Oracle® Fusion Middleware

What's New in Oracle WebLogic Server

12*c* (12.1.2)

E28063-05

March 2015

This document describes new features for all components of Oracle WebLogic Server in the current release.

Oracle Fusion Middleware What's New in Oracle WebLogic Server, 12*c* (12.1.2)

E28063-05

Contents

Preface

This preface describes the document accessibility features and conventions used in this guide—*What's New in 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 and Roadmap

This chapter describes the contents and audience for this guide—*What's New in Oracle WebLogic Server*.

This chapter includes the following sections:

- Section 1.1, "Document Scope and Audience"
- Section 1.2, "Guide to This Document"

1.1 Document Scope and Audience

This document is a resource for system administrators and operators responsible for monitoring and managing a WebLogic Server installation. It is relevant to all phases of a software project, from development through test and production phases.

1.2 Guide to This Document

This document is organized as follows:

- This chapter, Chapter 1, "Introduction and Roadmap," provides a roadmap for this guide and describes the audience for this guide.
- Chapter 2, "What's New in Oracle WebLogic Server," describes new features for all components of Oracle WebLogic Server in the current release.

2 What's New in Oracle WebLogic Server

This chapter describes the following new and changed functionality in this WebLogic Server release:

- Install, Patch, and Upgrade
- New Configuration Features
- Updated Maven Support
- OSGi Bundles for WebLogic Server Applications
- WebSockets
- Oracle TopLink 12.1.2
- Server Templates
- Dynamic Clusters
- Cluster Targeted JMS Servers
- Managed Coherence Servers
- WebLogic JMS Resource Adapter for Foreign Application Server Integration
- Oracle Database 12c Integration
- Optimizations for Exalogic Elastic Cloud Software
- Fusion Middleware Control
- Additional Component-Level Features and Changes
- Deprecated Functionality (WebLogic Server 12c)
- Standards Support, Supported Configurations, and WebLogic Server Compatibility

2.1 Install, Patch, and Upgrade

The following sections describe the new install, patch, and upgrade features in this release of WebLogic Server:

2.1.1 New Installation Features

The WebLogic Server installation program includes the changes described in the following sections:

- JDK Is No Longer Included
- Available Installers

- New Graphical User Interface
- Samples Installation and Configuration

The WebLogic Server installation process is described in *Installing and Configuring Oracle WebLogic Server and Coherence*.

2.1.1.1 JDK Is No Longer Included

The WebLogic Server installation program no longer includes a JDK. Prior to installing this release of WebLogic Server, you must complete the following steps:

1. Install the required JDK.

   For information about the JDKs supported on your platform, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.

2. Set the `JAVA_HOME` environment variable to the path of the JDK.

3. Update the `PATH` environment variable to include:

   – The `JAVA_HOME` environment variable

   – The `*JAVA_HOME*/bin` directory, where `*JAVA_HOME*` represents the path to the JDK.

For more information about installing the JDK, see "Starting the Installation Program" in *Installing and Configuring Oracle WebLogic Server and Coherence*.

2.1.1.2 Available Installers

The available WebLogic Server installers are generic and can be used on any supported platform. Platform-specific WebLogic Server installers are no longer provided.

The following self-extracting installer JARs are available:

- `wls_121200.jar`—Installs WebLogic Server and Coherence
- `fmw_infra_121200.jar`—Installs WebLogic Server, Coherence, and infrastructure components for Fusion Middleware product platforms.

All installations must be initiated from the command line using the `java` command, for example:

```
java -jar wls_121200.jar
```

WebLogic Server 12.1.2 continues to be available in ZIP format from the Oracle Technology Network as in prior releases. The WebLogic Server ZIP Distribution is intended for development use only and is supported on Windows, Linux and Mac OS X systems. It contains the necessary artifacts to develop and test applications on WebLogic Server. For information about its use, see the README file included in the distribution. Note that installations based on the WebLogic Zip Distribution cannot be patched. If you require a patch, you must install WebLogic using a full distribution, and then use OPatch to apply the required patch.

> **Note:** Upgrade installers are not provided to upgrade WebLogic Server 12.*x* releases to a later WebLogic Server 12.*x* release. For information on how to upgrade a WebLogic Server 12.*x* release, see *Upgrading Oracle WebLogic Server*.

2.1.1.3 New Graphical User Interface

The following changes have been made to the WebLogic Server installer graphical user interface (GUI):

- The installer GUI has been updated to use the Oracle Universal Installer (OUI) look-and-feel, consistent with other Oracle product installers.
- Context-sensitive help is available for each installer screen. To access help for the current screen, click **Help**.
- The Choose Product Installation Directories and Confirm Product Installation Directories screens are no longer included. Instead, you are prompted to specify the Oracle home directory. Product installation directories for WebLogic Server and Coherence have been standardized to *ORACLE_HOME*\wlserver and *ORACLE_HOME*\coherence.
- Installation types are changed to the following:
  - **WebLogic Server—**Installs WebLogic Server and Oracle Coherence, but excludes the WebLogic Server code examples.
  - **Coherence Installation—**Installs Oracle Coherence and WebLogic Server components needed to administer the life cycle of managed Coherence servers.
  - **Complete with Examples—**Installs WebLogic Server, Oracle Coherence, and the WebLogic Server code examples.

    Note that the WebLogic samples domain is configured as a separate, post-installation task.

  For a complete list of components installed with each of the preceding options, see *Installing and Configuring Oracle WebLogic Server and Coherence*.
- A Prerequisite Checks screen has been added. At this point, the installer automatically verifies that installation prerequisites have been met, and notifies you of any issues.
- A Software Updates screen has been added in which you can specify how you want to receive software updates.
- The available custom installation options have changed. For more information, see "Understanding the WebLogic Server and Coherence Distribution" in *Installing and Configuring Oracle WebLogic Server and Coherence*.
- The Install Windows Service screen, which provided the option to install Node Manager as a service on Windows systems, is no longer included.
- The Choose Shortcut Location screen is no longer available.
- An option to create a response file for silent installation has been added to the Installation Summary screen.
- An option to automatically launch the Fusion Middleware Quick Start Configuration Wizard to configure the samples upon exiting the installer has been added to the Installation Complete screen. You can select this option to configure the WebLogic code examples and sample domains.
- The **Run Quickstart** option has been removed from the Installation Complete screen.

2.1.1.4 Samples Installation and Configuration

If you choose to install the WebLogic code examples component, which is available by selecting **Complete with Examples** in this release of the installer, the installation of the code examples and the configuration of the WebLogic samples domains are optionally performed as a post-installation procedure by the Quick Start Configuration Wizard (see Section 2.2.4, "Quick Start Configuration Wizard").

Note that as of this release of WebLogic Server, no default administrator password is set for the WebLogic samples domain. You set the administrator password when you configure the samples domain, which can be done either immediately after installation or at any time later time using the samples domain template.

> **Note:** If you run the Configuration Wizard (or WLST) separately from installation to configure the WebLogic samples domain (that is, you do not launch the Quick Start Configuration Wizard from the installer), you must also manually set the administrator username and password in the `*EXAMPLES_HOME*\wl_server\examples\src\examples.properties` file. For more information, see "Setting the Administrator Username and Password" in the Examples Guide, available from the browser when you launch the samples Web application.

2.1.2 New Patching Features

This release of WebLogic Server introduces the use of Oracle OPatch, which is a Java-based utility used to patch an existing Oracle Fusion Middleware installation, including WebLogic Server. OPatch runs on all supported operating systems.

Smart Update is no longer included with WebLogic Server.

2.1.3 New Upgrade Features

The following changes have been made to the WebLogic Server upgrade process.

2.1.3.1 Reconfiguration Wizard

Reconfiguration templates are provided for Fusion Middleware products to make it easier to upgrade WebLogic Server and Fusion Middleware product installations from a previous release.

The Reconfiguration Wizard automatically applies the appropriate reconfiguration templates for WebLogic Server and all Fusion Middleware products that are detected in the WebLogic Server installation. These templates update the domain version to the current release, and may also take advantage of new features that are available in WebLogic Server.

For more information, see "Reconfiguring WebLogic Domains" in *Upgrading Oracle WebLogic Server* and "Types of Templates" in *Domain Template Reference*.

2.1.3.2 Miscellaneous Changes

The following changes impact the WebLogic Server upgrade process:

- The Domain Upgrade Wizard is no longer included with WebLogic Server. Use the Reconfiguration Wizard instead to upgrade your domains.

- Domains must be backed up manually prior to starting the upgrade process. Automatic backup, which was a function of the Domain Upgrade Wizard, is no longer available.
- Domains on remote machines can be upgraded using the WLST `writeTemplate` command in WLST online mode. For more information, see "writeTemplate" in *WLST Command Reference for WebLogic Server*.

2.2 New Configuration Features

This section describes the configuration changes that have been implemented in this release of WebLogic Server.

- Oracle Coherence Configuration
- Configuration Wizard
- Template Location
- Quick Start Configuration Wizard
- Managed Server Cloning
- Migratable Target Configuration
- Node Manager Configuration
- Domain Reconfiguration
- Updating Domains on Remote Machines

See also Section 2.8, "Dynamic Clusters," for information about dynamic server configuration, which is new in this release of WebLogic Server.

2.2.1 Oracle Coherence Configuration

Oracle Coherence configuration support has been added to WebLogic domain configuration, using either the Configuration Wizard or WLST.

Using the Configuration Wizard, you can change the **Coherence Listen Port** for the default Coherence cluster.

Using existing WLST commands, you can:

- Create Coherence clusters
- Enable Coherence on the Administration Server
- Assign WebLogic clusters and Managed Servers to Coherence clusters
- Configure the parameters for a Coherence cluster, such as multicast listen address/port, clustering mode, transport, and Security Framework setting
- Configure Coherence well-known addresses
- Configure the logging parameters for a Coherence cluster

For more information, see:

- *Creating WebLogic Domains Using the Configuration Wizard*
- "Setting Up a WebLogic Server Domain Topology for Coherence" in *Administering Oracle Coherence*
- Section 2.10, "Managed Coherence Servers"

2.2.2 Configuration Wizard

The following changes have been made to the Configuration Wizard in this release:

- A roadmap panel has been added to the GUI to indicate the sequence of screens that will be displayed when configuring a domain.
- Support has been added for configuring OHS and ODI system components in a WebLogic domain.
- A **Template Categories** drop-down list has been added to the **Templates** screen to restrict the list of displayed templates to the selected category.
- Node Manager configuration options have been changed and expanded. This includes selecting the Node Manager type and Node Manager home, as well as specifying the username and password for Node Manager.
- You can use the Configuration Wizard to update a domain without adding additional templates to the domain.
- A **Server Groups** column has been added to the Managed Servers screen. If available, you can optionally assign a Managed Server to a defined user-expandable server group.
- A **Clone** function has been added to the Managed Servers screen, enabling you to create a new server by cloning an existing server.
- The **Cluster messaging mode**, **Multicast address**, and **Multicast port** columns have been removed from the Clusters screen.
- The functionality to create Windows **Start** menu entries for a domain has been removed.

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

2.2.3 Template Location

WebLogic Server template locations have changed. All templates that were previously stored in *WL_HOME*/common/templates/applications and *WL_HOME*/common/templates/domains have been moved to *WL_HOME*/common/templates/wls.

If you have any WLST scripts that incorporate these templates, you must modify them to use the new path.

2.2.4 Quick Start Configuration Wizard

This release of WebLogic Server includes a new utility called the Quick Start Configuration Wizard, which is provided for configuring the WebLogic samples domain. If you select **Complete Installation** to install the WebLogic samples, and then select Configure WebLogic Samples in the Installation Complete screen, the Quick Start Configuration Wizard is automatically launched. You then use this utility to configure the WebLogic samples domain, including setting the administrator username and password.

For more information, see "Quick Start Configuration Wizard" in *Creating WebLogic Domains Using the Configuration Wizard*.

2.2.5 Managed Server Cloning

Using either the Configuration Wizard or WLST, you can create a new Managed Server by cloning an existing Managed Server. The cloned server inherits the attributes of the original server.

2.2.6 Migratable Target Configuration

WebLogic Server now automatically designates servers that are part of a cluster as migratable targets. As a best practice, all JMS services, including JMS servers, path services, custom stores, and SAF agents, should be targeted to migratable targets instead of individual server instances or clusters. By default, WebLogic Server will always target JMS services to a migratable target.

2.2.7 Node Manager Configuration

In this release, the Java version of Node Manager has a simplified, out-of-the-box configuration. For each WebLogic domain you create, a domain-specific Node Manager configuration is created by default, complete with security credentials, properties file, domain registration, and start scripts, including pre-configured scripts to install and uninstall Node Manager as a Windows service.

For more information, see "Default Node Manager Configuration" in *Administering Node Manager for Oracle WebLogic Server*.

2.2.8 Domain Reconfiguration

The Domain Upgrade Wizard has been removed from WebLogic Server, and has been replaced with a Reconfiguration Wizard. The Reconfiguration Wizard updates an existing WebLogic domain to the current WebLogic Server release to take advantage of new features.

For more information, see "Reconfiguring WebLogic Domains" in *Upgrading Oracle WebLogic Server*.

2.2.9 Updating Domains on Remote Machines

The WLST `writeTemplate` command has been modified to work in online mode, which enables updates to domains that reside on remote machines using WLST instead of the pack/unpack commands.

2.3 Updated Maven Support

This release of WebLogic Server contains the following new features for building and managing Oracle WebLogic Server applications using Apache Maven:

- An embedded distribution of Maven 3.0.4 that can be used as a Maven repository. You can also download your own copy of Maven from the Maven Web Site at `http://maven.apache.org/`.
- A Maven Synchronization plug-in, and supporting POMs, which allows you to populate a local or shared Maven repository with Oracle product artifacts from an Oracle home.
- An updated version of the WebLogic Server Maven plug-in. The new plug-in adds support for three new goals:
  - `ws-clientgen` — Generates client Web service artifacts from a WSDL.

  - `ws-wsdlc` — Generates a set of artifacts and a partial Java implementation of the Web service from a WSDL.
  - `ws-jwsc` — Builds a JAX-WS Web service.

  The new plug-in also enables execution of certain plug-in goals referencing WebLogic Server product jars in a Maven repository, without requiring a local Oracle home.

- Four archetypes for use in building Oracle WebLogic Server applications:
  - Basic WebApp archetype
  - WebApp with EJB archetype
  - Basic MDB archetype
  - Basic WebServices archetype

For more detail about the updated Maven support, see the following documentation:

- *Developing Applications Using Continuous Integration*
- "Using the WebLogic Development Maven Plug-in" in *Developing Applications for Oracle WebLogic Server*

2.4 OSGi Bundles for WebLogic Server Applications

In this release of WebLogic Server, developers who want to use OSGi in their applications can easily share OSGi facilities, such as the OSGi service registry, class loaders, and other OSGi services. WebLogic Server allows you to add a list of OSGi frameworks to the server configuration. After the OSGi framework has been booted, a bundle object for the framework is placed into the local server JNDI tree. Applications can then get this bundle from JNDI and thereafter use that as their entry point into the OSGi system. Applications can also deploy their own OSGi bundles.

OSGi is described in "Developing OSGi Bundles for WebLogic Server Applications" in *Developing Applications for Oracle WebLogic Server*.

2.5 WebSockets

This release of WebLogic Server adds support for the WebSocket Protocol (RFC 6455), which provides two-way, full-duplex communication over a single TCP connection between clients and servers, where each side can send data independently from the other. The WebSockets communication model occurs in real-time and promotes user interaction.

In WebLogic Server, you can use the WebSocket Protocol implementation and accompanying programming API to develop and deploy applications that communicate bidirectionally with clients.

For more information, including code examples, see "Using WebSockets in WebLogic Server" in *Developing Applications for Oracle WebLogic Server*.

2.6 Oracle TopLink 12.1.2

WebLogic Server continues to include the Oracle TopLink mapping and persistence framework for use in building and deploying Java EE applications. Oracle TopLink supports key standards such as the Java Persistence API (JPA) 2.0 and Java Architecture for XML Binding (JAXB) 2.0 standards, as well as extensions to these standards. See "Overview of Oracle TopLink" in *Understanding Oracle TopLink* for more information about TopLink.

This release of WebLogic Server includes Oracle TopLink 12.1.2, which contains the following new and changed features that are available for use with WebLogic Server:

- JSON bindings, for converting objects directly to and from JavaScript Object Notation (JSON). This can be useful when creating RESTful services, using JSON messages with Java API for RESTful Web Services (JAX-RS) services.
- RESTful persistence, where Java Persistence API (JPA) entities can be exposed through standards-based RESTful services such as JAX-RS, using either JSON or XML media.
- TopLink Live Data queries ("live queries"), where live queries track changes to the backing data store.
- NoSQL database support, allowing objects to be mapped to non-relational (NoSQL) data sources.
- Client isolation, where multiple application tenants may share database tables and schemas. This allows applications to manage entities for multiple tenants in the same application.

For more information about new and changed features in TopLink, see "New and Changed Features for 12c (12.1.2)" in *Solutions Guide for Oracle TopLink*.

2.7 Server Templates

This release of WebLogic Server introduces server templates. Server templates allow you to define common, non-default attributes that you can apply to different server instances. Because common attributes are contained in the server template, you only need to change them in one place and they take effect in all of the server instances that use the server template. The primary use case for server templates is for creating dynamic clusters.

For more information, see "Server Templates" in *Understanding Domain Configuration for Oracle WebLogic Server*.

2.8 Dynamic Clusters

This release of WebLogic Server introduces dynamic clusters. A dynamic cluster is any cluster that contains one or more dynamic servers. Dynamic clusters are based on a single shared server template. You use the server template to specify the configuration of the servers in your dynamic cluster so that each server does not need to be manually configured when expanding the cluster. Dynamic clusters allow you to easily scale up the number of server instances in your domain. When configuring your dynamic cluster, you can specify the number of server instances you anticipate needing at peak load. WebLogic Server will create the specified number of server instances and apply the calculated attribute values to each one. When you need additional server capacity, you can then start a server instance without having to first manually configure it and add it to the cluster.

For more information, see "Creating Dynamic Clusters" in *Administering Clusters for Oracle WebLogic Server*.

2.9 Cluster Targeted JMS Servers

Using clustered targeted JMS Servers simplifies the configuration process by eliminating the need to individually configure many JMS resource artifacts for every server in a cluster by targeting JMS Servers and persistent stores directly to a cluster. It also enables dynamic scaling of JMS resources in a dynamic cluster and the dynamic members of a mixed cluster.

For more information, see "Simplified JMS Cluster Configuration" in *Administering JMS Resources for Oracle WebLogic Server*.

2.10 Managed Coherence Servers

Although Coherence can continue to be used independent of WebLogic Server, and independent of the WebLogic Server Administration infrastructure, this release of WebLogic Server supports new capabilities for managing Coherence cluster members. Managed Coherence servers are Managed Servers that are configured using the WebLogic Server Administration infrastructure as Coherence cluster members. Management functions supported include managing the Coherence life cycle (starting and stopping servers and Coherence cluster members), managing Coherence configurations, deploying Coherence applications, and monitoring Coherence cluster members. Coherence applications and the managed Coherence servers are not mandated by the Java EE specification but are specific to WebLogic Server and Coherence. Managed Coherence servers can be used with WebLogic Server Java EE applications, or independent of WebLogic Server Java EE applications.

Coherence applications use a specific directory structure for deployment to the managed Coherence server. You can deploy a Coherence application as a collection of files that use this directory structure, known as exploded directory format, or as an archived file called a Grid ARchive (GAR) with a `.gar` extension. The GAR module includes the artifacts that make up a Coherence application and may also contain a cache configuration file (`coherence-cache-config.xml`) and a Portable Object Format (POF) serialization configuration file (`pof-config.xml`).

Note: WebLogic Server is not a required platform for Oracle Coherence. You can continue use Coherence independently of WebLogic Server, including configuring and managing Coherence servers and clusters, data management, and for building and deploying Coherence applications.

For more information, see *Developing Oracle Coherence Applications for Oracle WebLogic Server*.

2.11 WebLogic JMS Resource Adapter for Foreign Application Server Integration

This release of WebLogic Server includes a Java EE Connector Architecture compliant resource adapter to provide a high performance integration of a WebLogic JMS client with foreign application servers. The JMS resource adapter is an implementation of the Java EE Connector Architecture outbound and inbound contract for JMS, and includes the following features:

- JNDI mapping to reference JMS connection factories and destinations
- MDB integration (including dynamic adjustment to changing message load)

- JMS connection pooling
- Lazy resolution of JMS operations (including start order independence, tolerance of dynamic management such as starts and stops of JMS providers, and connection retries in case of provider failure).
- Cluster-capable XA support for WebLogic JMS that transparently integrates with non-WebLogic Transaction Managers and correctly recovers from all typical failure conditions.
- Support for asynchronous message processing that ensures all active members of a distributed destination are always serviced — no trapped messages. In addition, asynchronous messaging provides advanced publish/subscribe options, such as a single logical durable subscription partitioned across a distributed topic.
- Support for WebLogic JMS extensions that allow you to cast adapter wrapped objects to WebLogic JMS interfaces.
- Support for the WebLogic security model.
- Advanced poison message handling. The JMS resource adapter can be configured to automatically redirect messages that have been redelivered multiple times to a designated error destination.
- Simplified destination references that allow an application to reference an arbitrary destination with no need to configure a mapping for each and every specific destination.

This release of the JMS resource adapter is supported for deployment on GlassFish version 3.1 and higher.

For more information, see *Administering the JMS Resource Adapter for Oracle WebLogic Server*.

2.12 Oracle Database 12c Integration

This release of WebLogic Server adds support for Oracle Database 12c, which enables the following features:

- JDBC 4.1 Support for JDK 7
- Application Continuity Support
- Database Resident Connection Pooling Support
- Container Database with Pluggable Databases
- Global Database Services Support
- Automatic ONS Listeners

For other database support features included in this release, see also Section 2.15.2, "JDBC Data Sources."

2.12.1 JDBC 4.1 Support for JDK 7

WebLogic Server supports the JDBC 4.1 Specification when the environment is using JDK 7 and the JDBC driver is JDBC 4.1 compliant. See "JDBC 4.1 Support for JDK 7" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.12.2 Application Continuity Support

Application Continuity is an Oracle database feature that provides a general purpose, application-independent infrastructure that enables recovery of work and masks many system, communication, and hardware failures. See "Application Continuity" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.12.3 Database Resident Connection Pooling Support

Database Resident Connection Pooling (DCRP) is an Oracle database server feature that provides the ability to share connections among multiple connection pools that can span across mid-tier systems. See "Database Resident Connection Pooling" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.12.4 Container Database with Pluggable Databases

Container Database (CDB) is an Oracle Database feature that minimizes the overhead of having many of databases by consolidating them into a single database with multiple Pluggable Databases (PDB) in a single CDB. See "Container Database with Pluggable Databases" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.12.5 Global Database Services Support

Global Data Services (GDS) is an Oracle database server feature that provides automated load balancing, fault tolerance and resource utilization in a distributed database environment. See "Global Database Services" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.12.6 Automatic ONS Listeners

If you are using an Oracle 12c database with WebLogic Server release12.1.2 and higher, you are no longer required to provide the ONS Listener list as part of an Active GridLink datasource configuration. The ONS list is automatically provided from the database to the driver. See "Enabling FAN Events" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.13 Optimizations for Exalogic Elastic Cloud Software

This release of WebLogic Server includes additional optimizations for Exalogic Elastic Cloud software, including the following:

- Advanced Encryption Standard (AES) support
- JMS optimizations, including message compression

For more information, see "Oracle WebLogic Server Optimizations for Exalogic Elastic Cloud Software" in *Licensing Information*.

2.14 Fusion Middleware Control

WebLogic Server can now be managed through Fusion Middleware Control. The following subsets of functionality are available:

- Server startup and shutdown
- Clustering

- WebLogic Server services, such as database connectivity (JDBC) and messaging (JMS)
- Application deployment
- Monitoring server and application performance

For more information, see *Administering Oracle WebLogic Server with Fusion Middleware Control*.

2.15 Additional Component-Level Features and Changes

This section describes additional features and changes in this release for the following WebLogic Server components:

- Deployment
- JDBC Data Sources
- JTA
- Logging
- pack and unpack Utilities
- Resource Adapters
- Security
- SNMP
- Stand-alone Clients
- Web Applications, Servlets, and JSPs
- WebLogic API Jar
- WebLogic Diagnostic Framework
- WebLogic RMI
- WebLogic Tuxedo Connector
- Web Services
- WLST

2.15.1 Deployment

This section describes the following new deployment features in this release of WebLogic Server:

- Staging Deployment Plans
- Introspection Plug-in for Oracle Virtual Assembly Builder
- JMX API for Deployment Operations

2.15.1.1 Staging Deployment Plans

An application's deployment plan can be staged independently of the application archive, allowing you to stage a deployment plan when the application is not staged. If you do not specify a staging mode, the deployment plan uses the value specified for application staging as the default.

For information about staging deployment plans using weblogic.Deployer, see "Staging Deployment Plans" in *Deploying Applications to Oracle WebLogic Server*.

For information about staging deployment plans using the WebLogic Deployment API, see "Deployment Plan Staging Modes" in *Deploying Applications with the WebLogic Deployment API*.

2.15.1.2 Introspection Plug-in for Oracle Virtual Assembly Builder

In this release of WebLogic Server, the introspection plug-in for Oracle Virtual Assembly Builder includes the following Node Manager support enhancements:

- A new option that allows you to specify the name of a Node Manager user.
- The ability to run Node Manager as a non-root user.
- The ability to use Node Manager configured with SSL.

The introspection plug-in also has additional features, such as the ability to set the rehydration user to an identity other than `oracle`. For more information, see "Using the Plug-in for Oracle Virtual Assembly Builder" in *Administering Server Environments for Oracle WebLogic Server*.

2.15.1.3 JMX API for Deployment Operations

The Java Management Extensions (JMX) API for deployment operations supports all of the common functionality available in the Java EE Deployment API specification (JSR-88). You can use the JMX API as an alternative to JSR-88 to perform deployment tasks on specified target servers.

Four runtime MBeans support the JMX API for deployment operations. These new MBeans for deployment are similar conceptually to JSR-88 and are located in the Domain Runtime MBean Server.

For more information, see "The JMX API for Deployment Operations" in *Deploying Applications with the WebLogic Deployment API*.

2.15.2 JDBC Data Sources

This release includes the following new and changed features:

- Oracle Data Base Testing Using PINGDATABASE
- Pinned-to-thread Support for GridLink Data Sources
- Using GridLink Data Sources without Fan Notification
- Set Identity and/or Client Identifier as WebLogic User
- Connection Testing Improvements
- JavaDB Support
- Edition-based Redefinition
- Maintenance Timer Improvements
- JDBC Related Changes in the Weblogic Server Distribution
- Datasource Configuration Changes
- Committing Local Transactions

2.15.2.1 Oracle Data Base Testing Using PINGDATABASE

You can improve the connection testing performance of your data source by setting the `Test Table Name` attributed of the connection pool to `SQL PINGDATABASE`. See "Global Database Services" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.15.2.2 Pinned-to-thread Support for GridLink Data Sources

This release extends the `Pinned-to-Thread` feature to a GridLink data source. Using `Pinned-to-Thread` can increase performance by minimizing the time it takes for an application to reserve a database connection from a data source and eliminating contention between threads for a database connection. See "Using Pinned-To-Thread Property to Increase Performance" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.15.2.3 Using GridLink Data Sources without Fan Notification

In this release, you can configure and use an Active GridLink data source without enabling Fast Application Notification (FAN). See "Using GridLink Data Sources without FAN Notification" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.15.2.4 Set Identity and/or Client Identifier as WebLogic User

To simplify the configuration of large numbers of users, WebLogic Server provides a datasource configuration option that sets the user for Identify based pooling and/or the client identifier directly using the WebLogic Server user, rather than using a credential mapper. See "Understanding Data Source Security" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.15.2.5 Connection Testing Improvements

This release provides two new connection pool tuning attributes, `CountOfTestFailuresTillFlush` and `CountOfRefreshFailuresTillDisable`, that can improve performance in some situations by minimizing the delay caused by testing dead connections. See "Connection Testing When Database Connections are Created" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.15.2.6 JavaDB Support

This document refers to the Derby database. The references regarding use of the Derby are equally applicable to Java DB. Java DB is Oracle's distribution of the open source Apache Derby Java database. It is distributed as part of the Java Development Kit. See Java DB at `http://www.oracle.com/technetwork/java/javadb/overview/index.html`.

2.15.2.7 Edition-based Redefinition

Edition-based redefinition enables you to upgrade the database component of an application while it is in use. See "Enabling Edition-Based Redefinition" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.15.2.8 Maintenance Timer Improvements

WebLogic JDBC provides several new timer properties that allow you to tune the performance or your JDBC data sources:

- `weblogic.jdbc.gravitationShrinkFrequencySeconds`
- `weblogic.jdbc.harvestingFrequencySeconds`
- `weblogic.jdbc.securityCacheTimeoutSeconds`

See "Tuning Maintenance Timers" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.15.2.9 JDBC Related Changes in the Weblogic Server Distribution

Many JDBC files that were located in the `wlserver/server/lib` directory have moved to the following locations:

The following files have moved to `ORACLE_HOME/oracle_common/modules/datadirect`:

64sqljdbc.dll
DDJDBC64Auth05.dll
DDJDBCAuth05.dll
DDJDBCx64Auth05.dll
fmwgenerictoken.jar
instjdbc.sql
sqljdbc.dll
wldb2.jar
wlinformix.jar
wlsybase.jar
wlsqlserver.jar
x64sqljdbc.dll

The following files have moved to `ORACLE_HOME/oracle_common/modules/oracle.jdbc_11.2.0`:

ojdbc5.jar
ojdbc6.jar
ojdbc6dms.jar
ojdbc5_g.jar
ojdbc6_g.jar

The following files have moved to `ORACLE_HOME/oracle_common/modules/oracle.nlsrtl_11.2.0`:

orai18n-mapping.jar
orai18n.jar

The following files have moved to `ORACLE_HOME/oracle_common/modules/oracle.ons_12.1.1`:

ons.jar

The following files have moved to `ORACLE_HOME/oracle_common/modules`:

oracle.ucp_11.2.0.jar

2.15.2.10 Datasource Configuration Changes

Identity based pooling was previously documented as not supported with `pinned-to-thread`. It is now flagged as a configuration error that causes a data source to fail to deploy. See "Using Pinned-To-Thread Property to Increase Performance" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.15.2.11 Committing 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 will be 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 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.

2.15.3 JTA

This release includes the following new and changed JTA features:

- You can configure JTA transaction options at the cluster level, see "Configuring JTA" in *Developing JTA Applications for Oracle WebLogic Server*.
- You can configure cluster-wide transaction recovery of distributed transactions across all the interposed transaction managers of a cluster, see "Cluster-wide Recovery" in *Developing JTA Applications for Oracle WebLogic Server*.
- Support for transactions that span transaction manager systems, see "Transactions that Span Transaction Manager Systems" in *Developing JTA Applications for Oracle WebLogic Server*.

2.15.4 Logging

In this release of WebLogic Server, when you start a server instance in **production mode**, by default the server rotates its server log file whenever the file grows to 5000 kilobytes in size. In addition, WebLogic Server now sets a threshold size limit of 2,097,152 kilobytes before it forces a hard rotation to prevent excessive log file growth.

2.15.5 pack and unpack Utilities

The following modifications have been made to the `pack` and `unpack` utilities:

- Persistent file stores are no longer included in a packed domain, even if they are located in the domain. If the file store is located within the domain, an empty file store directory will be created on the target system.
- Deployment plans located in external directories will be included in the template that is created by `pack`. These will be copied by `unpack` to the following standard location in the domain:

  ```
  domain_home/config/deployments/deployment_name/plan
  ```

- Node Manager configuration is preserved by `pack` and `unpack` for both managed and non-managed modes if the Node Manager type is either `DomainNodeManager` or `CustomLocationNodeManager`. For more information, see "Node Manager Configuration" in *Creating Templates and Domains Using the Pack and Unpack Commands*.

2.15.6 Resource Adapters

In resource adapters configured with multiple connection pools, this release of WebLogic Server adds the ability to isolate failed connection pools from healthy ones during deployment. This enables you to locate, diagnose, and fix failed connection pools, and then dynamically update the adapter deployment, without redeploying the resource adapter.

The ability to detect outbound connection pool failures is available through the health monitoring feature, which is extended to resource adapters. You can access the health state of a resource adapter deployment using WLST or the WebLogic Server Administration Console. For more information, see "Creating and Configuring Resource Adapters" and "Packaging and Deploying Resource Adapters" in *Developing Resource Adapters for Oracle WebLogic Server*.

2.15.7 Security

This section describes the following security changes in WebLogic Server 12.1.2:

- Oracle OPSS Keystore Service Support
- Location Change of Demonstration Identity Keystore
- Behavior Change in response.sendError Method

2.15.7.1 Oracle OPSS Keystore Service Support

The Oracle Platform Security Services (OPSS) Keystore Service provides an alternate mechanism to manage keys and certificates for message security. The OPSS Keystore Service makes using certificates and keys easier by providing central management and storage of keys and certificates for all servers in a domain. You use the OPSS Keystore Service to create and maintain keystores of type KSS.

You can use the OPSS Keystore Service with WebLogic Server only if you have installed the Oracle JRF template on the WebLogic Server system and used this template to create the domain. The OPSS Keystore Service is available only with the JRF template and is not available with the default WebLogic Server configuration.

For more information, see "Configuring Oracle OPSS Keystore Service" in *Administering Security for Oracle WebLogic Server*.

2.15.7.2 Location Change of Demonstration Identity Keystore

When you create a WebLogic domain in this release of WebLogic Server, the demonstration identity keystore is placed in the `DOMAIN_HOME/security` directory. (In previous releases, this keystore was placed in the `WL_HOME/server/lib` directory.)

> **Note:** The demonstration digital certificates, private keys, and trusted CA certificates should be used in a development environment only.

For more information about using the demonstration digital certificates, private keys, and trusted CA certificates in the WebLogic Server environment, see "Configuring Identity and Trust" in *Administering Security for Oracle WebLogic Server*.

2.15.7.3 Behavior Change in response.sendError Method

If you use the `response.sendError(int, String)` method, where `String` includes a script, the `String` is now xss-encoded; for example, `&lt;`*`scriptname`*`&gt;`. This behavior change prevents the script contained in the `String` from being executed.

2.15.8 SNMP

The WebLogic Server SNMP agent includes a new name-value pair, `trapConfigName`, in the variable bindings of each monitor notification. This name-value pair can be used to more easily correlate a generated TRAP notification to the SNMP monitoring MBean that defines the criteria that, when reached, causes the TRAP notification to be generated.

The `trapConfigName` name-value pair affects the TRAP notifications generated due to the monitoring criteria defined in the following SNMP monitoring MBeans:

- SNMPCounterMonitorMBean
- SNMPGaugeMonitorMBean
- SNMPStringMonitorMBean

The `trapConfigName` name-value pair is also used in the variable bindings for attribute change notifications. This enables you to correlate an attribute change notification to the SNMPAttributeChangeMBean that defines the MBean attributes that, when changed in any way, cause the notification to be generated.

For more information, see "Monitor Notifications" in *Monitoring Oracle WebLogic Server with SNMP*.

2.15.9 Stand-alone Clients

This release includes the following new features for Stand-alone Clients:

- WebLogic Store-and-Forward Service support for the WebLogic Thin T3 Client.
- RMI over HTTPS (HTTP Tunneling over SSL) support for the WebLogic Thin T3 Client.
- Network Classloading support for the WebLogic Thin T3 Client.

See "Developing a WebLogic Thin T3 Client" in *Developing Stand-alone Clients for Oracle WebLogic Server*.

2.15.10 Web Applications, Servlets, and JSPs

This section describes the following changes and new Web application, servlet, and JSP features in this release of WebLogic Server:

- Coherence Applications
- MaxRequestParamterCount Attribute Name Change
- HTTP Request Parameters

2.15.10.1 Coherence Applications

This release of WebLogic Server standardizes the way Coherence applications are packaged, deployed, and managed within a WebLogic Server domain. Coherence applications and managed Coherence servers are not mandated by the Java EE specification and are specific to WebLogic Server.

For more information, see *Developing Oracle Coherence Applications for Oracle WebLogic Server*.

2.15.10.2 MaxRequestParamterCount Attribute Name Change

The spelling of the `WebServer` MBean `MaxRequestParamterCount` attribute that was introduced in some patches to earlier WebLogic Server releases has been changed to `MaxRequestParameterCount`. This does not impact existing WLST scripts that set this parameter using the incorrect spelling. You should, however, use the corrected spelling when writing new WLST scripts.

This parameter can be set via WLST online from the Servers/*server_name*/WebServer/*server_name* directory.

2.15.10.3 HTTP Request Parameters

In previous WebLogic Server releases, HTTP request parameters were stored in an ArrayMap by default. In WebLogic Server 12.1.2, HTTP request parameters are stored in a TreeMap by default.

If you want to continue using an ArrayMap for storing HTTP request parameters, you can include the following new startup parameter in the server startup command:

```
-Dweblogic.utils.http.requestparams.useArrayMap=true
```

2.15.11 WebLogic API Jar

In conjunction with the Java API Reference for Oracle WebLogic Server, Oracle recommends using the `wls-api.jar` file to develop and compile Java EE applications for your WebLogic Server environment. The `wls-api.jar` file is located in the `wlserver/server/lib` directory of your WebLogic Server distribution.

For more information, see "Java API Reference and the wls-api.jar File" in *Developing Applications for Oracle WebLogic Server*.

2.15.12 WebLogic Diagnostic Framework

The following sections describe the new and changed features of the WebLogic Diagnostic Framework (WLDF) made in this release of WebLogic Server:

- Built-in System Diagnostic Modules
- Targeting Multiple Diagnostic Modules to a Server or Cluster
- Dynamic Activation/Deactivation of Diagnostic System Modules
- New Diagnostics WLST Commands
- Using Java Flight Recorder with Oracle HotSpot
- WLDF Schema Changes

2.15.12.1 Built-in System Diagnostic Modules

WLDF adds a set of built-in system diagnostic modules, which are provided as a simple and easy-to-use mechanism for performing basic health and performance monitoring of a WebLogic Server instance. When configured in a WebLogic Server instance, the built-in diagnostic modules are particularly useful for providing a low-overhead, historical record of server performance. As server workload changes over time, or the performance characteristics change as a result of updates made to the server's configuration, you can examine the data collected by the built-ins to obtain details about performance changes.

In WebLogic domains configured to run in production mode, a built-in diagnostic module is enabled by default in each server instance. (In domains configured to run in development mode, built-ins are disabled by default.) However, a built-in diagnostic module can be enabled or disabled for a server instance easily and dynamically, using either the WebLogic Server Administration Console or WLST.

To simplify the process of creating a diagnostic system module, the built-in diagnostic modules can be "cloned" and customized as appropriate. For example, you can add or remove individual metrics that are collected by the built-in diagnostic module. The built-in diagnostic modules also include a set of watches and notifications that are configured but not activated. You can choose to activate one or more of those watches and notifications, or customize them as desired, in the diagnostic module you create.

For more information, see "Using the Built-in System Diagnostic Modules" in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

2.15.12.2 Targeting Multiple Diagnostic Modules to a Server or Cluster

In previous releases of WebLogic Server, you could target only one diagnostic system module to a server or cluster at any one time; you could not have two files in the `DOMAIN_HOME/config/diagnostics` directory whose active target is the same server or cluster. However, this restriction is lifted in this release of WebLogic Server. You can now target multiple diagnostic system modules to a server or cluster instance.

For more information, see "Understanding WLDF Configuration" in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

2.15.12.3 Dynamic Activation/Deactivation of Diagnostic System Modules

In this release of WebLogic Server, WLDF includes the following new features to enhance control over the operative state of diagnostic system modules without restarting the targeted server or cluster instances or making a change to the domain configuration:

- The ability to dynamically activate and deactivate diagnostic system modules that are defined in the domain configuration.
- The ability to create and activate a diagnostic system module on-the-fly that is not defined in the domain configuration

These features are supported by the following:

- New WLST commands to activate and deactivate diagnostic system modules that are part of the WLDF configuration, and also to create and destroy diagnostic system modules on-the-fly. See Section 2.15.12.4, "New Diagnostics WLST Commands".
- A new run-time API for the dynamic control of diagnostic system modules without making a change to the domain configuration:
  - `weblogic.management.runtime.WLDFControlRuntimeMBean`
  - `weblogic.management.runtime.WLDFSystemResourceControlRuntimeMBean`

For more information, see "Understanding WLDF Configuration" in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

2.15.12.4 New Diagnostics WLST Commands

WLDF adds the following new WLST commands in this release

- `listSystemResourceControls()` — Lists all available diagnostic system modules.

- `enableSystemResource()` — Activates a diagnostic system module.
- `disableSystemResource()` — Deactivates a diagnostic system module.
- `createSystemResourceControl()` — Creates a diagnostics system module on-the-fly using a specified descriptor file.
- `destroySystemResourceControl()` — Destroys a diagnostics system module previously created on-the-fly.
- `dumpDiagnosticData()` — Dumps the diagnostics data from a harvester to a local file.

For more information about these commands, see "Diagnostics Commands" in *WLST Command Reference for WebLogic Server*.

2.15.12.5 Using Java Flight Recorder with Oracle HotSpot

By default, Java Flight Recorder is disabled in WebLogic Server as of this release. To enable Java Flight Recorder, you must specify the following JVM options in the WebLogic Server instance in which the JVM runs:

```
-XX:+UnlockCommercialFeatures -XX:+FlightRecorder
```

For more information about how to specify these JVM options, see "Using Java Flight Recorder with Oracle HotSpot" in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

2.15.12.6 WLDF Schema Changes

In the WLDF schema, the `THREADNAME` column has changed from varcat(128) to varcat(250) in the wls_events table, and the `WLDFMODULE` column has been added to the wls_hvst table. If you use a JDBC-based for event and harvester data, you must manually update the WLDF schema. See "WLDF Schema Upgrade" in *Upgrading Oracle WebLogic Server* for more information.

2.15.13 WebLogic RMI

This release of WebLogic Server introduces the following changes and new features in its support for WebLogic RMI:

- New WebLogic RMI annotations that provide remote access to plain java objects. See "Using WebLogic RMI Annotations" in *Developing RMI Applications for Oracle WebLogic Server*.
- A new connection attribute, `WLContext.CONNECT_TIMEOUT`, to define the length of time a client waits for connections to the server to be bootstrapped or re-established. `WLContext.REQUEST_TIMEOUT` is deprecated. See "Using a Connect Timeout" in *Developing RMI Applications for Oracle WebLogic Server*.
- A new connection attribute, `WLContext.RESPONSE_READ_TIMEOUT`, to define the length of time that a client waits to receive a response from a server. `WLContext.RMI_TIMEOUT` is deprecated. See "Using a Read Timeout" in *Developing RMI Applications for Oracle WebLogic Server*.

2.15.14 WebLogic Tuxedo Connector

This release includes the following new and changed WebLogic Tuxedo Connector features:

- Support for `readonly` transactions, see "Read-only, One-Phase Commit Optimizations" in *Developing JTA Applications for Oracle WebLogic Server*.
- Support for tightly coupled transaction between WebLogic and Tuxedo, see "Transactions that Span Transaction Manager Systems" in *Developing JTA Applications for Oracle WebLogic Server*.
- Nested view support, see "Translating Nested Views" in *Developing Oracle WebLogic Tuxedo Connector Applications for Oracle WebLogic Server*.

2.15.15 Web Services

This section describes new and changed WebLogic Web Services features in this release of WebLogic Server, including:

- OWSM Policy Support for RESTful Web Services and Clients
- SOAP Over JMS Transport
- Fast Infoset
- Enhanced Web Services Test Client
- Annotation Support for Attaching OWSM Policies to Java EE Web Service Clients
- SHA-256 Secure Hash Algorithm
- Standalone Web Service Client JARs
- Web Services Maven Goals

2.15.15.1 OWSM Policy Support for RESTful Web Services and Clients

Secure RESTful Web services using Oracle Web Services Manager (OWSM) policies. For more information, see "Securing RESTful Web Services and Clients Using OWSM Policies" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

2.15.15.2 SOAP Over JMS Transport

SOAP over Java Messaging Service (JMS) transport is supported as a connection protocol for WebLogic Web services using Java API for XML Web Services (JAX-WS). This feature supports the new *W3C SOAP over Java Message Service 1.0* standard (February 2012), available at: `http://www.w3.org/TR/soapjms/`

For more information, see "Using SOAP Over JMS Transport" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

2.15.15.3 Fast Infoset

Fast Infoset, a compressed binary encoding format that provides a more efficient serialization than the text-based XML format, is supported in this release. Fast Infoset optimizes both document size and processing performance. See "Optimizing XML Transmission Using Fast Infoset" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

2.15.15.4 Enhanced Web Services Test Client

The Web Services Test Client for testing Web services has been enhanced to enable you to test advanced features of Web services. See "Using the Web Services Test Client" in *Administering Web Services*.

2.15.15.5 Annotation Support for Attaching OWSM Policies to Java EE Web Service Clients

Annotations are now supported for attaching Oracle Web Services Manager (OWSM) security policies to Web service clients. See "Attaching Policies to Java EE Web Services and Clients Using Annotations" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

2.15.15.6 SHA-256 Secure Hash Algorithm

In addition to SHA-1, the WebLogic Server Web service security policies support the much stronger SHA-2 (SHA-256) secure hash algorithms for hashing digital signatures. For more information, see "Use the SHA-256 Secure Hash Algorithm" in *Securing WebLogic Web Services for Oracle WebLogic Server*.

2.15.15.7 Standalone Web Service Client JARs

The installation includes a set of standalone Web service client JAR files that support the following functionality:

- Basic JAX-WS and OWSM security policy support, as described in "Invoking a Web Service from a Standalone Java SE Client" in *Developing JAX-WS Web Services for Oracle WebLogic Server*
- Basic JAX-RS and OWSM security policy support, as described in "Invoking a RESTful Web Service from a Standalone Client" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*
- Basic JAX-RPC support, as described in "Using a Standalone Client JAR File When Invoking a Web Service" in *Developing JAX-RPC Web Services for Oracle WebLogic Server*

2.15.15.8 Web Services Maven Goals

WebLogic Server now provides support for the following Web services Maven goals:

- `ws-clientgen` — Generates client Web service artifacts from a WSDL.
- `ws-wsdlc` — Generates a set of artifacts and a partial Java implementation of the Web service from a WSDL.
- `ws-jwsc` — Builds a JAX-WS Web service.

See "Using the WebLogic Development Maven Plug-in" in *Developing Applications for Oracle WebLogic Server* for complete documentation.

2.15.16 WLST

This section describes changes to existing WLST commands and new WLST commands.

The following existing WLST commands have changed:

- `writeTemplate` has been modified to work in online mode, making it easier to create or update remote Managed Server domains.
- the `inheritance` argument has been added to the `ls` command.
- support for system components, such as Oracle HTTP Server, has been added to the `state`, `shutdown`, and `start` commands. This applies only to installations that include Fusion Middleware system components co-installed with WebLogic Server.

- the default value of the `block` argument for the `shutdown` command has been changed to `true`.
- the `pluginProps` argument has been added to the `nmkill` and `nmStart` commands. This argument is used only to plug-in handling system components such as Oracle HTTP Server.
- the `jvmargs` argument has been added to the `startNodeManager` command.

The following WLST commands have been added:

- `readDomainForUpgrade` opens an existing domain for reconfiguration.
- `getNodeManagerHome` returns the path of the Node Manager home directory.
- `getNodeManagerType` returns the Node Manger type.
- `clone` clones an existing Managed Server to create a new Managed Server.
- `listSystemResourceControls()`lists all available diagnostic system modules.
- `enableSystemResource()` activates a diagnostic system module.
- `disableSystemResource()` deactivates a diagnostic system module.
- `createSystemResourceControl()` creates a diagnostics system module on-the-fly using a specified descriptor file.
- `destroySystemResourceControl()` destroys a diagnostics system module previously created on-the-fly.
- `dumpDiagnosticData()` dumps the diagnostics data from a harvester to a local file.

The following WLST commands have been added to support WebLogic Server installations that include Fusion Middleware products:

- `listServerGroups` retrieves a map of the config-groups.xml server groups that are user-expandable.
- `setServerGroups` sets the server groups for the specified server.
- `getServerGroups` retrieves a list of the server groups of which the specified server is a member.
- `getStartupGroup` retrieves the server startup group for a server.
- `addStartupGroup` adds a new server startup group.
- `getDatabaseDefaults` retrieves schema information from a database.
- `setStartupGroup` sets the startup group for a server.
- `setSharedSecretStoreWithPassword` sets the shared secret store and password for a shared database store.
- `softRestart` restarts a system component server instance (such as an Oracle HTTP Server instance).
- `nmSoftRestart` restarts the specified system component server instance using Node Manager.

2.16 Deprecated Functionality (WebLogic Server 12*c*)

Information about deprecated functionality for WebLogic Server 12*c* Release 2 can be found on My Oracle Support at `https://support.oracle.com/`. Search for "Deprecated Features" in the **Search Knowledge Base** field.

2.16.1 weblogic.management.username and weblogic.management.password

As of WebLogic Server 12.1.1, the boot username and password system properties `weblogic.management.username` and `weblogic.management.password` have been deprecated and will be removed in a future release, and you will no longer be able to specify the username and password in the command for starting WebLogic Server in production mode.

As an alternative, Oracle recommends that you use the `boot.properties` file to specify the boot username and password for WebLogic Server. For more information about the `boot.properties` file, see "Boot Identity Files" in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

For information about other methods you can use to provide user credentials, see "Provide User Credentials to Start and Stop Servers" in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

2.16.2 Maven 11x Plug-In Deprecated

The `weblogic-maven-plugin` plug-in delivered in WebLogic Server 11*g* Release 1 is deprecated in this release. Oracle recommends that you instead use the WLS 12.1.2 Maven plug-in. See "Using the WebLogic Development Maven Plug-in" in *Developing Applications for Oracle WebLogic Server* for complete documentation.

2.16.3 JSP Tags for XML Processing

As of WebLogic Server 12.1.2, XSLT JSP tags and the WebLogic XSLT JSP Tag Library have been deprecated and will be removed in a future release. You can use JAXP to transform XML data. For more information, see "Transforming XML Documents" in *Developing XML Applications for Oracle WebLogic Server*.

2.16.4 ActiveCache

ActiveCache is deprecated as of WebLogic Server 12.1.2.

2.17 Standards Support, Supported Configurations, and WebLogic Server Compatibility

The following sections describe WebLogic Server standards support, supported system configuration, and WebLogic Server compatibility:

- Standards Support
- Supported Configurations
- WebLogic Server Compatibility
- Database Interoperability

2.17.1 Standards Support

This release of WebLogic Server supports the following standards and versions.

- Java Standards
- Web Services Standards
- Other Standards

2.17.1.1 Java Standards

Table 2–1 lists currently supported Java standards.

Table 2–1 Java Standards Support

Standard	Version
Contexts and Dependency Injection for Java EE	1.0
Dependency Injection for Java EE	1.0
Expression Language (EL)	2.2, 2.1, 2.0 Only JSP 2.0 and greater supports Expression Language 2.x.
JAAS	1.0 Full
JASPIC	1.0
Java API for XML-Based Web Services (JAX-WS)	2.2, 2.1, 2.0
Java API for RESTful Web Services (JAX-RS)	1.1
Java Authorization Contract for Containers (JACC)	1.4
Java EE	6.0
Java EE Application Deployment	1.2
Java EE Bean Validation	1.1
Jave EE Common Annotations	1.0
Java EE Connector Architecture	1.6
Java EE EJB	3.1, 3.0, 2.1, 2.0, and 1.1
Java EE Enterprise Web Services	1.3, 1.2, 1.1
Jave EE Interceptors	1.1
Java EE JDBC	4.0, 3.0
Java EE JMS	1.1, 1.0.2b
Java EE JNDI	1.2
Java EE JSF	2.1, 2.0, 1.2, 1.1
Java EE JSP	2.2, 2.1, 2.0, 1.2, and 1.1 JSP 1.2. and 1.1 include Expression Language (EL), but do not support EL 2.x or greater.
Java EE Managed Beans	1.0
Java EE Servlet	3.0, 2.5, 2.4, 2.3, and 2.2
Java RMI	1.0
JavaMail	1.4
JAX-B	2.2, 2.1, 2.0
JAX-P	1.3, 1.2, 1.1
JAX-R	1.0
JAX-RPC	1.1
JCE	1.4
JDKs	7.0. (6.0 and 7.0 clients only.)
JMX	1.4

Table 2–1 (Cont.) Java Standards Support

Standard	Version
JPA	2.0, 1.0
JSR 77: Java EE Management	1.1
JSTL	1.2
Managed Beans	1.0
OTS/JTA	OTS 1.2 and JTA 1.1
RMI/IIOP	1.0
SOAP Attachments for Java (SAAJ)	1.3, 1.2
Streaming API for XML (StAX)	1.0
Web Services Metadata for the Java Platform	2.0, 1.1

2.17.1.2 Web Services Standards

For the current list of standards supported for WebLogic Web services, see "Features and Standards Supported by WebLogic Web Services" in *Understanding WebLogic Web Services for Oracle WebLogic Server*.

2.17.1.3 Other Standards

Table 2–2 lists other standards that are supported in this release of WebLogic Server.

Table 2–2 Other Standards

Standard	Version
SSL	v3
X.509	v3
LDAP	v3
TLS	v1.1, v1.2
HTTP	1.1
SNMP	SNMPv1, SNMPv2, SNMPv3
xTensible Access Control Markup Language (XACML)	2.0
Partial implementation of Core and Hierarchical Role Based Access Control (RABC) Profile of XACML	2.0
Internet Protocol (IP)	Versions: ▪ v6 ▪ v4

For more information about IPv6 support for all Fusion Middleware products, see the Oracle Fusion Middleware Supported System Configurations page on Oracle Technology Network.

2.17.2 Supported Configurations

For the most current information on supported configurations, see the Oracle Fusion Middleware Supported System Configurations page on Oracle Technology Network.

2.17.3 WebLogic Server Compatibility

For the most current information on compatibility between current version of WebLogic Server and previous releases, see "WebLogic Server Compatibility" in *Understanding Oracle WebLogic Server*.

2.17.4 Database Interoperability

The certification matrices and My Oracle Support Certifications define the following terms to differentiate between types of database support:

- Application Data Access
- Database Dependent Features

2.17.4.1 Application Data Access

Application Data Access refers to those applications that use the database for data access only and do not take advantage of WebLogic Server features that are Database dependant. WebLogic Server support of databases used for application data access only are less restrictive than for database dependent features.

WebLogic Server provides support for application data access to databases using JDBC drivers that meet the following requirements:

- The driver must be thread safe.
- The driver must implement standard JDBC transactional calls, such as `setAutoCommit()` and `setTransactionIsolation()`, when used in transactional aware environments.

Note the following restrictions:

- JDBC drivers that do not implement serializable or remote interfaces cannot pass objects to an RMI client application.
- Simultaneous use of automatic database connection failover and load balancing and global transactions (XA) with a highly-available (HA) DBMS architecture is supported with Oracle DB RAC only, and only for the Oracle DB RAC versions indicated on the **System** worksheet. These HA capabilities are only supported by Active GridLink for RAC and Multi Data Sources with RAC. These HA capabilities are not supported on other Oracle DB RAC versions or with other HA DBMS technologies on other non-Oracle DB products. Multi Data Sources are supported on other Oracle DB versions, and with non-Oracle DB technologies, but not with simultaneous use of automatic failover and load balancing and global transactions.
- Application data access to databases meeting the restrictions articulated above is supported on other Oracle DB versions, in addition to those documented in the certification matrix.
- WebLogic Type 4 JDBC drivers also support the following databases. For these databases, WebLogic Server supports application data access only, and does not support WebLogic Server database dependent features:
  - DB2 V9.1 for z/OS, DB2 V10 for z/O
  - Informix 11.0, Informix 11.5, Informix 11.7

2.17.4.2 Database Dependent Features

When WebLogic Server features use a database for internal data storage, database support is more restrictive than for application data access. The following WebLogic Server features require internal data storage:

- Container Managed Persistence (CMP)
- Rowsets
- JMS/JDBC Persistence and use of a WebLogic JDBC Store
- JDBC Session Persistence
- RDBMS Security Providers
- Database Leasing (for singleton services and server migration)
- JTA Logging Last Resource optimization
- JDBC TLog