Oracle® Fusion Middleware

Upgrading Oracle WebLogic Server

12c (12.2.1.3.0)
E80398-04
April 2018

Oracle Fusion Middleware Upgrading Oracle WebLogic Server, 12c (12.2.1.3.0)

E80398-04

Contents

Preface

This preface describes the document accessibility features and conventions used in this guide—*Upgrading Oracle WebLogic Server.*

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

You can upgrade WebLogic servers and domains from an earlier version of WebLogic Server to WebLogic Server 12*c*. You can also update an existing application to run on WebLogic Server 12.2.1.3.0.

While upgrading to version 12.2.1.3.0, you might want to change your application or you might have to change the application. However, this document focuses only on issues that you should consider when moving an application to WebLogic Server 12.2.1.3.0 without making any application changes.

The instructions in this document are for the following upgrade scenarios:

- Upgrading from any WebLogic Server 10.3.x release to WebLogic Server 12.2.1.3.0
- Upgrading from WebLogic Server 12.1.x to WebLogic Server 12.2.1.3.0

> **Note:**
>
> If you are upgrading from a release prior to WebLogic Server 10.3.1, see Upgrading From a WebLogic Version Prior to WebLogic Server 10.3.6.
>
> If you are upgrading from version 12.2.1.1.0 and later to version 12.2.1.3.0, you do not need to run the Upgrade Assistant (UA) for schemas or configuration upgrades. You must only run the Reconfiguration Wizard. However, you must install the binaries in a new Oracle home and run the reconfiguration offline.

This document also describes how to update (reconfigure) an existing WebLogic Server 10.3.x or 12.1.x domain to be compatible with WebLogic Server 12.2.1.3.0, as well as how to upgrade Web Services.

WebLogic Server generally supports high levels of upgrade capability across WebLogic Server versions. This document is intended to provide WebLogic Server upgrade support and identify issues that may surface during an upgrade so that they can be easily resolved.

> **Note:**
>
> For information about upgrading your Java EE environment and your deployed applications from Oracle Application Server 10*g* and Oracle Containers for Java EE (OC4J) to WebLogic Server 12*c* release (12.2.1.3.0), see Fusion Middleware Upgrade Guide for Java EE.
>
> This document describes the upgrade process for Oracle product installations that include only WebLogic Server. If your installation includes other Oracle Fusion Middleware products, prior to beginning the upgrade, see *Planning an Upgrade of Oracle Fusion Middleware* and the upgrade guides for each Fusion Middleware product in your installation.

WebLogic Server 12.2.1.3.0 includes the Fusion Middleware Reconfiguration Wizard to assist you with upgrading WebLogic Server and your application environments.

Most WebLogic Server applications can be run without modifications in the new WebLogic Server 12.2.1.3.0 application environment.

This chapter includes the following sections:

1.1 Version Compatibility

Before you upgrade WebLogic Server, review the WebLogic Server and domain compatibility requirements for WebLogic Server 12.2.1.3.0.

See *Compatibility Within a Domain* in *Understanding Oracle WebLogic Server*.

Within a WebLogic domain, the Administration Server, all Managed Server instances, and the WebLogic domain must be at the same WebLogic Server Major and Minor Version. This means that in WebLogic Server 12.2.1.x, the Administration Server, Managed Servers, and the WebLogic domain must all be at version 12.2.1.x. Versions of WebLogic Server prior to 12.1.2 have slightly different compatibility allowances regarding specific WebLogic Server versions that are supported in a given domain.

1.2 Important Terminology

The documentation for upgrading WebLogic Server uses various terms when describing its features and functionality. It is important that you have a good understanding of these terms.

- **Upgrade**—In this document, the term upgrade refers to the process of upgrading WebLogic Server and moving an existing application, unchanged, to a new (upgraded) WebLogic Server version.
- **Reconfiguration**—The process of upgrading a domain that was created with a previous WebLogic Server version so that it is compatible with the WebLogic Server version to which you have upgraded. This can be done using either the Reconfiguration Wizard or WLST.
- **Application Environment**—An application environment includes applications and the WebLogic domains in which they are deployed. It also includes any application data associated with the domain, and may include resources such as database servers, firewalls, load balancers, and LDAP servers.
- **Migrate**—To move an application or domain configuration from a third-party product to an Oracle product.
- **Interoperability**—(1) The ability of an application deployed in one WebLogic Server version to communicate with another application that is deployed in a different WebLogic Server version. (2) The ability of Oracle product components to communicate with third-party software using standard protocols.
- **Compatibility**—The capability of an application built using one WebLogic Server release to run in another WebLogic Server release, regardless of whether the application was rebuilt.

1.3 Upgrading From a WebLogic Version Prior to WebLogic Server 10.3.6

To upgrade to WebLogic Server 12.2.1.3.0 from a version prior to 10.3.1, you must first upgrade to WebLogic Server 10.3.6, and then upgrade to 12.2.1.3.0.

If you are currently using a WebLogic version prior to WebLogic Server 10.3.6, upgrading to version 12.2.1.3.0 is a two-stage process:

- Upgrade your installation to WebLogic Server 10.3.6.

  To do so, follow the instructions in *Upgrade Guide for WebLogic Server 10.3.6* at `http://docs.oracle.com/middleware/11119/wls/WLUPG/intro.htm`.

  Be sure to run the **WebLogic Server 10.3.6 Domain Upgrade Wizard** to upgrade your domains.

> **Note:**
>
> To download a WebLogic Server 10.3.6 upgrade installer, enter the appropriate patch number on My Oracle Support:
>
> – Patch 13529623—10.3.6 Generic Upgrade Installer (does not include a bundled JDK)
> – Patch 13529653—10.3.6 Linux 32-bit Upgrade Installer
> – Patch 13529639—10.3.6 Windows 32-bit Upgrade Installer
> – Patch 13529649—10.3.6 Solaris 32-bit Upgrade Installer

- Upgrade WebLogic Server 10.3.6 to WebLogic Server 12.2.1.3.0 per the instructions in this guide.

> **Note:**
>
> As of WebLogic Server 12.1.2, Oracle no longer provides upgrade installers. You must install WebLogic Server 12.2.1.3.0 to a new directory location. You cannot install it over an existing installation.

1.4 Overview of the Upgrade Process

You can upgrade all WebLogic Server applications and domains simultaneously, upgrade them in a well-defined sequence, or upgrade some applications and domains while leaving other applications and domains on older WebLogic Server versions.

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

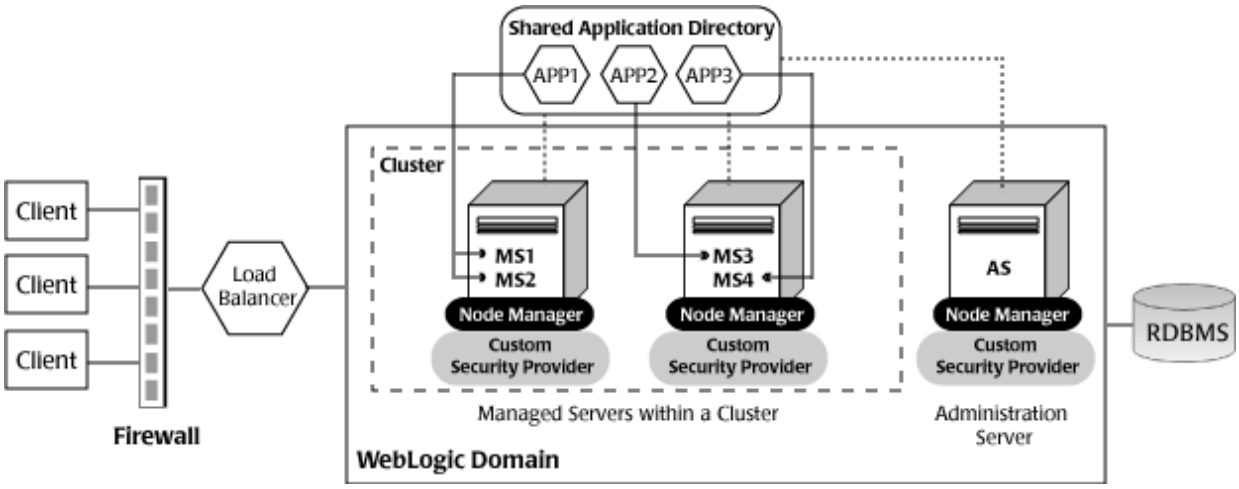


Table 1-1 lists the components of the WebLogic application environment shown in Figure 1-1 and the upgrade requirements for each.

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 Oracle WebLogic domains in *Understanding Domain Configuration for Oracle WebLogic Server*.	Upgrade the domain directory on each computer in the domain.
Applications	Any Java EE applications, including Web applications, EJBs, and so on. Typically, applications are deployed to one or more Managed Servers in a domain. Depending on the deployment strategy, applications may reside locally on a computer or be accessible using 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 12.2.1.3.0 application environment. See Interoperability and Compatibility with Previous Releases.
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 12.2.1.3.0. See the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.

Upgrading business applications that are deployed to WebLogic Server may involve upgrading multiple WebLogic Server applications, and in some cases domains, in a coordinated fashion to:

- maintain consistency in the WebLogic Server versions being used
- use the same supported configurations environment across the entire installation
- meet specific interoperability requirements.

1.5 Before You Begin

Before you upgrade WebLogic Server, verify that your machine is set up to meet the requirements to upgrade and run WebLogic Server. You must also consider the scope of the environments that you are upgrading and which applications are upgraded in which sequence.

As covering all the permutations of an upgrade is beyond the scope of this document, consider the following items prior to planning your upgrade. These items focus on upgrades that involve a single application running in a single domain.

- Oracle recommends that you upgrade an application in development environments and use a standard QA, testing, and staging process to move upgraded applications to a production environment.
- You typically upgrade an application either by upgrading an existing domain or by creating a new domain, from which you can run the application on the new

WebLogic Server version. Sometimes, you prefer to create new domains using the **Fusion Middleware Configuration Wizard** or other configuration tools (such as WLST) to test the applications that you are upgrading.

> **Note:**
>
> If the domain was created using a WebLogic Server version prior to version 10.3.1, you must first upgrade to WebLogic Server 10.3.6. See Upgrading From a WebLogic Version Prior to WebLogic Server 10.3.6. After upgrading to WebLogic Server 10.3.6, run the **WebLogic Server 10.3.6 Domain Upgrade Wizard** to upgrade the domain. You can then use the **Reconfiguration Wizard** to upgrade the domain to WebLogic Server 12.2.1.3.0.

- When planning a WebLogic Server version upgrade, you should review the Fusion Middleware Supported Systems Configurations page on Oracle Technology Network (OTN) to ensure that your upgraded environment is supported by Oracle, in particular:
  - current and planned JVM and JDK versions
  - operating system versions
  - database versions
  - Web services versions
  - versions of other products that interoperate with or run on WebLogic Server, to ensure that the upgraded environment is supported by Oracle or other vendors' products that you are using with WebLogic Server.
- On an ongoing basis, Oracle documents APIs and features that have been deprecated (that is, planned for removal in a future release). This is intended to inform you that you should avoid using these APIs and features to ensure upgradability. Oracle also documents the APIs and features that have actually been removed in the current release so that if you are upgrading from prior versions, you can determine if your applications will be affected by an upgrade.

  APIs and feature removals are cumulative. For example, if you are upgrading from WebLogic Server 10.0 to WebLogic Server 12.2.1.3.0, your applications may be affected by APIs or features that were removed in WebLogic Server 10.3, as well as by APIs or features that were removed in WebLogic Server 12.2.1.3.0. When upgrading, you should review all documentation of deprecated and removed features for all applicable WebLogic Server versions.
- You should consider the impact (if any) that the upgrade process may have on any automation (such as WLST scripts) that you are using to configure, deploy, start/stop, or monitor your WebLogic Server applications. You may need to upgrade such automation along with the applications and domains you are upgrading.
- You should consider the potential impact that may result from the use of third-party libraries in your applications, as they may conflict with different versions of those same libraries that are embedded in WebLogic Server. In particular, new versions of WebLogic Server may change the version of open source libraries that are embedded in WebLogic Server. Applications that may run successfully on earlier WebLogic Server versions may encounter new class conflicts after upgrade.

If you are upgrading an application that contains embedded third-party libraries, you should consider using the Classloader Analysis Tool, and filtering classloaders when upgrading WebLogic Server applications to WebLogic Server 12.2.1.3.0. This tool enables you to identify, diagnose and resolve such conflicts, and may simplify the upgrade process.

- If you are running applications on prior versions of WebLogic Server, and are using WebLogic Server patches or bug fixes, you should investigate whether or not those patches or bug fixes have been incorporated into the version of WebLogic Server to which you are upgrading.

1.6 Interoperability and Compatibility with Previous Releases

Most existing WebLogic Server version 10.3.1 or later applications can be run without modification in the new WebLogic Server 12.2.1.3.0 application environment.

To determine whether any feature changes affect the applications in your environment, review the compatibility information described in WebLogic Server 12.2.1.3.0 Compatibility with Previous Releases. If your application uses APIs that have been deprecated or removed, then you may encounter warnings or exceptions at run time.

1.7 Patching an Existing WebLogic Server Installation

You can patch an existing WebLogic Server installation using either Zero Downtime Patching or manually rolling update of your servers.

Zero Downtime Patching (ZDT Patching) is a highly automated way to roll out updates to the servers in your domain with no loss of services to your customers. Use this method only if your domain contains three or more nodes and all the servers you want to patch are assigned to clusters. See About Zero Downtime Patching.

Manually performing a rolling update of your servers results in no loss of service to your customers. Use this method to patch individual servers that are not part of a cluster or if the domain contains fewer than three nodes. See About Zero Downtime Patching.

Obtaining a List of Applied Patches explains how you can see the list of patches that have already been applied to a WebLogic Server instance.

1.7.1 About Zero Downtime Patching

As of WebLogic Server 12.2.1, you can use Zero Downtime Patching (ZDT Patching) to automate the process of applying patches, bundle patches or patch set updates to a WebLogic Server installation.

With ZDT Patching, you can use the OPatchAuto tool, WLST, or the WebLogic Server Administration Console to orchestrate the rollout of updates across some or all of the servers in your domain. In brief, this involves:

- Creating and patching a second Oracle Home. You can do this manually or you can use OPatchAuto to automate the process.
- Distributing the patched Oracle Home to all of your nodes. Again, you can do this manually or you can use OPatchAuto to do it for you.

- Using OPatchAuto, WLST, or the WebLogic Server Administration Console to configure a patching workflow to update the desired servers in your domain.

With ZDT Patching, you can also use a patching workflow to revert patches that you have previously applied to a WebLogic Server installation using ZDT Patching.

For more details about ZDT Patching, see *Administering Zero Downtime Patching Workflows*.

1.7.2 About Rolling Updates

A rolling update preserves the state of active client sessions. During the rolling update process, client sessions are failed over from inactive servers to active servers to provide a seamless experience for your application users.

Rolling update of WebLogic Server refers to the process of installing a patch, bundle patch, or patch set update to an existing WebLogic Server installation without shutting down the entire cluster or domain.

During the rolling update of a cluster, each server in the cluster is individually patched and restarted while other servers in the cluster continue to operate. You can also perform a rolling update on Managed Servers that are not part of a cluster.

> **Note:**
>
> If your installation includes Oracle Enterprise Manager (EM), see Patching Software Deployments in *Oracle Enterprise Manager Lifecycle Management Administrator's Guide*.

Note the following limitations for rolling updates:

- You cannot use a rolling update to upgrade to a new minor version of WebLogic Server, for example from version 12.1.2 to 12.1.3. You can install only individual patches, patch bundles, or patch set updates (for example, 12.1.2.0.0 to 12.1.2.0.1, 12.1.2.0.1 to 12.1.2.0.2, or 12.1.2.0.0 to 12.1.2.0.5). To upgrade to a new minor version, you must install the new version in a new Oracle Home directory. See *Installing and Configuring Oracle WebLogic Server and Coherence*.
- You must update the machine on which the Administration Server is running first, as that machine must be running the same or a newer version of the software than the machines in the domain that are running only Managed Servers.
- For machines on which WebLogic Server is installed, if multiple servers are running on the machine, you must shut down all servers on the machine, including the Administration Server if it is running on that machine, before you can perform the rolling update.
- You should not make configuration changes during the rolling update process until all the servers in the domain have been updated. This is especially true for new configuration options. Servers silently ignore settings that they do not understand, and the local configuration file may not be updated properly. In addition, using new configuration options may prohibit the deinstallation of a patch, patch set, or patch set update in a rolling fashion.

1.7.2.1 Performing a Rolling Update

To perform a rolling update using patches, bundle patches, or patch set updates for Oracle WebLogic Server, use the Oracle OPatch tool. The general process is as follows, starting with the Administration Server. See Patching with OPatch.

1. Back up your applications, database schema, other application data, and domains.
2. Download the WebLogic Server patch, bundle patch, or patch set update to a server in the cluster.
3. Shut down the server or servers on the machine to be upgraded:
   a. Complete any pending processes.
   b. Gracefully shut down the server or servers.
4. Apply the patch or patch set update.
5. Restart the server or servers.
6. Repeat the above steps for each server machine you need to patch.

> **Note:**
>
> As a best practice, in order to preserve the state of active client sessions, you should wait a reasonable amount of time before shutting down the servers on the next machine in your upgrade sequence. The amount of time that you should wait can be as little as 5-10 minutes, depending on how long it takes your applications to invalidate idle client sessions.

1.7.2.2 Rolling Back a Patch, Bundle Patch, or Patch Set Update

Use the **Oracle OPatch** tool to roll back an applied patch, bundle patch, or patch set update.

- To roll back a single patch, see Rolling Back a Patch You Have Applied.
- To roll back multiple patches, see Rolling Back Multiple Patches You Have Applied.

1.7.3 Obtaining a List of Applied Patches

Oracle WebLogic Server provides the ability to display the list of patches that have been applied to a WebLogic Server instance. The patch list can be obtained from either of the following sources:

- weblogic.log.DisplayPatchInfo System Property
- ServerRuntimeMBean.PatchList Attribute

When you use one of the preceding sources, the following details are provided for each applied patch:

- Associated bug number

- Patch number
- Date the patch was applied
- Brief description

The following sections explain the available ways to obtain a server's list of applied patches.

weblogic.log.DisplayPatchInfo System Property

The `weblogic.log.DisplayPatchInfo` system property contains a log of all patches that have been applied to a WebLogic Server instance, and can be accessed by either of the following methods:

- Specifying the `-Dweblogic.log.DisplayPatchInfo=true` JVM option in the command line that starts the server instance. As the server starts, the startup messages in `stdout` include the list of applied patches, and they are also retained in the server log file. Note that to minimize logging overhead during startup, the default value of this option is `false`.
- Running the `weblogic.version` utility. This utility can obtain the patch list regardless of whether the `-Dweblogic.log.DisplayPatchInfo=true` startup option is used, and does not require the WebLogic Server instance to be starting or running.

The following example shows running the `weblogic.version` utility. This example includes specifying the classpath of the `weblogic.jar` file corresponding to the specific server instance whose patch list is to be displayed.

```
bash-4.1$ java -classpath wlserver/server/lib/weblogic.jar weblogic.version

WebLogic Server 12.2.1.1.0 Thu Jun  2 16:21:58 PDT 2016 1784838
24907328;20845986;Mon Mar 13 14:40:42 PDT 2017;WLS PATCH SET UPDATE 12.2.1.1.170117
19795066;19149348;Mon Mar 13 14:33:28 PDT 2017;One-off
18905788;18668039;Mon Mar 13 14:32:57 PDT 2017;One-off
19632480;19278519;Mon Mar 13 14:32:26 PDT 2017;One-off
19002423;18804275;Mon Mar 13 14:31:50 PDT 2017;One-off
19030178;19234068;Mon Mar 13 14:31:22 PDT 2017;One-off
19154304;19278518;Mon Mar 13 14:30:54 PDT 2017;One-off

Use 'weblogic.version -verbose' to get subsystem information

Use 'weblogic.utils.Versions' to get version information for all modules
```

ServerRuntimeMBean.PatchList Attribute

The list of patches that have been applied to a WebLogic Server instance is also available from the `ServerRuntimeMBean.PatchList` attribute. The value of this attribute is independent of the `weblogic.log.DisplayPatchInfo` system property. You can access the `ServerRuntimeMBean.PatchList` attribute using any of the following clients:

- WLST — See Example 1-1
- REST API — See Example 1-2
- WebLogic Server Administration Console — See Example 1-3
- JMX — See Example 1-4

> **Note:**
>
> To access the patch list from the `ServerRuntimeMBean`, you must be an authenticated user whose identity can be mapped to the `Admin` role.

Regardless of the client that you use to obtain patch information, each patch entry has the following format:

```
<BugNumber>;<PatchID>;<DateApplied>;<Description>
```

Example 1-1 Using WLST

The following example shows using WLST to connect to a server instance and obtain its list of applied patches:

```
wls:/offline> connect('username','password','t3://localhost:7001')
Connecting to t3://localhost:7001 with userid weblogic ...
Successfully connected to Admin Server "myserver" that belongs to domain "mydomain".

Warning: An insecure protocol was used to connect to the server.
To ensure on-the-wire security, the SSL port or Admin port should be used instead.

wls:/mydomain/serverConfig/> serverRuntime()
Location changed to serverRuntime tree.
 This is a read-only tree with ServerRuntimeMBean as the root.
For more help, use help('serverRuntime').

wls:/mydomain/serverRuntime/> print cmo.getPatchList()
array(java.lang.String,['24907328;20845986;Mon Mar 13 14:40:42 PDT 2017;WLS PATCH
SET UPDATE 12.2.1.1.170117', '19795066;19149348;Mon Mar 13 14:33:28 PDT 2017;One-
off', '18905788;18668039;Mon Mar 13 14:32:57 PDT 2017;One-off',
'19632480;19278519;Mon Mar 13 14:32:26 PDT 2017;One-off', '19002423;18804275;Mon Mar
13 14:31:50 PDT 2017;One-off', '19030178;19234068;Mon Mar 13 14:31:22 PDT 2017;One-
off', '19154304;19278518;Mon Mar 13 14:30:54 PDT 2017;One-off'])
wls:/mydomain/serverRuntime/>
```

Example 1-2 Using the REST API

The following example shows using the REST API to return the patch list:

```
Request:
http://localhost:7001/management/weblogic/latest/serverRuntime?
links=none&fields=name,patchList

Response: {
    "patchList": [
        "24907328;20845986;Mon Mar 13 14:40:42 PDT 2017;WLS PATCH SET UPDATE
12.2.1.1.170117",
        "19795066;19149348;Mon Mar 13 14:33:28 PDT 2017;One-off",
        "18905788;18668039;Mon Mar 13 14:32:57 PDT 2017;One-off",
        "19632480;19278519;Mon Mar 13 14:32:26 PDT 2017;One-off",
        "19002423;18804275;Mon Mar 13 14:31:50 PDT 2017;One-off",
        "19030178;19234068;Mon Mar 13 14:31:22 PDT 2017;One-off",
        "19154304;19278518;Mon Mar 13 14:30:54 PDT 2017;One-off"
    ],
    "name": "myserver"
}
```

Example 1-3 Using the WebLogic Server Administration Console

To use the WebLogic Server Administration Console:

1. In the left pane of the Console, expand **Environment** and select **Servers**.
2. Select the name of the server whose applied patch list you want to view.
3. Select **Configuration** > **General** > **Monitoring**.

The list of applied patches is displayed beneath the field labeled **Patch List**.

Example 1-4 Using a JMX Client

Using a JMX application, you can access the applied patch list of a WebLogic Server instance by invoking the `getPatchList` method, as in the following example:

```
/**
 * @include-api for-public-api
 * Returns array of informational strings for installed patches. Each info string
 * is of the form: <bug-id>;<patch-id>;<date-applied>;<patch-description>
 * For example:
 * 24907328;20845986;Mon Mar 13 14:40:42 PDT 2017;WLS PATCH SET UPDATE
12.2.1.1.170117
 *
 * @return Array of informational strings for installed patches at a server.
 * @roleAllowed Monitor
 * @unharvestable
 */
public String[] getPatchList();
```

2

Roadmap for Upgrading Your Application Environment

Use the upgrade roadmap to identify the procedure required to upgrade your WebLogic application environment. An upgrade of WebLogic application environment is complete when you upgrade, configure, and deploy your WebLogic application environments.

This document describes the upgrade process for Oracle product installations that include only WebLogic Server. If your installation includes other Oracle Fusion Middleware products, prior to beginning the upgrade, refer to *Planning an Upgrade of Oracle Fusion Middleware* and the upgrade guides for each Fusion Middleware product in your installation.

Complete the following steps to upgrade your application environment:

- Plan for an Upgrade
- Prepare for the Upgrade
- Upgrade Your Application Environment
- Procedure for Upgrading an Application Environment
- What to Do If the Upgrade Process Fails

2.1 Plan for an Upgrade

Before upgrading your WebLogic application environment, plan the upgrade path. Planning the upgrade path includes generating an inventory of the application environment, verifying the supported system configurations, reviewing the compatibility information of application environment, and creating an upgrade plan.

To ensure that your plan addresses all 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

2.1.1 Step 1: Inventory the Application Environment

Generate an inventory of the application environment by identifying the following components:

- Administration Server and the computer on which it resides
- Managed Servers and the computer(s) on which they reside
- Location of the applications (including all external client applications)

- External resources, for example:
  - Databases used to store persisted and application data
  - Firewalls
  - Load balancers
- Tools, scripts, templates, and source code used for automating the tasks required to create the application environment

You can view a sample application environment in Overview of the Upgrade Process.

2.1.2 Step 2: Verify Supported Configuration Information

Supported configurations (for example, JDK versions, Operating System versions, Web server versions, and database versions) have changed for WebLogic Server 12.2.1.3.0. You may be required to upgrade your environments to the supported versions of these and other products.

For information about supported configurations, see *Oracle Fusion Middleware Supported System Configurations* on Oracle Technology Network (OTN).

For databases, note that:

- WebLogic Server 12.2.1.3.0 supports PointBase 5.7. However, PointBase is no longer included in the WebLogic Server installation program. Derby replaces PointBase for running WebLogic Server samples.

  To upgrade to WebLogic Server 12.2.1.3.0, you must create a new WebLogic Server installation. Therefore, the PointBase installation directory is not included. To continue using PointBase, see Upgrading a Domain that Uses an Evaluation Database.

  Note: The pre-5.7 version of PointBase that was distributed with earlier versions of WebLogic Server can be used only for WebLogic domains.
- As of WebLogic Server 10.3.3, the evaluation database available from the installation program that is provided for use by the sample applications and code examples, and as a demonstration database, is changed from PointBase to Derby. Derby is an open source relational database management system based on Java, JDBC, and SQL standards. For more information about Derby, see `http://db.apache.org/derby/`.

  If you have a domain based on PointBase from an earlier version of WebLogic Server, and you plan to upgrade that domain to WebLogic Server 12.2.1.3.0, you can continue to use PointBase. But you must obtain a license from `http://www.pointbase.com` to use it. See Upgrading a Domain that Uses an Evaluation Database.
- As of WebLogic Server 10.3, the Oracle Thin Drivers are included as part of the WebLogic Server installation.
- If you are using an Oracle OCI database driver and want to change to use a Thin database driver, you must remove the `server` property (as illustrated below) from the generated JDBC module. For example:

```
<property>
  <name>server</name>
  <value>servername</value>
</property>
```

- The Oracle Thin Drivers are installed with WebLogic Server and are ready for use. For more information about using these drivers, see JDBC Drivers Installed with WebLogic Server in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.1.3 Step 3: Review the Compatibility Information

Most existing WebLogic Server applications can be run without modification in the new WebLogic Server 12.2.1.3.0 application environment. However, you should review WebLogic Server 12.2.1.3.0 Compatibility with Previous Releasesto determine whether any feature changes affect the applications in your environment.

2.1.4 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. Note the following points:

- Oracle does not recommend upgrading an application environment that is currently deployed in production. Instead, you should upgrade your application environment while it is under development or test and execute standard procedures for quality assurance and performance tuning before promoting the upgraded environment to production.
- If your application is complex, for example, if it includes multiple clustered domains and a large number of deployed applications, you may choose to upgrade the components of the application environment in stages.
- You may consider limiting the number of WebLogic Server versions used in any single application environment to minimize the diversity and cost of systems being administered.
- If you plan to use the RDBMS security store in a WebLogic domain, Oracle recommends that you create a new domain in which the RDBMS security store is configured. If you have an existing domain in which you want to use the RDBMS security store, you should create the new domain, and then migrate your security realm to it. Oracle does not recommend "retrofitting" the RDBMS security store to an existing domain. See Managing the RDBMS Security Store in *Administering Security for Oracle WebLogic Server*.

2.2 Prepare for the Upgrade

Before you start the upgrade process, you should verify whether there are any upgrade compatibility issues that apply to your applications. You then shut down all running server instances and back up the application components in your domain.

Complete the following tasks before you upgrade the application environment:

- Step 1: Check Your Applications (Undeploy If Necessary)
- Step 2: Shut Down Servers in the Application Environment
- Step 3: Back Up the Application Environment
- Step 4: Install Required Oracle Products

- Step 5: Set Up the Environment

2.2.1 Step 1: Check Your Applications (Undeploy If Necessary)

It is not necessary for WebLogic Server applications to be undeployed before upgrading the domain. In most cases, WebLogic Server applications can be run without modifications in the new WebLogic Server 12.2.1.3.0 application environment. To determine whether any features changes affect the applications in your environment, review the compatibility information in WebLogic Server 12.2.1.3.0 Compatibility with Previous Releases. Note that if you use deprecated or removed APIs in the application, you might encounter warnings or exceptions at run time.

2.2.2 Step 2: Shut Down Servers in the Application Environment

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

2.2.3 Step 3: Back Up the Application Environment

Oracle recommends that before upgrading your application environment, you manually back up the components defined in Table 2-1. You should back up the relevant information on all machines in the domain.

Table 2-1 Recommendations for Backing Up the Application Environment

Component	Recommendations
Domain directory	Back up the Administration Server and any remote Managed Server domain directories that are defined in the application environment. **Note:** The Domain Upgrade Wizard, which automatically backed up the domain being upgraded, is no longer provided with WebLogic Server. You must manually back up your domain directory prior to upgrading the domain.
Applications and application-persisted data	Back up any applications and data that reside outside of the domain directory.
Log files	If it is important for you to maintain a record of all messages that are logged, back up the log files. As log files can be large, you may want to delete them to conserve disk space if it is not important to retain them.

2.2.4 Step 4: Install Required Oracle Products

Before upgrading your application environment, you must install the Oracle WebLogic Server 12.2.1.3.0 products that you require on each computer in the domain. For more information about installing Oracle WebLogic products, see *Installing and Configuring Oracle WebLogic Server and Coherence*.

2.2.5 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 perform this step by running the *WL_HOME*\server\bin\setWLSEnv script.

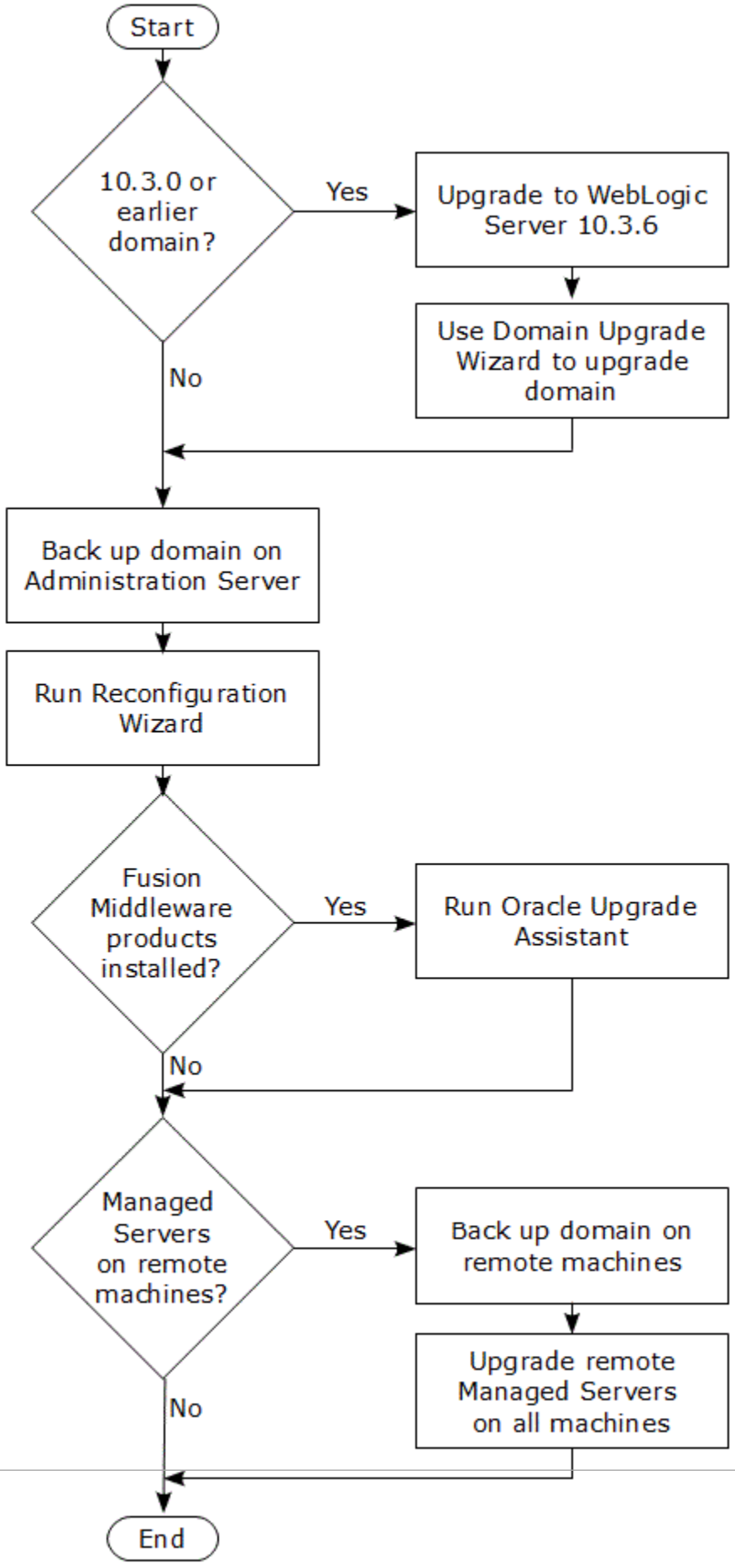
> **Note:**
>
> On UNIX operating systems, the setWLSEnv.sh command does not set the environment variables in all command shells. Oracle recommends that you execute this command using the Korn shell or bash shell.

2.3 Upgrade Your Application Environment

The specific upgrade steps you perform depend upon whether you are upgrading from WebLogic Server 10.3.0 or earlier, whether other Fusion Middleware products are installed, and whether Managed Servers are installed on remote machines.

Figure 2-1 shows the steps required to upgrade your application environment.

Figure 2-1 Roadmap for Upgrading Your Application Environment



2.4 Procedure for Upgrading an Application Environment

To upgrade your application environment to the latest version of WebLogic Server, you may first need to upgrade to 10.3.6. You then back up the domain, upgrade the Administration Server host machine, configure the Node Manager, and upgrade each Managed Server instance.

The Table 2-2table summarizes the steps for updating an application environment. Each step that is performed must be done on every computer in the domain and in the given sequence shown in this table.

Table 2-2 Procedure for Upgrading an Application Environment

Task	Description
Upgrade to WebLogic Server 10.3.6	If the domain was created prior to WebLogic Server 10.3.0, you must first upgrade to WebLogic Server 10.3.6. You can do this using the WebLogic Server 10.3.6 upgrade installer. See *Installation Guide for Oracle WebLogic Server 10.3.6*.
Run the Domain Upgrade Wizard	If the domain was created prior to WebLogic Server 10.3.0, run the WebLogic Server 10.3.6 Domain Upgrade Wizard to upgrade the domain. See Upgrading Domains Created Prior to WebLogic Server 10.3.0.
Back up the domain	Before upgrading the domain on the Administration Server, ensure to backup the domain. See Backing Up the Domain.
Upgrade WebLogic domain (Administration Server)	Run the Reconfiguration Wizard to upgrade the WebLogic domain on the computer that hosts the Administration Server. See Reconfiguring a WebLogic Domain. **Notes:** Oracle recommends that you completely upgrade the domain on the Administration Server before upgrading the domain on the Managed Servers. Depending on the Node Manager configuration of the original domain and the desired Node Manager configuration of the upgraded domain, you may be able to upgrade Node Manager by using the Reconfiguration Wizard. See Determining Node Manager Upgrade Procedure.
Complete Node Manager configuration	If necessary, configure Node Manager as a per host Node Manager. This is needed only if your existing domain is using a per host Node Manager configuration and you want to continue using a per host Node Manager in the upgraded domain. See Completing the Node Manager Configuration.
Back up the domain on each Managed Server.	Prior to upgrading the domain on a Managed Server, make a backup copy of the domain. See Backing Up the Domain.
Upgrade WebLogic domain (remote Managed Servers)	Use the `pack` and `unpack` commands or the WLST `writeTemplate()` command in online mode to upgrade the WebLogic domain on every computer that hosts any Managed Servers. **See Also**: • Updating a Managed Server Domain on a Remote Machine • *Creating Templates and Domains Using the Pack and Unpack Commands* **Note:** • The `unpack` command works only with the same version used to `pack` the WebLogic domain. • Managed Servers that reside on the same computer as the Administration Server do not require additional upgrade steps.

2.5 What to Do If the Upgrade Process Fails

If any step in the upgrade process fails, the Reconfiguration Wizard displays a message indicating the reason for the failure, and then terminates the upgrade process.

To proceed with the upgrade process, perform the following steps:

1. Restore the application environment to its original state using the backup files you created in Step 3: Back Up the Application Environment.
2. Correct the failure reported by the Reconfiguration Wizard.
3. Run the Reconfiguration Wizard again to upgrade the domain.

3
Reconfiguring WebLogic Domains

You can use the Reconfiguration Wizard to upgrade any WebLogic domain that was created with WebLogic Server 10.3.1 or later, or from WebLogic Server 12.1.1 or later.

When you use the Reconfiguration Wizard to reconfigure a WebLogic Server domain, the following items are automatically updated, depending on the applications in the domain:

- WLS core infrastructure
- Domain version

> **Note:**
>
> The Reconfiguration Wizard does not update any of your applications that are included in the domain. For information about how to upgrade your applications, see WebLogic Server 12.2.1.3.0 Compatibility with Previous Releases.

Learn how to use the Reconfiguration Wizard to reconfigure WebLogic Server domains.

3.1 Before You Begin

Before you run the Reconfiguration Wizard, make sure the domain is upgraded to version WebLogic Server 10.3.6, if necessary. You then perform additional tasks, such as configuring the `CONFIG_JVM_ARGS` environment variable, backing up the domain, and choosing the Node Manager configuration that you want to use with the upgraded domain.

3.1.1 Upgrading Domains Created Prior to WebLogic Server 10.3.0

Domains created with WebLogic Server versions 10.3.0 and later can be upgraded directly to version 12.2.1.3.0. However, if your domain was created prior to WebLogic Server 10.3.0, you must first upgrade it to WebLogic Server 10.3.6.

After upgrading to WebLogic Server 10.3.6, run the Domain Upgrade Wizard to upgrade the domain.

To upgrade to WebLogic Server 10.3.6 and run the Domain Upgrade Wizard, see *Upgrade Guide for Oracle WebLogic Server 10.3.6* at the following URL:

`http://docs.oracle.com/cd/E23943_01/web.1111/e13754/toc.htm`

3.1.2 Setting CONFIG_JVM_ARGS on UNIX and Linux Systems

Prior to running the Reconfiguration Wizard to reconfigure a domain on a UNIX or Linux operating system, if you have not already done so, set the `CONFIG_JVM_ARGS` environment variable to the following value to use the operating system's random number generator:

```
-Djava.security.egd=file:/dev/./urandom
```

This decreases the amount of time it takes for the Reconfiguration Wizard to reconfigure a domain.

3.1.3 Backing Up the Domain

Prior to running the Reconfiguration Wizard, make a backup copy of the domain directory. For example, copy C:\domains\mydomain to C:\domains\mydomain_backup.

Prior to updating the domain on each remote Managed Server, make a backup copy of the domain directory on each remote machine.

If domain reconfiguration fails for any reason, you must copy all files and directories from the backup directory into the original domain directory to ensure that the domain is returned entirely to its original state prior to reconfiguration.

3.1.4 Determining Node Manager Upgrade Procedure

Prior to WebLogic Server 12.1.2, a default Node Manager configuration was provided by WebLogic Server via a default startup script and a default Node Manager home location. By default, any new domains that were created on that machine were associated with the Node Manager in the default Node Manager location. This is commonly referred to as a per host Node Manager.

As of WebLogic Server 12.1.2, Node Manager default configuration is a per domain Node Manager configuration, that is, the Node Manager configuration is specific to a given domain. This allows multiple domains on a given machine to have different Node Manager configurations. See Default Node Manager Configuration in *Administering Node Manager for Oracle WebLogic Server*.

Table 3-1 shows the supported Node Manager upgrade paths when upgrading WebLogic Server from version 12.1.1 or earlier to the current version or when upgrading from version 12.1.2 or greater to the current version. Per host in this context means any Node Manager configuration that is outside of your per domain Node Manager configurations.

Table 3-1 Supported Node Manager Upgrade Paths

Node Manager Upgrade Paths	From WebLogic Server 12.1.1 or earlier	From WebLogic Server 12.1.2 or greater
Per domain to per domain	Not available	Supported

Table 3-1 (Cont.) Supported Node Manager Upgrade Paths

Node Manager Upgrade Paths	From WebLogic Server 12.1.1 or earlier	From WebLogic Server 12.1.2 or greater
Per domain to per host	Not available	Not supported
Per host to per domain	Supported	Supported
Per host to per host	Manual configuration	Manual configuration

Table 3-2 shows the Node Manager upgrade details for each supported upgrade path.

Table 3-2 Node Manager Upgrade Details

Per Domain to Per Domain	Per Host to Per Domain	Per Host to Per Host
This is an automatic upgrade for all WebLogic Server 12.1.2 or later releases that are already configured for per domain Node Manager. The environment is updated to standard settings and can be customized later. The upgrade is automatic whether you are using the Reconfiguration Wizard or WLST to upgrade the domain.	In this case, the Reconfiguration Wizard provides a Node Manager screen during domain reconfiguration. Use this screen to select the Node Manager configuration to use for the reconfigured domain. The resulting configuration depends on the combination of options that you select for **Node Manager Type** and **Node Manager Configuration**. You can also use WLST to upgrade the domain and Node Manager configuration as desired. See Reconfiguring a WebLogic Domain Using WebLogic Scripting Tool. If multiple per domain Node Managers run on the same machine, see Configuring Multiple Per Domain Node Managers on the Same Machine. Click the **Help** button on the screen on the Fusion Middleware Reconfiguration Wizard window to see the *Reconfiguration Wizard Context-Sensitve Help*.	Node Manager configuration must be completed manually as described in Completing the Node Manager Configuration. If only some 11g domains are upgraded to 12c, you would need to configure a second Node Manager for the 12c domains. Before starting the reconfiguration process, see Running Two Per Host Node Managers on the Same Machine. After domain reconfiguration, complete the Node Manager configuration as described in Completing the Node Manager Configuration (Two Per Host Node Managers).

3.1.4.1 Configuring Multiple Per Domain Node Managers on the Same Machine

If you have multiple domains on the same machine using a Per Domain Node Manager configuration, when running the Reconfiguration Wizard, do the following:

- If you are upgrading an 11g domain, the Node Manager screen appears. Select either **Per Domain Default Location** or **Per Domain Custom Location**.
- On the Advanced Configuration screen, select **Managed Servers, Clusters, and Coherence** to reconfigure the existing machines for the 12c Node Manager.
- No changes are needed on the Managed Servers and Clusters screens. When the Machines screen appears, ensure that you use a unique Node Manager port for each domain. For example, if you have three per domain Node Managers running on the machine, use port 5556 for Domain 1, port 5557 for Domain 2, and port 5558 for Domain 3.

Click the **Help** button on the screen on the Fusion Middleware Reconfiguration Wizard window to see the *Reconfiguration Wizard Context-Sensitve Help* .

3.1.4.2 Running Two Per Host Node Managers on the Same Machine

If all the following items apply to your upgrade scenario, extra steps are needed during the reconfiguration process to create a second Node Manager for the 12c domains:

- You want to upgrade only some of your 11g domains to 12c.
- You want to continue using a per host Node Manager for the 12c domains.
- Both the 11g and 12c per host Node Managers are running on the same machines.

When running the Reconfiguration Wizard:

- On the Node Manager screen, select **Manual Node Manager Setup**. This option keeps the Node Manager configuration as a per host Node Manager for the 12c domain being upgraded.
- On the Advanced Configuration screen, select **Managed Servers, Clusters, and Coherence** to reconfigure the existing machines for the 12c Node Manager. In addition, select **Deployments and Services** to check machine assignments for your deployments and services.
- No changes are needed on the Managed Servers and Clusters screens. When the Machines screen appears, change the name of each machine to something other than the name that is being used for the 11g domains. In addition, enter a Node Manager port number that is different than the Node Manager port number that is being used for the 11g Node Manager. Use the same port number for each 12c machine in this domain.
- Verify that your deployments and services are assigned to the new machine names.

3.2 Reconfiguring a WebLogic Domain

Oracle provides a choice of two tools for reconfiguring a WebLogic domain: the graphical Fusion Middleware Reconfiguration Wizard or the WebLogic Scripting Tool (WLST).

> **Caution:**
>
> Once the domain reconfiguration process starts, it is irreversible. Before using the Reconfiguration Wizard or WLST to upgrade the domain, ensure that you have backed up the domain as described in Backing Up the Domain. If an error or other interruption occurs during the reconfiguration process, you must restore the domain by copying the files and directories from the backup location to the original domain directory. This workaround is the only way to ensure that the domain has been returned to its original state before reconfiguration.

When you reconfigure a domain:

- The domain version number in the config.xml file for the domain is updated to the Administration Server's installed WebLogic Server version major and minor version number (for example, 12.2.1.0).
- Reconfiguration templates for all installed Oracle products are automatically selected and applied to the domain. These templates define any reconfiguration tasks that are required to make the WebLogic domain compatible with the current WebLogic Server version.
- Start scripts are updated.
- After reconfiguring the domain on the Administration Server, you must port the reconfigured domain to all remote Managed Servers in the domain. See Updating a Managed Server Domain on a Remote Machine.
- After reconfiguring a domain to a per host Node Manager by using either WLST or the Reconfiguration Wizard, you must take additional steps to complete the Node Manager configuration. See Completing the Node Manager Configuration and Completing the Node Manager Configuration (Two Per Host Node Managers).

3.2.1 Reconfiguring a WebLogic Domain in Graphical Mode

To reconfigure a domain using the Reconfiguration Wizard, you first launch it from a DOS command prompt or UNIX shell, and then provide the required upgrade details in a sequence of screens that are displayed.

> **Note:**
>
> If you cannot run the Reconfiguration Wizard in GUI mode, Oracle recommends that you use a WLST script to reconfigure your domain. See Reconfiguring a WebLogic Domain Using WebLogic Scripting Tool.

To start the Reconfiguration Wizard in graphical mode from a Windows command prompt or on UNIX systems:

1. Log in to the system on which the domain resides.
2. Open an MS-DOS command prompt window (on Windows) or a command shell (on UNIX).
3. Go to the following directory, where `ORACLE_HOME` is your Oracle home directory:

   On Windows: `ORACLE_HOME\oracle_common\common\bin`

   On UNIX: `ORACLE_HOME/oracle_common/common/bin`
4. Run the following commands:

   On Windows: `reconfig.cmd`

   On UNIX: `sh reconfig.sh`

> **Note:**
>
> When you run the `reconfig.cmd` or `reconfig.sh` command, the following error message appears if the default cache directory is not valid:
>
> `*sys-package-mgr*: can't create package cache dir`
>
> You can change the cache directory by including the `-Dpython.cachedir=valid_directory` option in the command.
>
> To create a log file of the Reconfiguration Wizard session, include the `-log=reconfig.log -log_priority=debug` parameter in the command. You can specify any file name for the log file, such as `config_today.log`. The log file is stored in the *logs* directory of the Oracle Home directory. Other valid values for `log_priority` are `OFF`, `SEVERE`, `WARNING`, `INFO`, `CONFIG`, `FINE`, `FINER`, `FINEST`, and `ALL`.

The Select Domain screen appears.

The Reconfiguration Wizard displays a sequence of screens in the order listed in Table 3-3.

> **Note:**
>
> Depending on the applications in your domain and other factors, extra configuration screens appear in addition to the screens shown in the following table. For information on these screens, click the **Help** button on the screen.
>
> If the Advanced Configuration screen appears during the reconfiguration process, do not select any options to skip all advanced configuration. If necessary, you can use WLST, the Configuration Wizard, or the WebLogic Server Administration Console later to perform advanced configuration such as adding more servers and clusters or changing deployment targeting.

Table 3-3 Reconfiguring an Existing WebLogic Domain

Screen	When Does This Screen Appear?	Perform the Following Action
Select Domain	Always	Enter the full path to the domain directory or click **Browse** to navigate to and select the domain directory. Click **Next** to continue.
Reconfiguration Setup Progress	Always	Shows the progress of the application of reconfiguration templates. When the process completes, click **Next** to continue.
Reconfiguration Summary	Always	Displays the information about the reconfiguration process for all the reconfigured templates. Click **Next** to continue.

Table 3-3 (Cont.) Reconfiguring an Existing WebLogic Domain

Screen	When Does This Screen Appear?	Perform the Following Action
Domain Mode and JDK	Always	Domain mode cannot be changed. Select the JDK to use in the domain or click **Browse** to navigate to the JDK you want to use. Click **Next** to continue.
Additional domain configuration screens may appear at this point	Additional screens depend on the domain configuration	Click the **Help** button on the screen or refer to Reconfiguration Wizard Screens, which describes all the screens in the order in which they are displayed.
Advanced Configuration	Always	Select the check box for each category (if any) for which you want to perform advanced configuration tasks. The available check boxes depend on the domain configuration. Click **Next** to continue.
Configuration Summary	Always	Review the configuration. Click the **Back** button to change the configuration or click the **Reconfig** button to complete the domain reconfiguration process.
Reconfiguration Success	Always	Shows the final status of the reconfiguration process. Click **Finish** to exit the Configuration Wizard.

3.2.2 Reconfiguring a WebLogic Domain Using WebLogic Scripting Tool

To reconfigure a domain using WLST, you use the `readDomainForUpgrade` command. You can also use this command to migrate an existing per host Node Manager configuration to a per domain configuration.

> **Note:**
>
> If the original domain is using a per domain Node Manager configuration, Node Manager is upgraded automatically and no additional steps are needed.
>
> If the original domain is using a per host Node Manager, and you want to continue using that configuration, you must manually reconfigure Node Manager as described in Completing the Node Manager Configuration.

Example 3-1 shows how to reconfigure a domain called `my_domain` using WLST offline.

Example 3-2 shows how to migrate an existing per host Node Manager configuration to a per domain configuration located in *DOMAIN_HOME*/nodemanager.

Example 3-3 shows how to migrate an existing per host configuration located in `/Oracle/Middleware/oracle_common/common/nodemanager` to a per domain configuration located in `/Oracle/Middleware/custom/nodemanager`.

For information about available Node Manager options for the `setOption` command, see setOption in *WLST Command Reference for WebLogic Server*. For information about available Node Manager WLST commands, see Node Manager Commands in *WLST Command Reference for WebLogic Server*.

Example 3-1 Reconfiguring a WebLogic Domain

```
# Open the domain for upgrade.
wls:/offline> readDomainForUpgrade('c:/domains/my_domain')

# Save the updated domain.
wls:/offline/my_domain> updateDomain()

# Close the domain.
wls:/offline/my_domain> closeDomain()
```

If your existing domain is using a per host Node Manager and you want to move to a per domain Node Manager configuration, you have several options:

- Create a per domain configuration in the default location (`DOMAIN_HOME/nodemanager`) by migrating an existing per host configuration.
- Create a per domain configuration in the default location (`DOMAIN_HOME/nodemanager`) with a new configuration based on Oracle-recommended defaults.
- Create a per domain configuration in a custom location by migrating an existing per host configuration.
- Create a per domain configuration in a custom location with a new configuration based on Oracle-recommended defaults.

Example 3-2 Creating a New Node Manager Configuration in the Default Location

```
#Read domain for reconfiguration
readDomainForUpgrade('domains/mydomain')

#Set Node Manager username and password.
cd('/')
cd('SecurityConfiguration/mydomain')
cmo.setNodeManagerUsername('username')
cmo.setNodeManagerPasswordEncrypted('password')

#Browse Node Manager properties
cd('/')
cd('NMProperties')

# Create per domain Node Manager with new default configuration. Existing
# Node Manager properties will not be migrated in this case.
setOption('NodeManagerType','PerDomainNodeManager')
setOption('NodeManagerUpgradeType','New')

# Update the domain to commit the changes.
updateDomain()
```

Example 3-3 Migrating an Existing Configuration to a Custom Location

```
#Read domain for reconfiguration
readDomainForUpgrade('/domains/mydomain')

#Set Node Manager username and password.
cd('/')
cd('SecurityConfiguration/mydomain')
cmo.setNodeManagerUsername('username')
cmo.setNodeManagerPasswordEncrypted('password')

#Browse node manager properties
cd('/')
cd('NMProperties')

# Create custom location Node Manager, migrating an existing Node Manager
# configuration with default values for Oracle-recommended default properties.
setOption('NodeManagerType','CustomLocationNodeManager')
setOption('NodeManagerHome','/Oracle/Middleware/custom/nodemanager/')
setOption('NodeManagerUpgradeType','Migrate')
setOption('OldNodeManagerHome','/Oracle/Middleware/Oracle_Home/oracle_common/
common/nodemanager')
setOption('NodeManagerUpgradeOverwriteDefault','true')

# Update the domain to commit the changes.
updateDomain()
```

3.2.3 Completing the Node Manager Configuration

If the domain you reconfigured was using a per host Node Manager configuration and you want to continue using a per host Node Manager for the domain, you must complete a set of configuration tasks for Node Manager.

1. In the new WebLogic Server installation, create the `nodemanager` directory in `ORACLE_HOME/oracle_common/common`.
2. Copy the nodemanager.properties and nodemanager.domains files from the `WL_HOME/common/nodemanager` directory of your previous WebLogic Server installation to the directory you created in Step 1.
3. If your previous installation includes an nm_data.properties, SerializedSystemIni.data, or security/SerializedSystemIni.dat file, copy it to the directory you created in Step 1. If copying SerializedSystemIni.dat, you must create a `security` directory under the `nodemanager` directory and store the file there.
4. Make the following edits to the nodemanager.properties file, where `ORACLE_HOME` is the Oracle home directory for your WebLogic Server installation:
   - Update `DomainsFile` to point to `ORACLE_HOME/oracle_common/common/nodemanager/nodemanager.domains` file.
   - Update `JavaHome` to point to the `jre` directory for the JDK that you are using for WebLogic Server. If the file also contains a `javaHome` property setting (lower-case `j`), remove it as it is not needed.
   - Update `NodeManagerHome` to point to `ORACLE_HOME/oracle_common/common/nodemanager`.
   - Update `LogFile` to point to `ORACLE_HOME/oracle_common/common/nodemanager/nodemanager.log`.

5. If you are using your own security certificates, verify that the location of those certificates is correct in nodemanager.properties. You may have to update the path if you moved the certificates to another location.

   If you were using the WebLogic Server demo certificate in your previous installation, you must run `CertGen` to create a demo keystore for this installation:

   a. Run `setWLSEnv`:

      `cd WL_HOME/server/bin`

      `. ./setWLSEnv.sh` (UNIX)

      `setWLSEnv.cmd` (Windows)

      > **Note:**
      >
      > On UNIX operating systems, the `setWLSEnv.sh` command does not set the environment variables in all command shells. Oracle recommends that you execute this command using the Korn shell or bash shell.

   b. Change to the `ORACLE_HOME/oracle_common/common/nodemanager/` directory and create a security directory if it does not exist.

   c. Change to the security directory and enter the following command:

      ```
      java utils.CertGen -certfile democert -keyfile demokey -keyfilepass
      DemoIdentityPassPhrase
      ```

   d. Tto generate the `DemoIdentity.jks` file, enter the following command:

      ```
      java utils.ImportPrivateKey -certfile democert.pem -keyfile demokey.pem -
      keyfilepass DemoIdentityPassPhrase -keystore DemoIdentity.jks -storepass
      DemoIdentityKeyStorePassPhrase -alias demoidentity
      ```

6. From the `ORACLE_HOME/wlserver/server/bin` directory, run `startNodeManager.cmd` (Windows) or `startNodeManager.sh` (UNIX).

7. Verify that you can start servers using Node Manager. See Using Node Manager to Control Servers in *Administering Node Manager for Oracle WebLogic Server*. To ensure that your `permgen` settings are adequate for starting the servers, you can use any one of the following methods:

   - Start the Managed Servers using the `startManagedWebLogic` script.
   - Set the `StartScriptEnabled` value in `nodemanager.properties` to `true`, which causes the `StartManagedWebLogic` script to be invoked when starting Managed Servers.
   - Set `permgen` space as described in Setting permgen space.
   - Use a `setUserOverrides` script to specify `permgen` settings for server startup. See Customizing Domain Wide Server Parameters in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

3.2.4 Completing the Node Manager Configuration (Two Per Host Node Managers)

If the domain you reconfigured was using a per host Node Manager configuration, you can continue using a per host Node Manager for the 12c domain on a machine that already has a per host Node Manager for 11g domains.

Perform these steps on each machine in the domain.

> **Note:**
>
> Prior to performing the steps in this section, ensure that you have unpacked the domain to each remote machine in the domain. Include the `-nodemanager_type=ManualNodeManagerSetup` and `-overwrite_domain=true` parameters in the command. For example:
>
> ```
> ./unpack.sh -domain=domain_home -template=template_jar -
> nodemanager_type=ManualNodeManagerSetup -overwrite_domain=true
> ```

1. In the new WebLogic Server installation, create the `nodemanager` directory in `ORACLE_HOME/oracle_common/common`.
2. Copy the nodemanager.domains and nodemanager.properties files from the `WL_HOME/common/nodemanager` directory of your previous WebLogic Server installation to the directory you created in Step 1. If any 11g domains are listed in the nodemanager.domains file, delete or comment out those lines.
3. Edit the nodemanager.properties file as appropriate on each machine. In particular:
   - Verify that `SecureListener` is set to `true` if using SSL Node Manager, or that it is set to `false` if using Plain Node Manager.
   - Change `DomainsFile` to point to `ORACLE_HOME/oracle_common/common/nodemanager/nodemanager.domains`.
   - Change `PropertiesVersion` to `12.1`.
   - Change `NodeManagerHome` to `ORACLE_HOME/oracle_common/common/nodemanager`.
   - Change `JavaHome` to point to the jre directory for the Java installation that you are using for WebLogic Server 12.2.1.2.0.
   - Remove the `javaHome` line as it is not needed in 12c.
   - Change `ListenPort` to the value you specified on the Machines screen of the Configuration Wizard.
   - Change `LogFile` to the desired location and file name. The recommended value is `ORACLE_HOME/oracle_common/common/nodemanager/nodemanager.log`.
4. If you are using your own security certificates, verify that the location of those certificates is correct in nodemanager.properties. If you moved the certificates to another location, you have to update the path.

If you used the WebLogic Server demo certificate in your previous installation, you must run `CertGen` to create a demo keystore for this installation:

a. Run `setWLSEnv`:

   `cd WL_HOME/server/bin`

   `. ./setWLSEnv.sh` (UNIX)

   `setWLSEnv.cmd` (Windows)

> **Note:**
>
> On UNIX operating systems, the `setWLSEnv.sh` command does not set the environment variables in all command shells. Oracle recommends that you execute this command using the Korn shell or bash shell.

b. Change to the `ORACLE_HOME/oracle_common/common/nodemanager/` directory and create a security directory if it does not exist.

c. Change to the security directory and enter the following command:

```
java utils.CertGen -certfile democert -keyfile demokey -keyfilepass
DemoIdentityPassPhrase
```

d. To generate the `DemoIdentity.jks` file, enter the following command:

```
java utils.ImportPrivateKey -certfile democert.pem -keyfile demokey.pem -
keyfilepass DemoIdentityPassPhrase -keystore DemoIdentity.jks -storepass
DemoIdentityKeyStorePassPhrase -alias demoidentity
```

5. From the `ORACLE_HOME/wlserver/server/bin` directory, start Node Manager.
6. If the Administration Server is running, restart the Administration Server.
7. Verify that you can start servers using Node Manager. See Using Node Manager to Control Servers in *Administering Node Manager for Oracle WebLogic Server*. To ensure that your `permgen` settings are adequate for starting the servers, you can use any one of the following methods:
   - Start the Managed Servers using the `startManagedWebLogic` script.
   - Set the `StartScriptEnabled` value in `nodemanager.properties` to `true`, which causes the `StartManagedWebLogic` script to be invoked when starting Managed Servers.
   - Set `permgen` space as described in Setting permgen space.
   - Use a `setUserOverrides` script to specify `permgen` settings for server startup. See Customizing Domain Wide Server Parameters in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

3.3 Updating a Managed Server Domain on a Remote Machine

If your WebLogic domain contains multiple Managed Servers, and each Managed Server domain directory is located on a machine that is remote to the Administration

Server host machine, you can use one of two methods to update the domain on the remote machine.

- Use the `pack` command to generate the domain template JAR. Ensure that you include the `-managed=true` argument in the `pack` command. Move the JAR to the remote machine and then use the `unpack` command on the remote machine to create the Managed Server domain. See Creating Templates and Domains Using the Pack and Unpack Commands.
- Use the WLST `writeTemplate` command in online mode. When you execute the `writeTemplate` command while connected to the Administration Server from a remote machine, it dynamically packs the domain on the Administration Server into a template JAR file and transfers the template JAR to the specified directory.

  The following sample WLST script demonstrates how to use `writeTemplate` to create or update a Managed Server domain on a remote machine. Run the script on each remote machine in the domain. This script does the following tasks:

  – logs in to the Administration Server
  – packs the Administration Server domain into a JAR file and writes it to the specified template directory on the remote machine
  – disconnects from the Administration Server
  – reads the template JAR
  – creates the domain on the remote machine

```
import os

wlsHome = os.getenv('WL_HOME')
mwHome = os.path.join(wlsHome, '..')

#Substitute the administrator user name and password values below as needed
connect('adminuser','adminpassword','admin_server_url')

#Create the path on the local machine where the template will be stored,
#The specified template JAR should not already exist. The timeout value
#specifies the number of milliseconds to elapse before the connection between
#the Administration Server and remote machine times out (default is 120000).
templatePath = '/user_templates/myTemplate.jar'
timeout = '180000'

#get the packed template from the Administration Server
writeTemplate(templatePath, timeout)

#disconnect from online WLST connection to the Administration Server
disconnect()

#read the template that was downloaded from the Administration Server
readTemplate(templatePath)

#specify the domain directory where the domain needs to be created
domainPath = 'domains/myDomain')

#create the domain
writeDomain(domainPath)
```

3.4 Important Notes About the Domain Upgrade Process

Bear in mind several key notes about the domain upgrade process, such as whether it is necessary to undeploy WebLogic Server applications, the minimum set of files that must exist in the domain directory, and more.

- It is not always necessary to undeploy WebLogic Server applications. Usually, WebLogic Server applications can run without modifications in the new WebLogic Server 12.2.1.3.0 application environment. Review the compatibility information in WebLogic Server 12.2.1.3.0 Compatibility with Previous Releases to determine whether any features changes affect the applications in your environment. If APIs that have been deprecated or removed are used in the application, then you might encounter warnings or exceptions at run time.
- At a minimum, the domain directory must contain the following files:
  - `config.xml`
  - Security-related files, including `SerializedSystemIni.dat`, `DefaultAuthenticatorInit.ldift`, `DefaultAuthorizerInit.ldift`, and `DefaultRoleMapperInit.ldift`

    If the security-related files are not available, the server fails to start and an authentication error message is logged.
  - Any transaction log (`.tlog`) files that reside in the domain. See Transaction Log Files in *Developing JTA Applications for Oracle WebLogic Server*.
- All contents of the domain directory on the target computer are updated during this process.
- You must upgrade the domain on every computer in the application environment.
- The reconfiguration wizard does not upgrade your own applications that may exist in the domain during a WebLogic domain upgrade.
- Domains that contain resources for WebLogic Liquid Data, or AquaLogic Data Services Platform cannot be upgraded to WebLogic Server 12.2.1.3.0.
- When you upgrade a domain on a remote Managed Server, a message similar to the following may appear to indicate that the referenced application path does not reside on the system:

  ```
  <Apr 12, 2009 6:42:06 PM EDT> <INFO> <Upgrade> <BEA-800000> <An invalid
  path, 'C:\bea\wlserver_10.3\user_projects\mydomain\medrecEar.ear', was
  specified for application, 'medrecEar'.>
  ```

  You can ignore this message.
- If you upgraded the Avitek Medical Records application from version 8.1 to 12.1.3 on a Solaris computer (only), before starting the server, you must edit the `setDomainEnv.sh` file to remove `-Xverify:none` from the start command. To remove the `-Xverify:none`, set `JAVA_OPTIONS=""` after the following line:

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

  Otherwise, the server start fails with a JVM error.

3.5 Completing Post-Upgrade Tasks

After you upgrade the application environment, it may be necessary to perform tasks such re-applying customizations to startup scripts, verifying file permissions and remote server startup options, and more.

This section includes the following topics:

- Re-apply Customizations to Startup Scripts
- Verify File Permissions
- Verify Remote Server Startup Options
- Recreating the Windows Node Manager Service
- Promote the Application Environment to Production

Not all these steps are required for all situations. Review the sections to determine which, if any, of these steps are appropriate for your environment. In addition, you should review the compatibility issues in WebLogic Server 12.2.1.3.0 Compatibility with Previous Releases to determine if any of the compatibility issues apply to your environment.

3.5.1 Re-apply Customizations to Startup Scripts

To complete the upgrade of your application environment to 12.2.1.3.0, it might be necessary to re-apply any customizations to startup scripts. The following sections describe how to customize the default startup scripts as well as any custom startup scripts.

3.5.1.1 Default Startup Scripts

The Reconfiguration 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 must customize the default scripts again.

If you are upgrading your domain to 12.2.1.3.0 and you want to continue using PointBase, you must add the PointBase JAR files to the beginning of the CLASSPATH environment variable definition. To do so, update the `set CLASSPATH` statement in your setDomainEnv files.

> **Note:**
>
> WebLogic Server 12.2.1.3.0 supports PointBase 5.7. However, the use of any version of PointBase with WebLogic Server 10.3.3 or later requires a PointBase license, available at http://www.pointbase.com.

3.5.1.2 Custom Startup Scripts

If you have created custom startup scripts, you must update them manually, as follows:

- Set the JDK version to the JDK that you are using with WebLogic Server.
- Update the CLASSPATH variable, as follows:
  - Add WebLogic Server 12.2.1.3.0 classes to the beginning of the variable.
  - Remove all *unused* pre-10.3 WebLogic classes.
  - To continue using PointBase, include the PointBase database JARs at the beginning of the CLASSPATH environment variable definition.

    To know about upgrading a domain that uses an evaluation database, see Upgrading a Domain that Uses an Evaluation Database.

3.5.2 Verify File Permissions

Verify the file permissions, as follows:

- If you backed up the domain directory as part of the upgrade, you must make your backup files secure because they might contain confidential information.
- During the upgrade process, file permissions are not preserved. If nondefault file permissions are set on files, they must 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 or modify the permissions on files created during the upgrade process.

3.5.3 Verify Remote Server Startup Options

When you start the Administration Server, verify that the remote server start options, such as `JAVA_HOME`, `BEA_HOME`, and `CLASSPATH`, reference the WebLogic Server installation on the target Managed Server. This can be accomplished using the WebLogic Server Administration Console, as described in Configure startup arguments for Managed Servers in *Oracle WebLogic Server Administration Console Online Help*.

> **Note:**
>
> 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
```

3.5.4 Recreating the Windows Node Manager Service

On Windows systems, if you were running Node Manager as a Windows service for your domain, you must reconfigure it if you want to continue using it.

For information about how to configure the Node Manager service for Windows, see Default Node Manager Configuration in *Administering Node Manager for Oracle WebLogic Server*.

Optionally, you can remove the Node Manager service from your installation by running `uninstallNodeMgrSrv.cmd`. See Default Node Manager Configuration in *Administering Node Manager for Oracle WebLogic Server*.

3.5.5 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.

When 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).

When the new 12.2.1.3.0 application environment is deployed into production, you can start redirecting requests to the new environment from the existing environment. Gradually, you can bring the existing environment to a safe state for shutdown. This might be accomplished using a load balancer, for example.

3.6 Upgrading a Domain that Uses an Evaluation Database

The evaluation database is provided for use by the sample applications and code examples and may be used as a demonstration database. As of WebLogic 10.3.3, the evaluation database that is available from the installation program is changed from PointBase to Derby.

If you are upgrading an examples or demonstration domain that was originally based on PointBase, you have the option to continue using PointBase in the domain.

To continue using PointBase as the database for a domain being upgraded to WebLogic Server 12.2.1.3.0, complete the following steps:

1. When installing WebLogic Server 12.2.1.3.0 as described in Prepare for the Upgrade, you must use the full installer. The full installer does not preserve the PointBase installation from the previous WebLogic Server installation.
2. Complete the steps described in Upgrade Your Application Environment and Completing Post-Upgrade Tasks.

   The domain's configuration settings for PointBase are preserved.
3. Obtain a PointBase licens, available from `http://www.pointbase.com`.
4. Restore your PointBase installation. PointBase is installed in the `WL_HOME/common/eval/pointbase` directory.

5. Add the PointBase database JARs at the beginning of the CLASSPATH environment variable definition.

4
Upgrading WebLogic Web Services

Learn the procedures for upgrading WebLogic and RESTful Web services from WebLogic Server 10.*x* to the WebLogic Server 12.2.*x* release. Also, learn how to upgrade 8.1 WebLogic Web services to 12.2.*x* WebLogic JAX-WS Web services. This chapter includes the following sections:

- Upgrading a 10.3.*x* RESTful Web Service (JAX-RS) to 12.2.*x*
- Upgrading a 10.x WebLogic Web Service (JAX-WS) to 12.2.*x*
- Upgrading an 8.1 WebLogic Web Service to the WebLogic JAX-WS Stack
- Upgrading a WebLogic JAX-RPC Web Service to the WebLogic JAX-WS Stack

> **Note:**
>
> 10.3.*x* WebLogic Web services (JAX-WS or JAX-RPC) will continue to run, without any changes, on version 12.2.*x* of WebLogic Server because the associated Web services run time is still supported in this release, although they are deprecated and will be removed from the product in future releases. For this reason, Oracle highly recommends that you follow the instructions in this chapter to upgrade your 10.3.*x* Web services to 12.2.*x*.
>
> The JAX-RPC API has been deprecated in 12.2.*x* and will be removed in a future release. Oracle does not recommend upgrading to the JAX-RPC stack.

4.1 Upgrading a 10.3.*x* RESTful Web Service (JAX-RS) to 12.2.*x*

In 10.3.*x*, a set of pre-built shared libraries were delivered with WebLogic Server to support Jersey 1.9 and 1.1.5.1 Java API for RESTful Web Services (JAX-RS) Reference Implementations (RIs). In 12.2.*x*, WebLogic Server supports Jersey 2.21.*x* (JAX-RS 2.0 RI) by default.To use the pre-built shared libraries of 10.3.*x*, you needed to register them with the WebLogic Server instance, and modify the `web.xml` and `weblogic.xml` deployment descriptors to use the Jersey servlet and reference the shared libraries, respectively.
In 12.2.*x*, as WebLogic Server supports Jersey 2.21.*x* (JAX-RS 2.0 RI) by default, registration as a shared library with WebLogic Server is no longer required.

To use the Jersey 2.21.*x* (JAX-RS 2.0 RI), you need to modify your 10.3.*x* RESTful Web service applications as follows:

1. Update your application deployment descriptors to reference the Jersey 2.*x* container. See Servlet-based Deployment in *Jersey 2.21 User Guide*.

> **Note:**
>
> For backward compatibility, references to `com.sun.jersey.spi.container.servlet.ServletContainer`, as shown in the following example, continues to work. However, Oracle recommends that you update your application deployment descriptors to reference the Jersey 2.*x* container instead.

For example, replace `com.sun.jersey.spi.container.servlet.ServletContainer` with `org.glassfish.jersey.servlet.ServletContainer` in the following `<web-app>` content:

```
<web-app>
    <servlet>
        <display-name>My Jersey Application</display-name>
        <servlet-name>MyJerseyApp</servlet-name>
        <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</
servlet-class>
        <init-param>
            <param-name>javax.ws.rs.Application</param-name>
            <param-value>myPackage.myJerseyApplication</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>MyJerseyApp</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>
</web-app>
```

For more advanced configuration options, see Jersey 2.21.1 User Guide.

2. If applicable, update all applications that use Jersey 1.*x* server APIs to use the corresponding standard JAX-RS 2.0 or Jersey 2.*x* APIs instead. Support for the Jersey 1.*x* (JAX-RS 1.1 RI) *server* APIs has been removed in this release and applications that reference them will not work.
3. Update your clients to use the `javax.ws.rs.client` API, as described in Developing RESTful Web Service Clients in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

> **Note:**
>
> Support for the Jersey 1.18 client packages, including the `com.sun.jersey`package, its nested packages, and the `weblogic.jaxrs.api.client` package, is deprecated in this release of WebLogic Server, but are maintained for backward compatibility. However, many Fusion Middleware components, such as Oracle Web Services Manager, have been migrated to the standard JAX-RS 2.0 client API and are not compatible with the Jersey 1.*x* JAX-RS client APIs. Therefore, Oracle strongly recommends that you update your RESTful client applications as soon as possible to use the standard JAX-RS 2.0 API.
>
> The Jersey 1.*x* JAX-RS RI client APIs are not compatible with Jersey 2.*x* (JAX-RS 2.0 RI).

4.2 Upgrading a 10.x WebLogic Web Service (JAX-WS) to 12.2.*x*

No steps are required to upgrade a 10.*x* WebLogic Web service to 12.2.*x*. You can redeploy the JAX-WS Web service to WebLogic Server 12.2.*x* without making any changes or recompiling.

4.3 Upgrading an 8.1 WebLogic Web Service to the WebLogic JAX-WS Stack

The 8.1 WebLogic Web services runtime was removed in the 12.1.2 release. If you are using 8.1 WebLogic Web services, you must upgrade the 8.1 WebLogic Web service applications to the JAX-WS (Java API for XML-Based Web Services) stack. The 8.1 WebLogic Web services rely on Apache XMLBeans for mapping XML elements in SOAP payloads into Java objects and vice versa. XMLBeans are not supported in 12.1.2 and later.

> **Note:**
>
> Upgrade 8.1 Web service to JAX-WS on your WebLogic Server release before you upgrade to 12.2.1.3.0. After you upgrade to 12.2.1.3.0, your 8.1 Web services will no longer work.

Upgrading to the JAX-WS stack allows you to take advantage of the latest technologies and standards support in WebLogic Server. This path requires a manual upgrade process, and the level of effort depends on the nature of the existing 8.1 Web service applications. For example, if the applications have little XMLBeans usage, then the upgrade process is relatively easy. For 8.1 Web Service applications with heavy XMLBeans dependencies, you must modify the business logic in the service implementation to use JAXB classes instead of XMLBeans classes. JAX-WS does not support RPC-encoded style. The 8.1 Web Service applications with RPC-encoded style must adopt more interoperable literal style service contracts.

The WebLogic JAX-WS runtime is based on the JAX-WS 2.2 specification and the Web Services for Java EE v1.3 (JSR 109) specifications. These define annotations that are used in a Java Web Service (JWS) source file to define a Web service. Ant tasks are then used to compile the JWS into a Java class and generate all the associated artifacts. The Java Web Service (JWS) is the core of your JAX-WS web service.

Upgrading your 8.1 Web service includes the following high-level tasks:

- Upgrade any Web service EJBs from 2.*x* to 3.*x*.

  JAX-WS supports EJB 3.0 and 3.*x*. It does not support EJB 2.*x*.
- Rewrite the 8.1 Web service class as a JAX-WS JWS file and map any proprietary 8.x features to similar JAX-WS features.

There is not a one-to-one correspondence between 8.1 Web service features and JAX-WS 12.1.*x* features.

- Update the Ant build script that builds the Web service to call the 12.1.*x* WebLogic Web service Ant task `jwsc` instead of the 8.1 `servicegen` task.
- Generate new JAX-WS clients using the JAX-WS `clientgen` Ant task.

JAX-WS Upgrade Considerations

Before upgrading to JAX-WS, consider the following:

- The JAX-WS specification supports the *document-literal* and *rpc-literal* styles, but not *rpc-encoded*.
- The JAX-WS specification does not support SOAP Arrays.

See Developing JAX-WS Web Services for Oracle WebLogic Server.

4.4 Upgrading a WebLogic JAX-RPC Web Service to the WebLogic JAX-WS Stack

The WebLogic JAX-WS run time is based on the JAX-WS (The Java API for XML-Based Web Services) 2.2 specification and the Web Services for Java EE v1.3 (JSR 109) specifications. Starting with JAX-WS 2.0, the JAX-WS technology has replaced JAX-RPC in the Java Platform and in WebLogic Server. JAX-RPC Web Services in WebLogic applications should be upgraded to JAX-WS.

This section summarizes how to upgrade a WebLogic JAX-RPC Web service to use the WebLogic JAX-WS stack.

Upgrading your WebLogic Server JAX-RPC Web service includes the following high-level tasks:

- Upgrade any Web service EJBs from 2.*x* to 3.*x*.

  JAX-WS supports EJB 3.0 and 3.*x*. It does not support EJB 2.*x*.
- Upgrade your JWS, mapping any proprietary JAX-RPC features to similar JAX-WS features.

  Note that there is not a one-to-one correspondence between WebLogic JAX-RPC Web service features and JAX-WS 12.*x* features.
- Update the Ant build script that builds the Web service to change the value of the `type` attribute on the `jws`, `wsdlc`, and `clientgen` tasks to be `"JAXWS"` (for example, `type="JAXWS"`).
- Generate new JAX-WS clients using the JAX-WS `clientgen` Ant task.

JAX-WS Upgrade Considerations

When upgrading to JAX-WS, you should consider the following:

- The JAX-WS specification supports the *document-literal* and *rpc-literal* styles, but not *rpc-encoded*.
- SOAP Arrays are not supported by JAX-WS.

See Developing JAX-WS Web Services for Oracle WebLogic Server

A

WebLogic Server 12.2.1.3.0 Compatibility with Previous Releases

Learn about important compatibility information that you should consider before upgrading to WebLogic Server 12.2.1.3.0 from a WebLogic Server 10.3.*x* or 12.1.*x* release. Also learn about the feature changes in various WebLogic Server versions that may impact the applications you plan to run in the upgraded environment.
See also:

- WebLogic Server Compatibility in *Understanding Oracle WebLogic Server*. This section provides general information about WebLogic Server compatibility goals and how they apply to this WebLogic Server release.
- *What's New in Oracle WebLogic Server 12.2.1.3.0* for this and prior releases. These documents provide information about new features that are available to you as well as behavior changes that may impact your applications.

Compatibility considerations are provided in the following sections. The sections that apply to your situation depend on the WebLogic Server version from which you are upgrading to WebLogic Server 12.2.1.3.0. See Table A-1for a list of sections to which you should refer based on your current WebLogic Server version.

Table A-1 Sections Applying to Upgrades From Each WebLogic Server Version

If You Are Upgrading From This WebLogic Server Version	Refer to These Sections
12.2.1.0.0	Upgraded Version of Apache Ant Removed the Option to Limit Run-Time Footprint When Starting WebLogic Server
12.1.3	All sections in the above row, plus: Random Number Generator Partitions, Applications, and Container Context Root Assumptions Automatic Binding of the Default CommonJ Work Manager Has Been Removed Parallel Deployment
12.1.2	All sections in the above row, plus: Server Logging Bridge Oracle Database Drivers Oracle Enable JavaNet FastPath
12.1.1	All sections in the above rows, plus: Maximum POST Size WLDF Schema Upgrade jdbc-connection-timeout-secs Element Has Been Removed Commitment of Local Transactions

Table A-1 (Cont.) Sections Applying to Upgrades From Each WebLogic Server Version

If You Are Upgrading From This WebLogic Server Version	Refer to These Sections
10.3.5 and 10.3.6	All sections in the above rows, plus: JVM Settings Node Manager startScriptEnabled Default Value Enterprise Java Beans (EJBs) WebLogic Server 8.1 Web Services Stack Has Been Removed Universal Description and Discover (UDDI) Registry Has Been Removed Certicom SSL Implementation Has Been Removed Oracle Coherence Version Deprecated and Obsolete Web Application Features Evaluation Database Changed From PointBase to Derby DataSource Profile Logging ONS Debugging Oracle Type 4 JDBC drivers from DataDirect Default Message Mode Has Changed
10.3.3 and 10.3.4	All sections in the above rows, plus: Modifications to SSLMBean
10.3.2	All sections in the above rows, plus: New Web Services Features Introduction of JSSE Performance Enhancements for Security Policy Deployment ActiveCache Class Caching Deprecated JDBC Drivers Changes to weblogic.jms.extension API Persistent Store Updates
10.3.1	All sections in the above rows, plus: Oracle Internet Directory and Oracle Virtual Directory Authentication Providers
10.3.0	All sections in the above rows, plus: CapacityIncrement Attribute Middleware Home Directory Resource Registration Name Servlet Path Mapping

A.1 Upgraded Version of Apache Ant

Oracle WebLogic Server 12.2.1.3.0 includes Apache Ant 1.9.8, which may have an impact on the use of the `clientgen` Ant task. The `clientgen` Ant task generates, from an existing WSDL file, the client component files that client applications use to invoke

both WebLogic and non-WebLogic web services. Note the following when using the `<binding>` child element of this Ant task:

- You use the `<binding>` element the same way as the standard Ant FileSet data type, using the same attributes.
- In Apache Ant 1.9.8, the Ant FileSet data type is changed so that it may specify either a single file, or a single directory. Therefore, if you use the `<binding>` element to specify multiple files or directories, the `clientgen` Ant task might fail.

For more information about specifying the `<binding>` child element, see `binding` in *WebLogic Web Services Reference for Oracle WebLogic Server*.

A.2 Removed the Option to Limit Run-Time Footprint When Starting WebLogic Server

When you start a WebLogic Server instance, all services are started including EJB, JMS, Connector, Clustering, Deployment, Management, and so forth. WebLogic Server provides a startup option that you can use to run the lighter weight run-time instance in any WebLogic domain.

This startup mode can result in quicker startup times for WebLogic Server and a smaller resource footprint on the host machine. You can start a lighter weight run-time instance by specifying the following `weblogic.Server` command option:

```
java weblogic.Server -DserverType= {"wlx" | "wls"}
```

As of Oracle WebLogic Server version 12.2.1.0.0, this startup option is removed.

A.3 Random Number Generator

Oracle WebLogic Server 12.2.1.3.0 uses a more secure random number generator algorithm than in previous releases. This can result in slow startup of Managed Servers, Configuration Wizard, Node Manager, and WebLogic Java utilities such as `utils.ImportPrivateKey` on low-entropy systems. Therefore, you should take steps to increase system entropy.

On UNIX systems, you use `rng-tools` to replace system entropy. To configure it, edit `/etc/sysconfig/rngd` and add the following line:

```
EXTRAOPTIONS="-i -r /dev/urandom -o /dev/random -b"
```

You can also use the `-t 60` and `-W 2048` parameters. These parameters add bits to the entropy pool every 60 seconds until the pool reaches the size of 2048.

Use the following command to generate entropy manually:

```
rngd -r /dev/urandom -o /dev/random -b
```

Use the following command to check current entropy:

```
cat /proc/sys/kernel/random/entropy_avail
```

A.4 Partitions, Applications, and Container Context Root Assumptions

Java EE applications that make assumptions concerning the context root of their web container may need to be modified if they are deployed to a virtual target that has a `uriPrefix` set. In this case, the context path of the application includes the `uriPrefix` of the virtual target.

Some examples of this are:

- The application parses the incoming URL to construct another URL. If this parsing assumes that the URL root ends at `host:port`, the application needs to be updated because the URL root will be `host:port/prefix` with URI-based routing.

  The MedRec sample application is one example of this. MedRec used to parse the incoming URL to construct another URL, and assumed that the root URL consisted of only `host:port`. To address this, the following changes were made to the MedRec application:

  – The original code in the `JaxWsProperties.java` file for the physician application was:

  ```
  public final static String WSURL = "http://"
              + ServerPropertiesUtils.getServerAddress("physician",
  "localhost")
              + ":"
              + ServerPropertiesUtils.getServerPort("7001")
              + "/medrec-jaxws-services/";
  ```

  This was changed to the following code. `ServerPropertiesUtils.getRegion()` accounts for the possibility of a partition URI prefix in the URL:

  ```
  public final static String WSURL = "http://"
          + ServerPropertiesUtils.getRegion() + "medrec-jaxws-services/";
  ```

  – The original code in the `GettingHostFilter.java` file for the physician web application was:

  ```
  ServerPropertiesUtils.setAddress(request.getServerName());
  ServerPropertiesUtils.setPort(String.valueOf(request.getServerPort()));
  chain.doFilter(request, response);
  ```

  This was changed to the following code to preserve `host:port` or, if using MT URI-based routing, `host:port/partition` for the web service client:

  ```
  if (ServerPropertiesUtils.getRegion() == null ||
   ServerPropertiesUtils.getRegion().equals("")) {
     StringBuilder builder = new StringBuilder();
     builder.append(request.getServerName());
     builder.append(":");
     builder.append(String.valueOf(request.getServerPort()));
     builder.append(partition);
     ServerPropertiesUtils.setRegion(builder.toString());
  ```

- The application links to "/" in HTML/JSP code. In a non-MT environment, the application may make an assumption about the context root to which it is deployed. For example, consider an application that is deployed with the context root `/fruits` and that includes a page that refers to:

```
<a href="/fruits/index.html">Back to Fruits List</a>
```

This type of absolute reference does not work if the application is deployed to a partition with a virtual target that uses a URI prefix. The preceding link will try to go to *host:port*`/fruits/index.html` instead of *host:port*`/partition1/fruits/index.html`. The safest approach is to use relative URLs in links, such as:

```
<a href="index.html">Back to Fruits List</a>
```

A.5 Automatic Binding of the Default CommonJ Work Manager Has Been Removed

The Work Manager API, `commonj.work`, provides a set of interfaces that allows an application to execute multiple work items concurrently within a container. Automatic binding of the default CommonJ Work Manager to `java:comp/env/wm/default` has been removed in WebLogic Server 12.2.1 because it is not in compliance with the Java EE 7 platform specification.

If you have an application that attempts to use the default CommonJ Work Manager, you can do either of the following:

- Add a resource-ref entry for `wm/default` in a deployment descriptor. For example:

```
<resource-ref>
      <res-ref-name>wm/default</res-ref-name>
      <res-type>commonj.work.WorkManager</res-type>
      <res-auth>Container</res-auth>
</resource-ref>
```

- Have the CommonJ Work Manager injected into the application component. For example:

```
@Resource commonj.work.WorkManager myWorkManager;
```

A.6 Parallel Deployment

WebLogic Server 12.2.1 adds support for parallel deployment of applications and modules, which improves startup and post-running deployment time.

By default, in WebLogic domains that are created with, or upgraded to, WebLogic Server 12.2.1 (or later):

- Parallel deployment of applications is enabled.
- Parallel deployment of modules for all applications in the domain is disabled.

In WebLogic Server 12.1.3 and earlier versions, applications are always deployed serially. The default deployment order is the natural order that is defined in the domain configuration (that is, as established in the `config.xml` file). However, in those earlier WebLogic Server releases, you can explicitly control deployment order by setting the `DeploymentOrder` attribute of the `AppDeploymentMBean`, using the WebLogic Server Administration Console or programmatically as explained in Changing the Deployment Order for Applications and Standalone Modules in *Deploying Applications to Oracle WebLogic Server*. The use of this feature with older releases is important if specific dependencies exist between applications.

If you create a new domain in WebLogic Server 12.2.1.3.0, or upgrade an existing domain to 12.2.1.3.0, you can restore the WebLogic Server 12.1.3 deployment order behavior by disabling the following attributes of the `DomainMBean`:

- `ParallelDeployApplications`: Determines whether applications are deployed in parallel. (This attribute is enabled by default.)
- `ParallelDeployApplicationModules`: Determines whether the modules of applications are deployed in parallel. (This attribute is disabled by default.)

However, disabling the preceding attributes prevents you from being able to take advantage of the significant performance benefits of parallel deployment. Instead of disabling parallel deployment altogether in the domain you are upgrading to WebLogic Server 12.2.1.3.0, Oracle recommends checking to see whether the deployment ordering of any applications or modules has been customized, and if so, whether it is necessary.

See:

- Enabling Parallel Deployment for Applications and Modules in *Deploying Applications to Oracle WebLogic Server*, which contains important considerations regarding application and module dependencies when using parallel deployment.
- Change the server deployment order in *Oracle WebLogic Server Administration Console Online Help*, which explains how to view or change the deployment order of deployments using the WebLogic Server Administration Console.

A.7 Server Logging Bridge

The Server Logging Bridge provides a lightweight mechanism for applications that currently use Java Logging or Log4J Logging to have their log messages redirected to WebLogic logging services. As of WebLogic Server 12.1.3, the Server Logging Bridge is added to the root logger of the `java.util.logging` Logger tree when WebLogic Server starts. Therefore, you no longer need to explicitly configure the Server Logging Bridge.

If you have configured the `weblogic.logging.ServerLoggingHandler` as described in Server Logging Bridge in *Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server*:

- If `weblogic.logging.ServerLoggingHandler` is attached to the root logger, Oracle strongly recommends that you remove it from your `logging.properties` file.
- If `weblogic.logging.ServerLoggingHandler` is attached to a logger other than the root, Oracle strongly recommends that you either remove it from the `logging.properties` configuration, or set the `useParentHandlers` attribute to `false` (for example, `com.foo.barUseParentHandlers=false`).

These situations also apply to Log4J. However, the terminology is different:

- `weblogic.logging.log4j.ServerLoggingAppender` is the bridge for Log4J.
- `useParentHandlers` is called `Additivity` in Log4J. It is configured as `log4j.additivity.com.foo.bar=false` in the log4j.properties file.

A.8 Oracle Database Drivers

As of release 12.1.2, WebLogic Server installation includes the Oracle Database 12c drivers.

This requires the following changes to your applications:

- Replace references to `wlserver/server/lib/ojdbc6.jar` with `${MW_HOME}/oracle_common/modules/features/com.oracle.db.dbc7-no-dms.jar`. Note that this is automatically included in the class path when using `weblogic.jar`.
- Replace references to `wlserver/server/lib/aqapi.jar` with `${MW_HOME}/oracle_common/modules/oracle.jdbc_12.1.0/aqapi.jar`, which also requires that you use `com.oracle.db.jdbc7-no-dms.jar`.

If you want to continue running with the Oracle Database 11g driver JARs, you must:

- add them to the front of the classpath
- move the Oracle Database 12c driver JARs out of the `MW_HOME/oracle_common/modules/oracle.jdbc_12.1.0` directory.

A.9 Oracle Enable JavaNet FastPath

Oracle Enable JavaNet Fastpath enables the Oracle JDBC JavaNet Fastpath to reduce data copies and fragmentation. As of WebLogic Server 12.1.3, this attribute is no longer supported in the WebLogic Server Administration Console.

In previous versions of WebLogic Server, you could configure the **Oracle Enable JavaNet FastPath** attribute on the **Configuration:Oracle** tab in the WebLogic Server Administration Console. To go to the **Configuration:Oracle** tab, click **Domain Structure**, click **Services**, and then click **Data Sources** in the WebLogic Server Administration Console.

A.10 Maximum POST Size

A new session descriptor, `max-save-post-size`, has been added in WebLogic Server 12.1.2, which may impact existing applications. This descriptor sets the maximum size, in bytes, of the POST that is saved or buffered by the application container during FORM authentication.

The default value of the `max-save-post-size` descriptor is 4096 bytes.

If your application posts a form for which the size exceeds 4096 bytes during FORM authentication, you must increase `max-save-post-size` to an appropriate value. Otherwise, a `MaxPostSizeExceededException` occurs in the browser.

A.11 WLDF Schema Upgrade

If you are using a JDBC-based store for WebLogic Diagnostics Framework (WLDF) event and harvester data, you must update or recreate the WLDF tables in your database.

In the `wls_events` table, change the `THREADNAME` column from `varchar(128)` to `varchar(250)`. In the `wls_hvst` table, add the column `WLDFMODULE varchar(250) default NULL`.

This upgrade applies only to WebLogic Server standalone installations. For installations that include Fusion Middleware products, the schema upgrade process is done through the Oracle Upgrade Assistant.

See Configuring a JDBC-Based Store in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

A.12 jdbc-connection-timeout-secs Element Has Been Removed

The `jdbc-connection-timeout-secs` element in the `weblogic.xml` deployment descriptor has been removed in WebLogic Server 12.1.2. If your application configures the `jdbc-connection-timeout-secs` element, you must remove it from the `weblogic.xml` deployment descriptor to prevent deployment of the application from failing.

A.13 Commitment of Local Transactions

As of WebLogic Server 12.1.2, local transactions on non-XA connections that were not committed or rolled back by the application are now explicitly committed by default when the connection is returned to the pool. In addition, the following two parameters have been added to set whether or not local transactions on non-XA and XA connections are committed when the connection pool is closed:

- `-Dweblogic.datasource.endLocalTxOnNonXAConWithCommit=false` can be used to avoid one extra DBMS round-trip with non-XA connections, for applications that are trusted to always complete their transaction explicitly. If this parameter is set to `false`, local transactions on non-XA connections are implicitly committed or rolled back when a connection pool is closed, according to what the particular JDBC driver being used does when `setAutoCommit(true)` is called. Per the JDBC specification, that action is to commit the transaction, but there is varied compliance among drivers. By default, or if the property is set to `true`, these transactions are now committed.
- `-Dweblogic.datasource.endLocalTXOnXAConWithCommit=true` can be used to commit local transactions on XA connections when a connection pool is closed. By default, these transactions are rolled back.

A.14 JVM Settings

The Java virtual machine (JVM) is a virtual execution engine instance that executes the bytecodes in Java class files on a microprocessor. How you tune your JVM affects the performance of WebLogic Server and your applications.

When upgrading a WebLogic Server 10.3.*x* domain to a WebLogic Server 12.1.2 or greater domain, you may have to:

- manually set the location of the Java endorsed directory (`JRE_HOME/lib/endorsed`) or directories.

- manually increase the permgen space and maximum permgen space for the Oracle HotSpot JDK.

A.14.1 Setting the Location of the Java Endorsed Directory

In the following situations, you *do not* need to manually set the location of the Java endorsed directory or directories:

- you are using JDK7.
- you are using one of the JDKs that is installed with WebLogic Server 12.1.1.
- you are using WLS 12*c* domains and start scripts that were generated by domain creation via the WebLogic Server 12*c* Configuration Wizard, or your start scripts reference `commEnv.cmd/sh` as installed by the WebLogic Server installer, or both.

If none of these situations apply, and any one of the following situations apply, you must manually set the location of the Java endorsed directory in the command you use to start your Managed Servers:

- you are using Node Manager to start your Managed Servers, but you are not using a start script, that is `startScriptEnabled=false`. Note that as of WebLogic Server 12.1.1, the default value for `startScriptEnabled` is `true`.
- you are using custom start scripts, that is, start scripts that are not provided by Oracle.
- you are trying to create an empty domain using `java.weblogic.Server`.

In any of these cases, include the `java.endorsed.dirs` parameter in the Managed Server startup command.

```
startWeblogic.sh -Djava.endorsed.dirs=WL_HOME/endorsed
```

To specify multiple Java endorsed directories, separate each directory path with a colon (:).

> **Note:**
>
> In all of the options described in this section, you must replace *`WL_HOME`* with the full path to your WebLogic Server installation.

You can also specify this value when calling `startServer` by passing the values as `jvmArgs`, or when calling `nmstart` by passing them as properties, such as:

```
wls:/nm/mydomain> prps = makePropertiesObject("Arguments=-Djava.endorsed.dirs=/
WL_HOME/endorsed")

wls:/nm/mydomain> nmStart("AdminServer",props=prps)
```

If you are using Node Manager to start the Managed Server, you can include the `-Djava.endorsed.dirs=/WL_HOME/endorsed")` parameter in the ServerStartMBean's `arguments` attribute, either using WLST or the WebLogic Server Administration Console. If using the WebLogic Server Administration Console, enter this parameter in the **Arguments** field on the server's **Configuration > Server Start** tab. This attribute will be applied when you call `start(server_name 'Server')` from a WLST client that is

connected to the Administration Server or when you click on the **Start** button for the server in the WebLogic Server Administration Console.

A.14.2 Setting permgen space

If you receive an `OutOfMemory: PermGen Space` error when starting a Managed Server, you have to manually set the permgen space to at least 128M and increase the maximum permgen space to at least 256M.

> **Note:**
>
> In all of the options described here, you must replace *`WL_HOME`* with the full path to your WebLogic Server installation.

This can be done by specifying the following in the ServerStartMBean's `arguments` attribute, using either WLST or the WebLogic Server Administration Console. If using the WebLogic Server Administration Console, enter `-XX:PermSize=128m -XX:MaxPermSize=256m` in the **Arguments** field on the server's **Configuration > Server Start** tab.

> **Note:**
>
> If you plan to start the server via the WebLogic Server Administration Console, you must apply the permgen settings prior to starting the server from the WebLogic Server Administration Console. Otherwise the server may go into an unrecoverable state.

This attribute will be applied when you call `start(server_name 'Server')` from a WLST client that is connected to the Administration Server or when you click on the **Start** button for the server in the WebLogic Server Administration Console.

Another method you can use is to start the Managed Server via the command line and specify the correct settings, as shown here:

(UNIX) `startManagedWebLogic.sh` *`server_name`* `-XX:PermSize=128m -XX:MaxPermSize=256m`

(Windows) `startManagedWebLogic.cmd` *`server_name`* `-XX:PermSize=128m -XX:MaxPermSize=256m`

You can also specify these values when calling `startServer` by passing the values as `jvmArgs`, or when calling `nmstart` by passing them as properties, such as:

```
wls:/nm/mydomain> prps = makePropertiesObject("Arguments= -XX:PermSize=128m -
XX:MaxPermSize=256m")

wls:/nm/mydomain> nmStart("AdminServer",props=prps)
```

A.15 Node Manager startScriptEnabled Default Value

As of WebLogic Server 12.1.1, the default value for `startScriptEnabled` has been changed to `true`. In all previous releases, the default was `false`. If you do not want to use a start script with Node Manager, change this value to `false` after upgrading.

A.16 Enterprise Java Beans (EJBs)

Oracle Kodo has been deprecated as of WebLogic Server 10.3.1. As of WebLogic Server 12.1.1, EclipseLink is the default JPA provider, replacing Kodo. Applications that use Kodo as the persistence provider with WebLogic Server 12.1.2 must be updated. See Updating Applications to Overcome Conflicts in *Developing Enterprise JavaBeans for Oracle WebLogic Server*.
As of WebLogic Server 12.1.1, support for JPA 2.0 is built in. JPA 2.0 includes improvements and enhancements to domain modeling, object/relational mapping, `EntityManager` and `Query` interfaces, and the Java Persistence Query Language (JPQL), and more. See Using JPA 2.0 with TopLink in WebLogic Server in *Developing Enterprise JavaBeans for Oracle WebLogic Server*.

A.17 WebLogic Server 8.1 Web Services Stack Has Been Removed

In WebLogic Server 12.1.1 release, the WebLogic Server 8.1 Web services stack has been removed. Therefore, WebLogic Server 8.1 Web services applications will no longer work.

Oracle recommends that you upgrade such applications to the WebLogic JAX-RPC or JAX-WS stacks, as described in Upgrading an 8.1 WebLogic Web Service to 12.1.x.

A.18 Universal Description and Discover (UDDI) Registry Has Been Removed

In WebLogic Server 12.1.1 release, UDDI has been removed.

If you are still using UDDI and want to upgrade to WebLogic Server 12.1.1, Oracle recommends that you migrate to the Oracle Service Registry (OSR). OSR UDDI 3.0 compliant.

A.19 Certicom SSL Implementation Has Been Removed

In WebLogic Server release 12.1.1, the Certicom SSL implementation has been removed.

This change may require you to update system properties and debug switches as described in Command Line Properties for Enabling SSL Debugging and System Property Differences Between the JSSE and Certicom SSL Implementations in *Administering Security for Oracle WebLogic Server*.

A.20 Oracle Coherence Version

The WebLogic Server 12.1.1 installer includes Coherence 3.7.1. All servers in a cluster must use the same version of Coherence. Therefore, all cache servers in the cluster must be upgraded to Coherence 3.7.1.

A.21 Deprecated and Obsolete Web Application Features

See the list of Web application features that have been deprecated from, or are no longer supported in, Oracle WebLogic Server 12.1.1.

- Information about deprecated functionality in Oracle WebLogic Server 11*g* Release 1 can be found on My Oracle Support at `https://support.oracle.com/`.

  In the Search Knowledge Base field, enter document ID `888028.1`.
- Information about functionality that is deprecated in Oracle WebLogic Server 12.1.1 can be found on My Oracle Support at `https://support.oracle.com/`. Search for **Deprecated Features**.

A.22 Evaluation Database Changed From PointBase to Derby

As of WebLogic Server 10.3.3, the evaluation database available from the WebLogic Server installation program has been changed from PointBase to Apache Derby. If you select the **Evaluation Database** option on the Choose Products and Components screen, the Derby database is installed in the `WL_HOME\common\derby` directory. If you select a Typical installation, Derby is installed by default.

If you have a domain based on PointBase and you want to continue using PointBase after upgrading the domain to WebLogic Server 10.3.3 or later, you must obtain a PointBase license from `http://www.pointbase.com`. Note that the full WebLogic Server installer does not preserve the PointBase installation directory. As an alternative to using PointBase, you can migrate the domain database to Derby.

See Upgrading a Domain that Uses an Evaluation Database.

A.23 DataSource Profile Logging

To provide better usability and performance, Oracle WebLogic Server 10.3.6 and later uses a data source profile log to store events. See Monitoring WebLogic JDBC Resources in *Administering JDBC Data Sources for Oracle WebLogic Server*.

A.24 ONS Debugging

In Oracle WebLogic Server version 10.3.6 and later, the package names for UCP and ONS are no longer repackaged. For information about how to set UCP and ONS

debugging, see Setting Debugging for UCP/ONS in *Administering JDBC Data Sources for Oracle WebLogic Server*.

A.25 Oracle Type 4 JDBC drivers from DataDirect

As of Oracle WebLogic Server 10.3.6, Oracle Type 4 JDBC drivers from DataDirect are referred to as WebLogic-branded DataDirect drivers. Oracle has removed the documentation in *Oracle® Fusion Middleware Type 4 JDBC Drivers for Oracle WebLogic Server* and no longer provides detailed information about DataDirect drivers. Oracle continues to provide information about how WebLogic-branded drivers are configured and used in WebLogic Server environments at Using WebLogic-branded DataDirect Drivers in *Developing JDBC Applications for Oracle WebLogic Server*.
Oracle recommends reviewing DataDirect documentation for detailed information about driver behavior. See *Progress DataDirect for JDBC User's Guide Release 5.1* and *Progress DataDirect for JDBC Reference Release 5.1*at `http://www.datadirect.com/index.html`.

A.26 Default Message Mode Has Changed

As of WebLogic Server 12.1.1, the default messaging mode has been changed from multicast to unicast. When creating a new cluster, Oracle recommends the use of unicast for messaging within a cluster. For backward compatibility with previous versions of WebLogic Server, you must use multicast for communications between clusters.

A.27 Modifications to SSLMBean

As of WebLogic Server 10.3.5, the `SSLMBean` has been modified to support additional SSL configuration capabilities, including the ability to enable or disable the JSSE adapter. See the following topics:

- For a list of the differences in the way the JSSE SSL implementation handles the WebLogic system properties, see System Property Differences Between the JSSE and Certicom SSL Implementations in *Administering Security for Oracle WebLogic Server*.
- For information about SSL support in WebLogic Server, see SSL: An Introduction in *Understanding Security for Oracle WebLogic Server*.
- For information about JSEE, see *Java Secure Socket Extension (JSEE) Reference Guide* at `http://download.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html`.

A.28 New Web Services Features

Oracle WebLogic Server 10.3.3 added several new Web services features, such as support for Web services atomic transactions, enhanced support for clustered environments, the ability to attach Oracle WSM policies to WebLogic Web services using Fusion Middleware Control, and more.

The following new features have been added in WebLogic Server as of release 10.3.3:

- Support for Web services atomic transactions: WebLogic Web services enable interoperability with other external transaction processing systems, such as WebSphere, JBoss, Microsoft .NET.
- Enhanced support for Web services in a clustered environment
- Enhanced monitoring of Web services and clients
- Attachment of Oracle WSM policies to WebLogic Web services using Fusion Middleware Control
- EclipseLink DBWS support for declarative Web service solution for accessing relational databases
- Method-Level policy attachment behavior change: Before WebLogic Server 10.3.3, if a policy was attached, through the WebLogic Server Administration Console, to a method of one Web service, the policy was also attached to all methods of the same name for all Web services in that module. As of WebLogic Server 10.3.3, the policy is attached only to the method of the appropriate Web service.
- `policy:` prefix now removed from OWSM policy names
- Web services WSDL tab now removed: Before WebLogic Server 10.3.3, you could view the WSDL for the current Web service by selecting the **Configuration** > **WSDL** tab. The WSDL tab has been removed as of WebLogic Server 10.3.3.
- New development tools: Oracle JDeveloper and Oracle Enterprise Pack for Eclipse (OEPE)
- Integration with Oracle Enterprise Manager Fusion Middleware Control
- Support for Oracle WebLogic Services Manager (WSM) security policies
- Support for WS-SecureConversation 1.3 on JAX-WS and MTOM with WS-Security on JAX-WS

See *What's New in Oracle WebLogic Server*.

A.29 Introduction of JSSE

As of WebLogic Server 10.3.3, Java Secure Socket Extension (JSSE) was introduced as an SSL implementation. JSSE is the Java standard framework for SSL and TLS and includes both blocking-I/O and non-blocking-I/O APIs, and a reference implementation including several commonly-trusted CAs.

A.30 Performance Enhancements for Security Policy Deployment

As of release 10.3.3, WebLogic Server includes a deployment performance enhancement for Deployable Authorization providers and Role Mapping providers that are thread safe. WebLogic Server by default supports thread-safe parallel modification to security policy and roles during application and module deployment. For this reason, deployable Authorization and Role Mapping providers configured in the security realm should support parallel calls. The WebLogic deployable XACML Authorization Provider and the WebLogic Server XACML Role Mapping Provider meet this requirement. However, if your custom deployable Authorization or Role Mapping providers do not support parallel calls, you must disable the parallel security policy and role modification and instead enforce a synchronization mechanism that results in each

application and module being placed in a queue and deployed sequentially. You can turn on this synchronization enforcement mechanism from the WebLogic Server Administration Console or by using the `DeployableProviderSynchronizationEnabled` and `DeployableProviderSynchronizationTimeout` attributes of the RealmMBean.

See Enabling Synchronization in Security Policy and Role Modification at Deployment in *Administering Security for Oracle WebLogic Server* .

A.31 ActiveCache

As of WebLogic Server 10.3.3, applications deployed on WebLogic Server can easily use Coherence data caches, and seamlessly incorporate Coherence*Web for session management and TopLink Grid as an object-to-relational persistence framework. Collectively, these features are called ActiveCache. See About ActiveCache in *Deploying Applications with Oracle WebLogic Server ActiveCache*.

A.32 Class Caching

As of WebLogic Server 10.3.3, you can enable class caching in WebLogic Server. The advantages of using class caching are:

- Server startup time is reduced.
- The package level index reduces search time for all classes and resources.

Class caching is supported in development mode when starting the server using a `startWebLogic` script. Class caching is disabled by default and is not supported in production mode. The decrease in startup time varies among different JRE vendors. See Class Caching With the Policy Classloader in *Developing Applications for Oracle WebLogic Server*.

A.33 Deprecated JDBC Drivers

The WebLogic Type 4 JDBC driver for Oracle and Sybase JConnect 5.5 and 6.0 drivers are deprecated.

- WebLogic Type 4 JDBC driver for Oracle

  This driver was deprecated in WebLogic Server 10.3 and is now removed. Instead of using this deprecated driver, use the Oracle Thin Driver that is provided with WebLogic Server. For details about the Oracle Thin Driver, see JDBC Drivers Installed with WebLogic Server in *Administering JDBC Data Sources for Oracle WebLogic Server*.

- Sybase JConnect 5.5 and 6.0 drivers

  These drivers are removed from WebLogic Server as of release 10.3.3 due to an Oracle security policy regarding default installation of code samples. You can download the driver from Sybase or you can use the Oracle-branded JDBC driver for Sybase that is packaged with WebLogic Server.

A.34 Changes to weblogic.jms.extension API

As of WebLogic Server 10.3.3, the internal methods of the `weblogic.jms.extensions.WLMessage` interface are removed from *Java API Reference for Oracle WebLogic Server*. These methods include the following:

```
public void setSAFSequenceName(String safSequenceName);
public String getSAFSequenceName();
public void setSAFSeqNumber(long seqNumber);
public long getSAFSeqNumber();
```

Your applications should not use these internal methods. Internal methods may change or be removed in a future release without notice.

A.35 Persistent Store Updates

As of WebLogic Server 10.3.3, WebLogic file store behavior and tuning have changed for default file stores and custom file stores. For information about the Synchronous Write Policy, which determines the behavior of the write operation of the file store, see Guidelines for Configuring a Synchronous Write Policy in *Administering the WebLogic Persistent Store*.

A.36 Oracle Internet Directory and Oracle Virtual Directory Authentication Providers

As of WebLogic Server 10.3.2, two new LDAP authentication providers, the Oracle Internet Directory Authentication Provider and the Oracle Virtual Directory Authentication Provider, are added to WebLogic Server. These authentication providers can store users and groups in and read users and groups from the Oracle Internet Directory and Oracle Virtual Directory LDAP servers, respectively.

For information about configuring and using these new security providers, see Configuring LDAP Authentication Providers in *Administering Security for Oracle WebLogic Server*.

A.37 CapacityIncrement Attribute

As of WebLogic Server 10.3.1, the `CapacityIncrement` attribute of the `JDBCConnectionPoolMBean` is no longer configurable and is set to a value of 1.

A.38 Middleware Home Directory

As of WebLogic Server 10.3.1, Middleware Home replaces the notion of the BEA Home directory. The default path of this directory is `<drive:>Oracle/Middleware`. This change has the following impact on WebLogic Server:

- A new environment variable is introduced in several WebLogic scripts in 10.3.1 to represent the Middleware Home directory: `MW_HOME`. The directory to which this variable is set generally is the same as `BEA_HOME`, which is also still used in WebLogic Server scripts.
- By default, the WebLogic Server installation program selects `<drive:>Oracle/Middleware` as the root product installation directory. However, if a directory containing an existing WebLogic Server installation is detected, that directory is selected instead by default.
- The WebLogic Server 10.3.1 documentation now uses the term `Middleware Home`, instead of BEA Home. However, this revision is functionally only a change in terminology. This revision does not imply that any WebLogic software, custom

domains, or applications must be moved, or that any existing environment variables that represent those locations must be changed.

This change does not affect any existing WebLogic Server installations, custom domains, applications, or scripts on your computer. You can continue to use the `BEA_HOME` environment variable as before.

A.39 Resource Registration Name

As of WebLogic Server 10.3.1, the behavior of the resource registration name for XA data source configurations has changed. In previous releases, the Java Transaction API (JTA) registration name was simply the name of the data source. Starting WebLogic Server 10.3.1, the registration name is a combination of data source name and domain.

See Registering an XAResource to Participate in Transactions in *Developing JTA Applications for Oracle WebLogic Server*.

A.40 Servlet Path Mapping

As of version 2.3 of the Java Servlet Specification, two additional characters, `/` and `*`, can be used to define mappings.

- A servlet path string that contains only the `/` (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 illustrate the change in behavior, consider the 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 information returned is non-null, replace all occurrences of the `/` (slash) servlet mapping string with `/*`.

If you define a servlet using both the `@WebServlet` annotation and the `<servlet>` element of the `web.xml` deployment descriptor file, you must make sure that the servlet name you specify is the same in both locations. If the names do not match, then an exception is generated that prevents the application from starting. This requirement was not enforced in versions of WebLogic Server 12c before 12.2.1.0.

You can download the Java Servlet Specification from the following location:

`http://www.oracle.com/technetwork/java/javaee/servlet/index.html`