displayed in the window, as shown in the following example.</p>  <p>Once processing is complete, the wizard advances automatically to the next window.</p>
<p>Domain Backup</p>	<p>Review progress of the wizard as it prepares to back up the domain. Progress messages are displayed in the window, as shown in the following example.</p> <p>Note: Messages that are not relevant to this step are shaded in the following illustration.</p>  <p>Once the preparation is complete, the wizard advances automatically to the next window.</p>

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

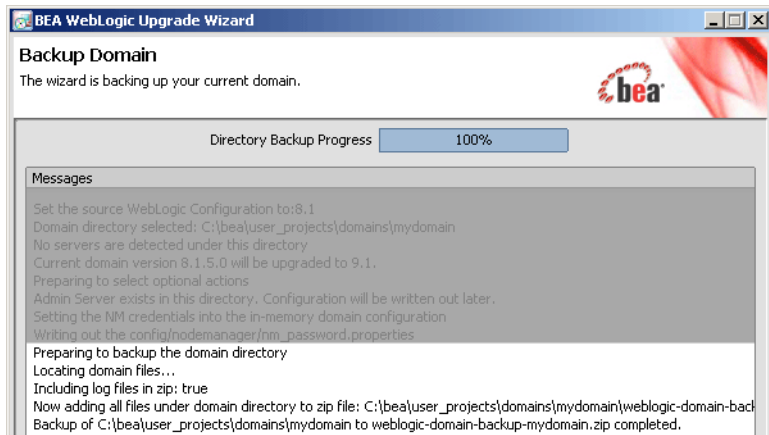
In this step ...	You ...
<p>Select Directory for Domain Backup</p>	<p>Note: If you did not select the Back up current domain upgrade option, as described in “Select Upgrade Options” on page 5-12, skip to “Restructure Domain Directory” on page 5-16.</p> <p>In this window, set values for the following:</p> <ul style="list-style-type: none"> • 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 <code>weblogic-domain-backup-domain.zip</code>, where <i>domain</i> specifies the name of the domain. <p>The following example shows how these values are set.</p>



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

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

In this step ...	You ...
Backup Domain	<p>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, as shown in the following example.</p> <p>Note: Messages that are not relevant to this step are shaded in the following illustration.</p>



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.

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

In this step ...	You ...
Restructure Domain Directory	Review progress as the wizard restructures the domain directory. Progress messages are displayed in the window, as shown in the following example.

BEA WebLogic Upgrade Wizard

Restructure Domain Directory

The wizard is creating a 9.1 directory layout and moving files to the appropriate locations.

Messages

- Created Directory C:\bea\user_projects\domains\mydomain\config\deployments\lib_modules to support 9.1 Doma...
- Finished creating the necessary domain directories for domain mydomain
- Ended execution of Domain Directory Restructure
- Beginning execution of Domain Directory Restructure LogFile Processor
- Log File C:\bea\user_projects\domains\mydomain\mydomain.log not present in default pre-9.1 log location
- Ended execution of Domain Directory Restructure LogFile Processor
- Beginning execution of Domain Directory Restructure SecurityFiles Processor
- Domain Directory Restructure Security File C:\bea\user_projects\domains\mydomain\SerializedSystemIni.dat mov...
- Domain Directory Restructure Security File C:\bea\user_projects\domains\mydomain\DefaultAuthenticatorInit.Idift
- Domain Directory Restructure Security File C:\bea\user_projects\domains\mydomain\DefaultAuthorizerInit.Idift mc
- Domain Directory Restructure Security File C:\bea\user_projects\domains\mydomain\DefaultRoleMapperInit.Idift n...
- Ended execution of Domain Directory Restructure SecurityFiles Processor
- Beginning execution of Domain Directory Restructure DeploymentFiles Processor
- Ended execution of Domain Directory Restructure DeploymentFiles Processor
- Beginning execution of Domain Directory Restructure DirectoryRestructureCleanup Processor
- Ended execution of Domain Directory Restructure DirectoryRestructureCleanup Processor

Once the restructuring process is complete, the wizard advances automatically to the next window.

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

In this step ...	You ...
<p>Upgrade Configuration Settings</p>	<p>Review progress as the wizard upgrades the configuration settings. Progress messages are displayed in the window, as shown in the following example.</p> <p>Note: Messages that are not relevant to this step are shaded in the following illustration.</p> <div data-bbox="440 539 1213 991" style="border: 1px solid gray; padding: 5px;"> </div> <p>The configuration information is not persisted until a later step.</p> <p>Once the configuration upgrade is complete, the wizard advances automatically to the next window.</p>
<p>Upgrade Persisted Messages and Transaction Log Formats</p>	<p>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, as shown in the following example.</p> <div data-bbox="440 1246 1213 1494" style="border: 1px solid gray; padding: 5px;"> </div> <p>Once the persisted message and transaction log upgrade process is complete, the wizard advances to the next window automatically.</p>

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

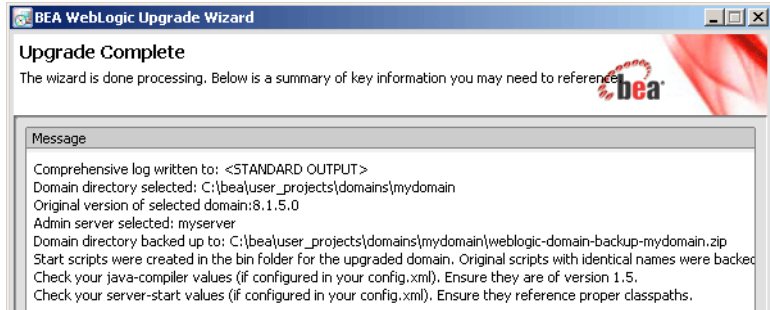
In this step ...	You ...
Finalize Domain Upgrade	<p>Review progress of the wizard as it saves the upgraded configuration and deletes any temporary files that were created during the upgrade process. Progress messages are displayed in the window.</p> <p>Note: When upgrading remote Managed Servers, the wizard does not persist the configuration information.</p> <p>For example:</p> <p>Note: Messages that are not relevant to this step are shaded in the following illustration.</p>



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, as shown in the following example.



Click **Done** to close the WebLogic Upgrade Wizard.

Upgrading a Domain in 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 5: 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 D-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]
```

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 D-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.

Compatibility with Previous Releases

This section describes important compatibility information that you should consider before upgrading to WebLogic Server 9.1.

Note: See also *Compatibility Statement for BEA WebLogic Server 9.1* at <http://e-docs.bea.com/wls/docs91/compatibility/compatibility.html>.

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](#)
- [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](#)
- [Deprecated Deployment Features](#)
- [Deployment Descriptor Validation and Conversion](#)
- [Deprecated Startup and Shutdown Classes](#)
- [Administration Console Extension Architecture](#)
- [Resource Adapters](#)
- [WLEC](#)
- [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. As a result of these incompatibilities in the reference implementation, JMX clients created for WebLogic Server 9.0 or later cannot be used with WebLogic Server 8.1 or earlier. However, despite these incompatibilities, JMX clients created for WebLogic Server 8.1 can be used with 9.1 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 9.1 instance without being upgraded.
- A JMX client in which WebLogic Server 8.1 classes are used can interact with 9.1 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 9.1 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 A-11](#).

BEA recommends that you update your JMX clients to be compliant with WebLogic Server 9.1. 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/docs91/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 and WebLogic Express* at <http://e-docs.bea.com/wls/docs91/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/docs91/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/docs91/secmanage/mbeans.html>.

For more information, see “Managing Configuration Changes” in *Understanding Domain Configuration* at http://e-docs.bea.com/wls/docs91/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-doc.bea.com/wls/docs91/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 9.1, 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/docs90/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 A-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/docs91/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/docs91/jms_admin/intro.html#WhatsNewJMS.
- “Understanding JMS Resource Configuration” in *Configuring and Managing WebLogic JMS* at http://e-docs.bea.com/wls/docs91/jms_admin/overview.html.
- “Deploying JDBC, JMS, and WLDF Application Modules” in *Deploying Applications to WebLogic Server* at http://e-docs.bea.com/wls/docs91/deployment/deploy.html#deploy_resources.

The WebLogic Upgrade Wizard automatically converts pre-9.0 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/docs91/jms_admin/overview.html#jms_interop_modules.

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-9.0 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/docs91/wlsmbeanref/mbeans/JMSServerMBean.html#AllowsPersistentDowngrade>.

- A `TemporaryTemplate` 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 `TemporaryTemplate` attribute.

You can control whether the JMS Server can host a temporary destination by setting the `HostsTemporaryDestinations` 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/docs91/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/docs91/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/docs91/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.

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/docs91/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/docs91/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 2-9](#).

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 E-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/docs91/jta/trxcon.html#interop>.

Security

The following sections identify changes to security.

- [XACML Security Providers](#)
- [New SAML V2 Providers](#)
- [Security MBeans](#)
- [Password Encryption](#)
- [Security for HTTP Requests](#)
- [Secure Access to MBeanHome](#)
- [Message-Level Security in Web Services](#)

XACML Security Providers

As of 9.1, 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 new 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 9.1 include the new 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 9.1 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 9.1* at <http://e-docs.bea.com/wls/docs91/security.html>.

New SAML V2 Providers

New versions of the SAML Credential Mapping provider and SAML Identity Assertion provider have been added in WebLogic Server 9.1. 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 A-2 Changes to Security MBeans as of WebLogic Server 9.0

Type of Security MBean	Description
All security MBeans	<p>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.</p> <p>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 A-3.</p>

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

Type of Security MBean	Description
RealmMBean, UserLockoutManagerMBean, and all security provider MBeans	<ul style="list-style-type: none"> The <code>wls_getDisplay</code> method is deprecated. In its place, you should use the new <code>getName</code> method. In addition, the following security methods have been removed: <ul style="list-style-type: none"> <code>wls_getAttributeTag</code> <code>wls_getConstructorTag</code> <code>wls_getMBeanTag</code> <code>wls_getNotificationTag</code> <code>wls_getOperationTag</code> The <code>wlconfig</code> and <code>weblogic.Admin</code> tools and pre-9.0 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. <p>For security provider MBeans (only):</p> <ul style="list-style-type: none"> 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 <i>any</i> (that is, non-dynamic or dynamic) changes will take effect. For more information, see “Dynamic Configuration Management” on page A-3.
All <i>custom</i> security provider MBeans	<ul style="list-style-type: none"> By default, all custom security provider MBeans attributes are non-dynamic. For more information, see “Dynamic Configuration Management” on page A-3. You can set an MBean attribute to be dynamic by setting <code>Dynamic="true"</code> for the attribute within the MDF file. For example: <pre data-bbox="512 1173 1185 1364"> <MBeanAttribute Name = "Foo" Type = "java.lang.String" Dynamic = "true" Description = "Specifies that this attribute is a dummy." /> </pre>

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/docs91/admin_ref/utils.html.

Security for HTTP Requests

By default, when an instance of WebLogic Server 9.1 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 create 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/docs91/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 with 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 has been 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/docs90/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 9.1, although the 8.1 Web Services run-time engine has been deprecated as of 9.0.

The WebLogic Server 6.1 or 7.0 Web Services need to be upgraded to 8.1 in order to run in 9.1. 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 9.1, all of your 8.1 Web Services, including any 6.1 and 7.0 Web Services that have been upgraded to 8.1. For information about upgrading your existing 8.1 Web Services, see “Upgrading an 8.1 Web Service to 9.0” in *Programming Web Services for WebLogic Server* at

<http://e-docs.bea.com/wls/docs91/webserv/upgrade.html>.

Note: See also “Message-Level Security in Web Services” on page A-14.

Web Applications, JSPs, and Servlets

The following sections provide important compatibility information for Web applications, JSPs, and Servlets in WebLogic Server 9.0:

- “Deprecated and Obsolete Web Application Features” on page A-15
- “Support for JSP 2.0” on page A-15
- “Servlet Path Mapping” on page A-16

Deprecated and Obsolete Web Application Features

For a list of Web application features that are deprecated or are not supported as of WebLogic Server 9.0, see “Deprecated and Obsolete Web Application Features” in *WebLogic Server and WebLogic Express Release Notes* at

http://e-docs.bea.com/wls/docs90/notes/new.html#deprecated_and_obsolete_web_application_features.

Support for JSP 2.0

JSP 2.0 is supported, as of WebLogic Server 9.0. For specific details, see *Developing Web Applications, Servlets, and JSPs for WebLogic Server* at

<http://e-docs.bea.com/wls/docs91/webapp/index.html>.

Please note the following changes to the JSP behavior:

- 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 `println` 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 5-12](#), 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/docs91/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/docs91/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/docs91/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 9.1. 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._forceXqr12002ForXPathXQuery));
```


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 9.1. BEA recommends that you update your scripts to take advantage of the new features provided as of WebLogic Server 9.0. For more information about new features and changes in the MBean hierarchy, see “What’s New in WebLogic Server 9.0” in the *WebLogic Server and WebLogic Express Release Notes* at

<http://e-docs.bea.com/wls/docs90/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.

Deprecated Deployment Features

For a list of application deployment features that are deprecated or no longer supported as of WebLogic Server 9.0, see “Deprecated and Unsupported Deployment Features” in *WebLogic Server and WebLogic Express Release Notes* at

http://e-docs.bea.com/wls/docs90/notes/new.html#deprecated_and_unsupported_deployment_features.

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/docs91/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 J2EE 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/docs91/programming/app_xml.html.

- So that your applications can take advantage of the features in the current J2EE 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 J2EE and WebLogic Server” in *Developing Applications with WebLogic Server* at <http://e-docs.bea.com/wls/docs91/programming/overview.html#ddconverter>.

Deprecated Startup and Shutdown Classes

As of 9.0, application-scoped startup and shutdown classes are 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 new 9.0 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/docs91/programming/lifecycle.html>.

Administration Console Extension Architecture

The WebLogic Server Administration Console has been completely re-designed in the current release. The Administration Console is now built on the WebLogic Portal Framework, which makes it more open and more readily extensible.

The new architecture necessitates new procedures for extending the Administration Console. WebLogic Administration Console extensions built for prior releases of WebLogic Server 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/docs91/console_ext/index.html.

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/docs91/resadapter/intro.html#features>.

Table A-3 Deprecated or Unsupported Resource Adapter Configuration Settings

This element ...	As of WebLogic Server 9.0 ...
<p>Link-Ref Mechanism</p>	<p>This element has been deprecated and replaced by the new J2EE libraries feature. For more information about J2EE libraries, see “Creating Shared J2EE Libraries and Optional Packages” in <i>Developing Applications with WebLogic Server</i> at http://e-docs.bea.com/wls/docs91/programming/libraries.html.</p> <p>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 <i>Programming WebLogic Resource Adapters</i> at http://e-docs.bea.com/wls/docs91/resadapter/creating.html#weblogic-ra-xml.</p>
<p><shrink-period-minutes></p>	<p>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.</p> <p>The <shrink-frequency-seconds> element overrides the <shrink-period-minutes> element if both are set.</p>
<p><connection-maxidle-time></p>	<p>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.</p> <p>The <inactive-connection-timeout-seconds> element overrides the <connection-maxidle-time> element if both are set.</p>
<p><security-principal-map></p>	<p>This element is no longer supported; the security principal map is configured using the Administration Console.</p> <p>You should remove the <security-principal-map> definition from the <code>weblogic-ra.xml</code> file. Otherwise, deployment of the resource adapter fails.</p>

Table A-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/docs91/wlec_migration/index.html.

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 A-4 Backward Compatibility Flags

Category	Backward Compatibility Flag	For More Information
Security	<ul style="list-style-type: none"> <code>EnforceStrictURLPattern</code>—Specifies whether the server should enforce strict adherence of URL patterns to the Servlet 2.4 specification. During an upgrade, this flag is set to <code>false</code> for backward compatibility. <code>WebAppFilesCaseInsensitive</code>—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 <code>os</code>, 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 	<p><code>SecurityConfiguration</code> MBean at http://e-docs.bea.com/wls/docs91/wlsmbearcf/mbeans/SecurityConfigurationMBean.html</p>

Table A-4 Backward Compatibility Flags (Continued)

Category	Backward Compatibility Flag	For More Information
Web Application	<ul style="list-style-type: none"> <code>AllowAllRoles</code>—Specifies that a wildcard (*) character can be used to set the role name to enable all users/roles in the <i>realm</i> 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 <i>Web application</i> will have access to the resource collection. <code>FilterDispatchedRequestsEnabled</code>—Specifies whether to apply filters to dispatched requests. As of WebLogic Server 9.0, the new <code>Dispatcher</code> element makes this behavior explicit. <code>JSPCompilerBackwardsCompatible</code>—Specifies whether to allow JSPs that do not comply with the JSP 2.0 specification. <code>ReLoginEnabled</code>—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. <code>RtexprvalueJspParamName</code>—Specifies whether to allow run-time expression values for the JSP <code><param name></code> tag. As of WebLogic Server 9.0, the JSP Compiler no longer allows run-time expression values. 	<p>WebAppContainerMBean at http://e-docs.bea.com/wls/docs91/wlsmbeanref/mbeans/WebAppContainerMBean.html</p>

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 9.1 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/utills/MedRecWebAppUtils.java` and `medrec/src/clients/com/bea/medrec/webservices/swing/EditProfileFrame.java`.

The 8.1, 9.0, and 9.1 packages, `weblogic.webservice.tools.wsdlp` and `weblogic.webservice.wsdl`, respectively, are not publicly available.

See also “Web Services” on page A-14.

Deprecated APIs

For a list of **deprecated** APIs, see

<http://e-docs.bea.com/wls/docs90/javadocs/deprecated-list.html>.

Removed APIs

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

- [Removed APIs, Deprecated in WebLogic Server 8.1](#)
- [Removed APIs, Deprecated in WebLogic Server 7.0](#)
- [Removed APIs \(Not Deprecated\)](#)

Removed APIs, Deprecated in WebLogic Server 8.1

The following table provides a list of APIs that have been removed in WebLogic Server 9.0 and were deprecated in WebLogic Server 8.1.

Table A-5 Removed APIs—Deprecated in WebLogic Server 8.1

Package Name	Interface/Class/Exception	Method/Variable
weblogic.jdbc.common	Oracle	*
weblogic.jdbc.oci	*	*
weblogic.jdbc.oxa	*	*
weblogic.management.configuration	JDBCDataSourceMBean NetworkChannelMBean SecurityMBean	
weblogic.management.console	extensibility	ExtensibleTag
weblogic.net.http	HttpsURLConnection	weblogic.security.SSL.HostnameVerifier getDefaultHostnameVerifier()
weblogic.security	Key	

Table A-5 Removed APIs—Deprecated in WebLogic Server 8.1 (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.security.SL	HostnameVerifier	boolean verify(java.lang.String, java.lang.String) boolean verify(java.net.InetAddress, weblogic.security.X509)
weblogic.security.service	WebResource	

Removed APIs, Deprecated in WebLogic Server 7.0

The following table provides a list of APIs that have been removed in WebLogic Server 9.0 and were deprecated in WebLogic Server 7.0.

Table A-6 Removed APIs—Deprecated in WebLogic Server 7.0

Package Name	Interface/Class/Exception	Method/Variable
weblogic.common	T3User	T3User (weblogic.security.SSL.SSLCertificate)
	WLSerializable	
weblogic.io.common	T3File	
	T3FileInputStream	
	T3FileOutputStream	
	T3FileSystem	

Table A-6 Removed APIs—Deprecated in WebLogic Server 7.0 (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.management .configuration	Acl	
	BasicRealmMBean	
	CachingRealmMBean	
	CustomRealmMBean	
	FileRealmMBean	
	Group	
	LDAPRealmMBean	
	NTRealmMBean	
	PasswordPolicyMBean	
	Principal	
	RDBMSRealmMBean	
	RealmException	
	RealmIterator	
	RealmMBean	
	RealmManager	
UnixRealmMBean		
User		
XMLRegistryEntryMBean		
weblogic.management .runtime	DeploymentRuntimeMBean	

Table A-6 Removed APIs—Deprecated in WebLogic Server 7.0 (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.security	acl	CertAuthentication weblogic.security.acl.User authenticate(java.lang.String, weblogic.security.Certificate[], boolean)
	acl	CertAuthenticator weblogic.security.acl.User authenticate(java.lang.String, weblogic.security.Certificate[], boolean)
	acl	Security void receiveSecurityMessage(weblogi c.rjvm.JVMID, weblogic.security
	acl	SecurityMessage void execute(weblogic.rjvm.JVMID)
	audit	Audit void certificateInvalid(java.lang.S tring, java.lang.Object, weblogic.security.X509)
	audit	Audit void rootCAInvalid(java.lang.String , java.lang.Object, weblogic.security.X509)
	audit	AuditProvider void certificateInvalid(java.lang.S tring, java.lang.Object, weblogic.security.X509)
	audit	AuditProvider void rootCAInvalid(java.lang.String , java.lang.Object, weblogic.security.X509)

Table A-6 Removed APIs—Deprecated in WebLogic Server 7.0 (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.security	AuthenticationException	
	Certificate	
	Cipher	
	Cipher3EDE	
	CipherCBC	
	CipherException	
	Coder	
	CoderException	
	CoderInputStream	
	CoderOutputStream	
	DES	
	DiffieHellman	
	DigestInputStream	
	DigestOutputStream	
	DigestSignature	
	Entity	
	HexOutputStream	
	JDK11Certificate	
	KeyManagementException	
	MD5RandomBitsSource	
NullCipher		
NullOutputStream		
PKCS5		

Table A-6 Removed APIs—Deprecated in WebLogic Server 7.0 (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.security	Padding	
	RC4	
	RSA	
	RSAKey	
	RSAMDSignature	
	RSAPrivateKey	
	RSAPrivateKeyPKCS8	
	RSAPublicKey	
	RSApkcs1	
	RandomBitsSource	
	service	InvocableResource
		java.lang.String toString()
		MBeanResource
		RealmAdapterAclResource
	Signature	
	SimpleRandomBitsSource	
	SpinnerRandomBitsSource	
	StreamCipher	
	Streamable	
	SymmetricCipher	
	TeeOutputStream	
	X509Name	
	X509	

Table A-6 Removed APIs—Deprecated in WebLogic Server 7.0 (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.security.SSL	BadMACException	
	Biguint	
	CipherSpec	
	RecordInputStream	
	SSLCertificate	
	SSLParams	
	SSLSocket	
	SessionParams	
	TrustManager boolean certificateCallback(weblogic.security.X509[], int)	
weblogic.xml.sax	XMLInputSource	

Removed APIs (Not Deprecated)

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

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

Table A-7 Removed APIs (Not Deprecated)

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

Table A-7 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 A-7 Removed APIs (Not Deprecated) (Continued)

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

Table A-7 Removed APIs (Not Deprecated) (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.management.security.authentication	UserLockoutManagerMBean	*setRealm*
weblogic.management.timer	Timer	void addNotification(weblogic.time.common.internal.TimerNotification) java.lang.String getDispatchPolicy() void removeNotification(weblogic.time.common.internal.TimerNotification) void removeNotification(weblogic.time.common.internal.TimerNotification, boolean) void run() void setDispatchPolicy(java.lang.String) void start(java.lang.ThreadGroup, java.lang.String, int)
weblogic.net.http	HttpClient HttpsClient HttpUnauthorizedException HttpURLConnection HttpsURLConnection KeepAliveCache KeepAliveStream MessageHeader	* *

Table A-7 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 A-7 Removed APIs (Not Deprecated) (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.security.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.webservice.async	KernelFeederImpl	ExecuteTask void execute(weblogic.kernel.ExecuteThread)
weblogic.webservice.client	BaseWLSecurityAdapter	void main(java.lang.String) void _setStrictChecking(boolean)
	JSSEAdapter	javax.net.ssl.SSLSocketFactory getSocketFactory() void setSocketFactory(javax.net.ssl.SSLSocketFactory)
	SSLAdapterFactory	void main(java.lang.String)
weblogic.xml	xpath	*XMLNodeXPath.*
weblogic.xml.security.specs	SecuritySpec	java.lang.String getName()

Table A-7 Removed APIs (Not Deprecated) (Continued)

Package Name	Interface/Class/Exception	Method/Variable
weblogic.xml.security.wsse	SecurityElementFactory	weblogic.xml.security.wsse.Token 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.

Compatibility with Previous Releases

WebLogic Domain Directory Structure Enhancements

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

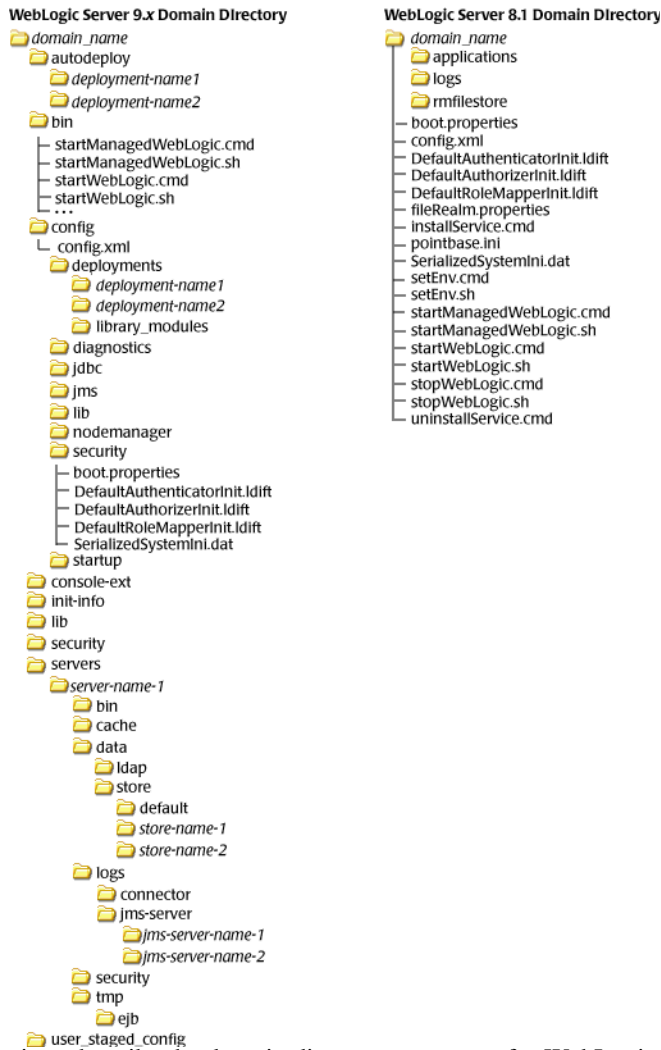
- To improve configuration management and promote XML file validation, WebLogic Server supports the specification of domain configuration data in multiple files, including `config.xml`, in the new `domain_name/config` directory. (Here `domain_name` specifies the domain directory.) In previous releases, 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 new utilities for managing changes to server configuration. These new 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/docs91/domain_config/changes.html.

For more details about the WebLogic Server 9.x domain directory structure, see “Domain Configuration Files” in *Understanding Domain Configuration* at http://e-docs.bea.com/wls/docs91/domain_config/config_files.html.

Figure B-1 compares the domain directory structures for WebLogic Server 9.x and WebLogic Server 8.1.

Figure B-1 WebLogic Server 9.x and WebLogic Server 8.1 Directory Structures



The following sections describe the domain directory structures for WebLogic Server 8.1, 7.0, and 6.1.

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/configwiz/tempref.html>.

WebLogic Server 7.0 Domain Directory Structure

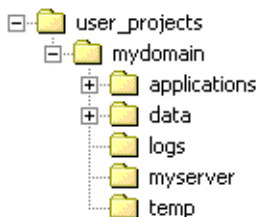
In WebLogic Server 7.0 environments, the root of the domain directory structure is a directory with the same name as the domain, such as `mydomain` or `petstore`. This directory should contain the following:

- `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.

WebLogic Server 6.1 Domain Directory Structure

In WebLogic Server 6.1 environments, the configuration information for all of the WebLogic administrative domains resides in the configuration repository, which is located, in turn, in the `WL_HOME/config` directory. Specifically, `config` contains a configuration subdirectory for each domain. The name of each domain-specific subdirectory must match the name of the domain.

The domain directory structure contains a root 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 subdirectory, typically named `applications`
- `logs` directory for storing domain-level logs

For more information about the WebLogic Server 6.1 domain directory structure, see “Creating a New Domain” in “Overview of WebLogic Server Management” in *WebLogic Server Administration Guide* at

<http://e-docs.bea.com/wls/docs61/adminguide/overview.html>.

WebLogic Domain Directory Structure Enhancements

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 C-1 Command-Line Arguments for the WebLogic Upgrade Wizard

Argument	Description
<code>-help -h -? -usage</code>	Displays help information.
<code>-type <i>type</i></code>	Specifies one of the following types of upgrade: <code>domain</code> , <code>nodemanager</code> , or <code>securityproviders</code> . The default value is <code>domain</code> .
<code>-mode <i>mode</i></code>	Specifies one of the following modes of upgrade: <code>gui</code> or <code>silent</code> . The default value is <code>gui</code> .
<code>-responses <i>xmlfile</i></code>	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 <code>-mode</code> is set to <code>silent</code> . For more information about the format of the responses file, see “Silent Upgrade XML Script Reference” on page D-1 .
<code>-out <i>file</i></code>	Specifies the file in which standard output and errors are stored. By default, log output is sent to <code>stdout</code> .

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 D-1](#). If the wizard does not locate an XML script, it uses the default system values, as defined in [Table D-2](#).

Table D-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/docs91/upgrade/scripts/weblogic-upgrade-securityproviders-responses.xml
Node Manager	weblogic-upgrade-nodemanager-responses.xml Sample: http://e-docs.bea.com/common/docs91/upgrade/scripts/weblogic-upgrade-nodemanager-responses.xml
Domain	weblogic-upgrade-domain-responses.xml Sample: http://e-docs.bea.com/common/docs91/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 D-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 D-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 D-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 <i>WL_HOME</i> \server\lib\mbeantypes, where <i>WL_HOME</i> specifies the root directory of the pre-9.0 installation.	Path of directory from which the Upgrade Wizard is run, such as: <i>c:\bea\weblogic81\server\lib\mbeantypes</i>
	OUTPUT_DIRECTORY	Path of the directory in which you want to save the new security provider JAR files.	<i>WL_HOME</i> \server\lib\mbeantypes, where <i>WL_HOME</i> specifies the root directory of the WebLogic Server 9.1 installation.

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

If you are upgrading...	For this keyword...	Set the value to the...	This keyword defaults to the...
Node Manager	NODE_MANAGER_HOME	<p>Path of the directory of Node Manager to be upgraded by navigating the local directory hierarchy.</p> <p>By default, the Node Manager home directory is located in <i>WL_HOME</i>\common\nodemanager, where <i>WL_HOME</i> specifies the root directory of the pre-9.1 installation.</p>	<p>Directory from which the Upgrade Wizard is run. For example: c:\bea\weblogic81\common\nodemanager</p>
Domains	WEBLOGIC_VERSION	Version of WebLogic Server.	<p>Version of WebLogic Server identified in the configuration file (<i>config.xml</i>). For example, 8.1.4.0.</p> <p>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-9.1 domain.</p>
	DOMAIN_DIR	<p>Path of the directory that contains the WebLogic domain to be upgraded.</p>	<p>Directory from which the Upgrade Wizard is run, such as: c:\bea\user_projects\domains\mydomain</p>

Table D-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)	ADMIN_SERVER_NAME	Name of the Administration Server.	Name of the Administration Server specified in the configuration file (<code>config.xml</code>), such as: <code>myserver</code> This keyword is required to be set for WebLogic Server 6.1 release only, as the Administration Server name was not persisted to the configuration file.
	NODE_MANAGER_USERNAME	Username for Node Manager.	<code>weblogic</code>
	NODE_MANAGER_PASSWORD	Password for Node Manager.	<code>weblogic</code>
	NODE_MANAGER_PASSWORD	Confirmation password for Node Manager.	<code>weblogic</code>

Table D-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	<p>One or more of the following options:</p> <p>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.</p> <p>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.</p> <p>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.</p>	DOMAIN_DIRECTORY_BACKUP_SELECTED_VALUE, DOMAIN_DIRECTORY_BACKUP_LOG_FILES_INCLUDED_SELECTED_VALUE
	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: <code>c:\bea\user_projects\domains\mydomain</code>
	BACKUP_FILE_NAME	Name of the backup zip file.	<code>weblogic-domain-backup-mydomain.zip</code>

Silent Upgrade XML Script Reference

Upgrading a Domain at Administration Server Startup (Implicit Mode)

The implicit mode of upgrade enables you to upgrade a 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.

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 5: Set Up the Environment” on page 2-8](#).
3. Update the startup scripts in the domain to reference the WebLogic Server 9.1 installation.

For example, make sure `WL_HOME` is set to the WebLogic Server 9.1 installation, as in the following example: `c:\bea\weblogic91`.

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.

Upgrading a Domain at Administration Server Startup (Implicit Mode)

The WebLogic Upgrade Wizard checks whether the domain directory was generated using a valid version of WebLogic Server (6.1, 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.