

Oracle® Fusion Middleware

Understanding Oracle WebLogic Server



14c (14.1.1.0.0)

F18327-08

November 2022

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Copyright © 2007, 2022, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Documentation Accessibility	viii
Diversity and Inclusion	viii
Conventions	viii

1 Introduction

Product Overview	1-1
Programming Models	1-2
High Availability	1-3
Diagnostic Framework	1-4
Security	1-4
Client Options	1-5
Integration with Oracle WebLogic Suite	1-5
Integration with Other Systems	1-5
Integration with Web Servers	1-6
Running Oracle WebLogic Server on Docker	1-6
WebLogic Deploy Tooling	1-6
WebLogic Image Tool	1-6
WebLogic Server Kubernetes Operator	1-7
WebLogic Monitoring Exporter	1-7
WebLogic Logging Exporter	1-7
WebLogic Server API Examples and Sample Application	1-7
Upgrade	1-8

2 System Administration

Overview of WebLogic Server System Administration	2-1
Choosing the Appropriate Technology for Your Administrative Tasks	2-1
Summary of System Administration Tools and APIs	2-5
Roadmap for Administering the WebLogic Server System	2-8

3 Overview of Administration Consoles

Using the WebLogic Server Administration Console	3-1
About the WebLogic Server Administration Console	3-1
Starting the WebLogic Server Administration Console	3-2
Enabling the WebLogic Server Administration Console	3-3
Elements of the WebLogic Server Administration Console	3-3
Using the Change Center	3-6

4 WebLogic Server Domains

Understanding Domains	4-1
Organizing Domains	4-1
Contents of a Domain	4-3
Administration Server	4-3
Managed Servers and Managed Server Clusters	4-4
Managed Coherence Servers and Coherence Clusters	4-4
Resources and Services	4-4
Roadmap for Understanding WebLogic Server Domains	4-5

5 WebLogic Server Clustering

Overview of WebLogic Server Clusters	5-1
Relationship Between Clusters and Domains	5-1
Relationship Between Coherence and WebLogic Server Clusters	5-2
Benefits of Clustering	5-2
Key Capabilities of Clusters	5-2
Objects That Can Be Clustered	5-3
About Dynamic Clusters	5-3
Roadmap for Clustering in WebLogic Server	5-4

6 Developing Applications in WebLogic Server

WebLogic Server and the Java EE Platform	6-1
Overview of Java EE Applications and Modules	6-2
Roadmap for Developing Applications in WebLogic Server	6-2

7 Deploying Applications in WebLogic Server

Overview of the Deployment Process	7-1
Java EE Deployment Implementation	7-1
Fast Track Deployment Guide	7-3

Java EE Deployment	7-3
Auto-Deployment	7-3
Deploying Multiple Applications	7-4
System Administrator Tools	7-4
JSP/HTML Deployment	7-4
Coherence Deployment	7-5
Roadmap for Deploying Applications in WebLogic Server	7-5

8 WebLogic Server Data Sources

Understanding JDBC Data Sources	8-1
Understanding Generic Data Sources	8-1
Understanding Active GridLink Data Sources	8-2
Understanding JDBC Multi Data Sources	8-2
Understanding Universal Connection Pool Data Sources	8-2
Roadmap for WebLogic Server Data Sources	8-3

9 WebLogic Server Messaging

Overview of JMS and WebLogic Server	9-1
Java Message Service	9-1
Roadmap for WebLogic Server Messaging	9-2

10 Understanding WebLogic Server Security

JSR 375 Security API Support in WebLogic Server	10-1
Overview of the WebLogic Server Security Service	10-2
WebLogic Server Security Service Architecture	10-3
WebLogic Security Framework	10-3
Single Sign-on with the WebLogic Server Security Framework	10-5
SAML Token Profile Support in WebLogic Web Services	10-5
The Security Service Provider Interfaces (SSPIs)	10-5
WebLogic Security Providers	10-6
Managing WebLogic Server Security	10-6
Security for Coherence	10-7
Roadmap for Securing WebLogic Server	10-7

11 WebLogic Server Web Services

Anatomy of a Web Service	11-1
Web Service Standards	11-2

12 Enterprise JavaBeans (EJBs)

Understanding EJBs	12-1
EJB Documentation in WebLogic Server	12-1
Additional EJB Information	12-1
Session EJBs Implement Business Logic	12-2
Message-Driven Beans Implement Loosely Coupled Business Logic	12-3
EJB Anatomy and Environment	12-3
EJB Components	12-3
The EJB Container	12-4
Embeddable EJB Container	12-5
EJB Metadata Annotations	12-5
Optional EJB Deployment Descriptors	12-5
EJBs Clients and Communications	12-6
Accessing EJBs	12-6
EJB Communications	12-6
Securing EJBs	12-7
Roadmap for EJBs in WebLogic Server	12-7

13 Monitoring, Diagnosing, and Troubleshooting

WebLogic Diagnostics Framework	13-1
Logging Services	13-2
SNMP Support	13-3
Custom JMX Applications	13-3
Java EE Management APIs	13-3
Roadmap for Monitoring, Diagnosing, and Troubleshooting in WebLogic Server	13-3

14 Sample Applications and Code Examples

Installing and Running the Examples	14-1
Installing the WebLogic Server Code Examples	14-1
Starting the WebLogic Server Samples Domain	14-1
Running the WebLogic Server Code Examples	14-2
Conventions	14-2
Java EE 8 Examples	14-3
Java EE 7 Examples	14-3
Additional API Examples	14-4
Avitek Medical Records	14-5

15 WebLogic Server Compatibility

Java EE 8 Compatibility	15-1
Generated Classes Compatibility	15-1
Compatibility Within a Domain	15-1
About WebLogic Server Version Numbers	15-1
WebLogic Version Compatibility	15-2
Hardware, Operating System, and JVM Platform Compatibility	15-3
Node Manager Compatibility	15-3
Persistent Data Compatibility	15-4
API Compatibility	15-4
Protocol Compatibility	15-4

Preface

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <https://www.oracle.com/corporate/accessibility/>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <https://support.oracle.com/portal/> or visit <https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab> if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

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.
<i>italic</i>	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

Oracle WebLogic Server is the industry's best application server for building and deploying enterprise Java EE applications with support for new features for lowering cost of operations, improving performance, enhancing scalability, and supporting the Oracle Applications portfolio.

The following sections provide an overview of Oracle WebLogic Server features and describe how you can use them to create enterprise-ready solutions:

Product Overview

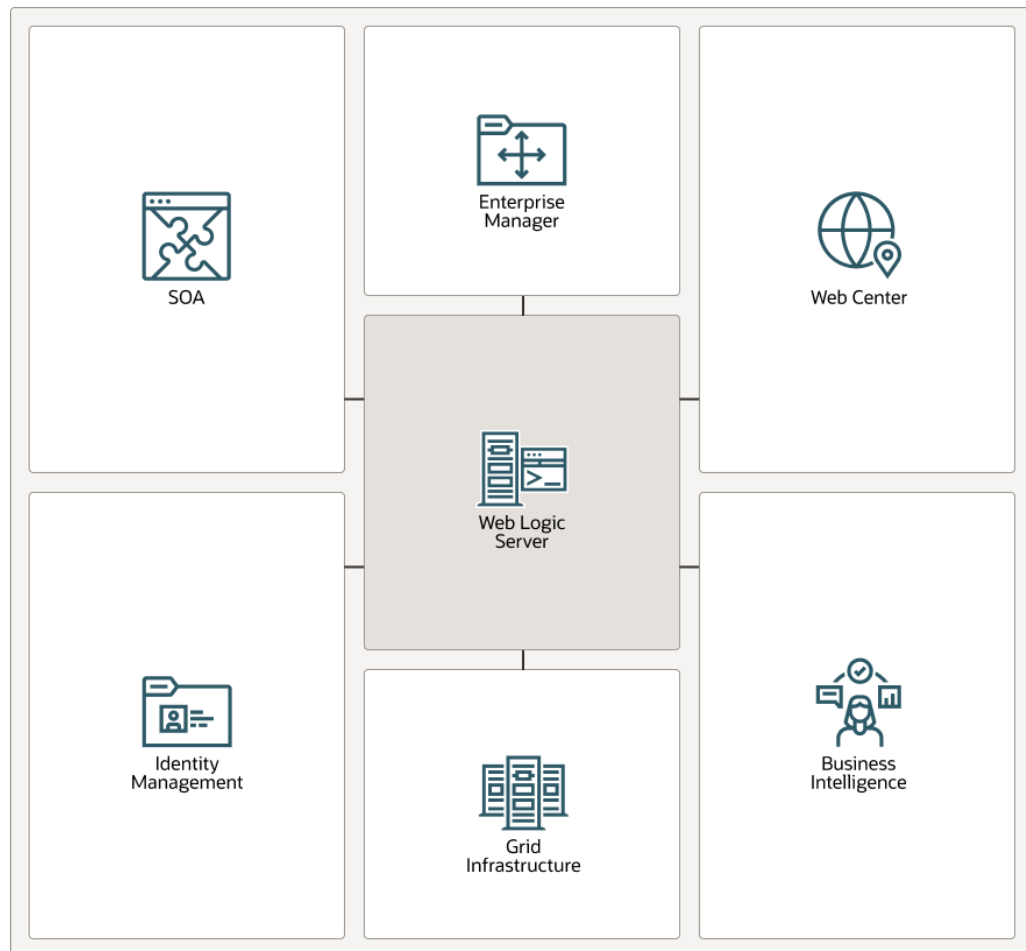
Oracle WebLogic Server provides a modern development platform for building applications, a runtime platform for high performance and availability, and rich management tooling for efficient and low cost operations.

The WebLogic Server complete implementation of the Java Platform, Enterprise Edition 8 (Java EE 8) specification provides a standard set of APIs for creating distributed Java applications that can access a wide variety of services, such as databases, messaging services, and connections to external enterprise systems. End-user clients access these applications using web browser clients or Java clients. See [Programming Models](#).

In addition to the Java EE implementation, WebLogic Server enables enterprises to deploy mission-critical applications in a robust, secure, highly available, and scalable environment. These features allow enterprises to configure clusters of WebLogic Server instances to distribute load, and provide extra capacity in case of hardware or other failures. New diagnostic tools allow system administrators to monitor and tune the performance of deployed applications and the WebLogic Server environment itself. You can also configure WebLogic Server to monitor and tune application throughput automatically without human intervention. Extensive security features protect access to services, keep enterprise data secure, and prevent malicious attacks.

[Figure 1-1](#) shows how WebLogic Server fits into the overall Oracle Fusion Middleware stack.

Figure 1-1 Oracle Fusion Middleware Overview



Programming Models

Oracle WebLogic Server provides complete support for the Java Platform, Enterprise Edition 8 (Java EE 8) platform, which reduces the complexity of enterprise application development by providing a development model, API, and runtime environment that allow developers to concentrate on functionality. The Java EE 8 specification is available at <https://www.oracle.com/java/technologies/javae8/javae8technologies.html#javae8>. For information about Java EE programming model support in WebLogic Server, see the following programming guides:

- Web Applications provide the basic Java EE mechanism for deployment of dynamic web pages based on the Java EE standards of servlets and JavaServer Pages (JSP). Web applications are also used to serve static web content such as HTML pages and image files.
- Web Services provide a shared set of functions that are available to other systems on a network and can be used as a component of distributed web-based applications.
- XML capabilities include data exchange, and a means to store content independent of its presentation, and more.

- Java Messaging Service (JMS) enables applications to communicate with one another through the exchange of messages. A message is a request, report, and/or event that contains information needed to coordinate communication between different applications.
- Java Database Connectivity (JDBC) provides pooled access to DBMS resources.
- Resource Adapters provide connectivity to Enterprise Information Systems (EISes).
- Enterprise JavaBeans (EJB) provide Java objects to encapsulate data and business logic.
- Remote Method Invocation (RMI) is the Java standard for distributed object computing, allowing applications to invoke methods on a remote object locally.
- Security APIs allow you to integrate authentication and authorization into your Java EE applications. You can also use the Security Provider APIs to create your own custom security providers.
- WebLogic Tuxedo Connectivity (WTC) provides interoperability between WebLogic Server applications and Tuxedo services. WTC allows WebLogic Server clients to invoke Tuxedo services and Tuxedo clients to invoke EJBs in response to a service request.
- Coherence provides distributed caching and data grid capabilities for WebLogic Server applications.
- Overview of WebLogic Server Application Development describes developer tools and best practices for coding WebLogic Server applications.

High Availability

Oracle WebLogic Server provides several features and tools to support the deployment of highly-available applications that can be automatically scaled to meet demand in a reliable and fault-tolerant manner. The high availability features provided in WebLogic Server include the following:

- WebLogic Server clusters provide scalability and reliability for your applications by distributing the work load among multiple instances of WebLogic Server. Incoming requests can be routed to a WebLogic Server instance in the cluster based on the volume of work being processed. In case of hardware or other failures, session state is available to other cluster nodes that can resume the work of the failed node. In addition, you can implement clusters so that services may be hosted on a single machine with options to migrate the service to another node in the event of failure.

In addition to replicating HTTP session state across servers within a cluster, WebLogic Server can also replicate HTTP session state across multiple clusters, thereby expanding availability and fault tolerance in multiple geographic regions, power grids, and internet service providers.

- Elasticity in dynamic clusters enables the automatic scaling of dynamic clusters and re-provisioning of their associated resources. The elasticity framework leverages the WebLogic Diagnostic Framework (WLDF) policies and actions system.
- Coherence clusters provide scalability and fault tolerance by distributing data across any number of cluster members ensuring that data is always available and easily accessed by any application hosted in WebLogic Server.

In addition, web applications can choose to use a Coherence data grid for storing and replicating HTTP session state to improve scalability, fault tolerance, and performance.

- Work Managers prioritize work based on rules you define and by monitoring actual run time performance statistics. This information is then used to optimize the performance of

your application. Work Managers may be applied globally to a WebLogic Server domain or to a specific application or component.

- Overload protection gives WebLogic Server the ability to detect, avoid, and recover from overload conditions.
- Network channels facilitate the effective use of network resources by segregating network traffic into channels based on the type of traffic.
- Simplified JMS cluster configuration and high availability allows applications to easily scale WebLogic JMS services such as JMS servers, SAF agents, and persistent stores. Cluster-targeted JMS servers and persistent stores allow targeting the JMS service artifacts directly to the cluster and eliminate the need to configure artifacts individually for every server in a cluster.
- WebLogic Server persistent store is a built-in, high-performance storage solution for WebLogic Server subsystems and services that require persistence. For example, it can store persistent JMS messages or temporarily store messages sent using the Store-and-Forward feature. The persistent store supports persistence to a file-based store or to a JDBC-enabled database.
- Store-and-forward services enable WebLogic Server to deliver messages reliably between applications that are distributed across WebLogic Server instances. If the message destination is not available at the moment the messages are sent, either because of network problems or system failures, then the messages are saved on a local server instance and are forwarded to the remote destination once it becomes available.
- Enterprise-ready deployment tools facilitate deployment and migration of applications from the development phase to a production environment.
- Production redeployment enables enterprises to deploy a new version of their application without interrupting work in progress on the older version.

Diagnostic Framework

The WebLogic Diagnostic Framework is a monitoring and diagnostic service that lets you create, collect, analyze, archive, and access diagnostic data generated by a running WebLogic Server instance and applications deployed on it. This diagnostic data provides insight into the runtime performance of WebLogic Server instances, and deployed applications, which helps you isolate and diagnose faults and performance bottlenecks when they occur. See *What Is the WebLogic Diagnostics Framework?* in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

Security

The WebLogic Server security architecture provides a comprehensive, flexible security infrastructure designed to address the security challenges of making applications available on the Internet. WebLogic security can be used standalone to secure WebLogic Server applications, or as part of an enterprise-wide, security management system that represents a best-in-breed security management solution. See *Overview of the WebLogic Security Service* in *Understanding Security for Oracle WebLogic Server* for more information.

Client Options

In addition to support for browser-based web application clients, WebLogic Server also supports a variety of client types for creating rich GUI applications or simple command-line utilities. These client types include: RMI-IIOP, T3, Java SE clients, Java EE thin clients, CORBA/IDL clients, and C++ clients that communicate with BEA Tuxedo. See Overview of Standalone Clients in *Developing Standalone Clients for Oracle WebLogic Server* for more information.

Integration with Oracle WebLogic Suite

WebLogic Server provides the core application server runtime within the integrated Oracle WebLogic Suite Java infrastructure. This integrated infrastructure enhances application performance, improves application availability, and enables predictable and reliable application scalability with high quality of service. WebLogic Suite includes highly productive development tools based on Oracle JDeveloper and [Oracle Enterprise Pack for Eclipse](#), and fully integrated management for large-scale administration and operations with Oracle Enterprise Manager. Taken together, the development, runtime and management capabilities of WebLogic Suite provide the foundation for implementing mission-critical enterprise applications.

WebLogic Suite contains the following server-side components:

- Oracle WebLogic Server
- Oracle Coherence

Oracle Coherence enables organizations to predictably scale mission-critical applications by providing fast and reliable access to frequently used data. By automatically and dynamically partitioning data in memory across multiple servers, Oracle Coherence enables continuous data availability and transactional integrity, even in the event of a server failure.

WebLogic Server includes a Coherence container that simplifies the management and deployment of Coherence clusters and Coherence-based applications.

- Overview of TopLink

Oracle TopLink builds high-performance applications that store persistent object-oriented data in a relational database. It successfully transforms object-oriented data into either relational data or Extensible Markup Language (XML) elements.

Oracle TopLink is an advanced, object-persistence and object-transformation framework that provides development tools and run time capabilities that reduce development and maintenance efforts, and increase enterprise application functionality.

Oracle TopLink includes support for EJB 3.0 in Java EE and Java SE environments, as well as support for EJB 2.n container-managed persistence (CMP). You can integrate Oracle TopLink with a variety of application servers, including Oracle WebLogic Server, JBoss Application Server, and IBM WebSphere Application Server.

Integration with Other Systems

WebLogic Server provides a variety of tools to integrate your applications with disparate systems. These tools include web services, resource adapters, the JMS .NET client, the JMS C client, tooling for integrating JMS providers options, advanced queuing, and RMI.

Integration with Web Servers

Your WebLogic Server installation includes a set of web server plug-ins that allow requests to Oracle WebLogic Server to be proxied from Oracle HTTP Server, Apache HTTP Server, or Microsoft Internet Information Server (IIS). Typically, these web servers serve static HTML content, while requests for dynamic web content, such as JSPs, are directed to the WebLogic Server environment.

Running Oracle WebLogic Server on Docker

Docker is a Linux-based container technology that enables you to quickly create lightweight clustered and nonclustered WebLogic Server domain configurations on a single or multi host OS, or virtual machines, for either development or production environments. Oracle provides Dockerfiles and supporting scripts for building images of Oracle WebLogic Server. These images are built as an extension of existing Oracle Linux and Server JRE images. These scripts and Dockerfiles are available on GitHub at the following location: <https://github.com/oracle/docker-images/tree/master/OracleWebLogic>.

For information about using Docker with WebLogic Server, and the combinations of Oracle WebLogic Server, JDK, Linux and Docker versions that are certified for building your Docker images, see the [Supported Virtualization and Partitioning Technologies for Oracle Fusion Middleware](#).

WebLogic Deploy Tooling

The Oracle WebLogic Deploy Tooling (WDT) simplifies the automation of WebLogic Server domain provisioning and applications deployment.

WDT creates a declarative, metadata model that describes the domain, applications, and resources used by applications. This metadata model makes it easy to provision, deploy, and perform domain lifecycle operations in a repeatable fashion. You can use WDT to migrate on-premises domain configuration and applications to a Docker image or a persistent volume in Kubernetes. For complete documentation and samples, see the open source [WebLogic Deploy Tooling](#) project in GitHub. Or, for a step-by-step guide, read the blog, [Make WebLogic Domain Provisioning and Deployment Easy!](#).

WebLogic Image Tool

The Oracle WebLogic Image Tool lets you automate building, patching, and updating your WebLogic Server Docker images, including your own customized images.

With the WebLogic Image Tool, you can:

- Create a customized WebLogic Server and FMW Infrastructure Docker image.
- Patch a base install image of WebLogic Server or FMW Infrastructure.
- Patch and build a domain image of WebLogic Server or FMW Infrastructure using a WebLogic Deploy Tool (WDT) model.
- Deploy an application and update the domain configuration.

In addition, you can incorporate these use cases into an automated process for patching and updating your WebLogic Server infrastructure and applications running in

Docker and Kubernetes. Find the open source WebLogic Image Tool GitHub project at <https://github.com/oracle/weblogic-image-tool>.

WebLogic Server Kubernetes Operator

The Oracle WebLogic Server Kubernetes Operator is an application-specific controller that extends Kubernetes to create, configure, and manage instances of complex applications. The operator follows the standard Kubernetes operator pattern, and simplifies the management and operation of WebLogic domains and deployments. The operator uses a common set of Kubernetes APIs to provide an improved user experience when automating operations such as provisioning, life cycle management, application versioning, product patching, scaling, and security.

The operator is developed as an open source project fully supported by Oracle. The fastest way to experience the operator is to follow the [Quick Start](#) guide. Alternatively, you can peruse the [documentation](#), read the [blogs](#), or try out the [samples](#). For project scripts, additional samples, and source files, see the [Oracle WebLogic Server Kubernetes Operator GitHub](#) repository.

WebLogic Monitoring Exporter

The Oracle WebLogic Monitoring Exporter is a web application that you can deploy on a WebLogic Server instance that you want to monitor.

The exporter uses the WebLogic Server RESTful Management Interface for accessing runtime state and metrics and then exports [Prometheus](#)-compatible metrics, which can be displayed in Grafana dashboards for monitoring. For practical examples, see these blog posts: [Exporting Metrics from WebLogic Server](#) and [Using Prometheus and Grafana to Monitor WebLogic Server on Kubernetes](#). For a detailed description of the Oracle WebLogic Monitoring Exporter, see the [WebLogic Monitoring Exporter](#) project in GitHub.

WebLogic Logging Exporter

The Oracle WebLogic Logging Exporter provides an easy to configure, robust, and production-ready solution to access WebLogic Server log information through Elasticsearch and displayed in Kibana dashboards.

Find the build and installation instructions in the open source WebLogic Logging Exporter project README at <https://github.com/oracle/weblogic-logging-exporter>.

WebLogic Server API Examples and Sample Application

Oracle WebLogic Server includes a comprehensive set of code examples and sample applications that show how to implement Java EE APIs and Oracle WebLogic Server-specific features in your applications. Code examples demonstrating Java EE APIs and other WebLogic Server features are provided with your WebLogic Server installation. To work with these examples, select the custom installation option when Installing WebLogic Server, and select to install the Server Examples. To access the code examples, launch the `startWebLogicEx.cmd` or `startWebLogicEx.sh` script from `ORACLE_HOME/user_projects/domains/wl_server`, where `ORACLE_HOME` is the directory you specified as the Oracle Home

when you installed Oracle WebLogic. As they become available, you can also download additional examples.

Along with the code examples, two versions of a complete sample application, called Avitek Medical Records (or MedRec), are installed when you install the examples, as described above.

The original MedRec (which was included in previous versions of WebLogic Server) is a WebLogic Server sample application suite that concisely demonstrates all aspects of the Java EE platform. MedRec is designed as an educational tool for all levels of Java EE developers. It showcases the use of each Java EE component and illustrates best practice design patterns for component interaction and client development. MedRec also illustrates best practices for developing applications on WebLogic Server.

The Spring version of MedRec, called MedRec-Spring is MedRec recast using the Spring Framework. If you are developing Spring applications on WebLogic Server, you should review the MedRec-Spring sample application. In order to illustrate how Spring can take advantage of the enterprise features of WebLogic Server, MedRec was rearchitected to replace core Java EE components with their Spring counterparts. The functionality in the original version of MedRec is reimplemented using Spring in MedRec-Spring. Refer to the MedRec-Spring sample for details.

To launch MedRec, run `startWebLogic.cmd` or `startWebLogic.sh` script from `ORACLE_HOME/user_projects/domains/medrec`, where `ORACLE_HOME` is the directory you specified as the Oracle Home when you installed Oracle WebLogic Server.

To launch MedRec-Spring, run the `startWebLogic.cmd` or `startWebLogic.sh` script from `ORACLE_HOME/user_projects/domains/medrec-spring`, where `ORACLE_HOME` is the directory you specified as the Oracle Home when you installed Oracle WebLogic Server.

Upgrade

WebLogic Server provides robust upgrade capabilities to support migrating your application environment from one version of WebLogic Server to the next. 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. Tools and documentation are provided to help you migrate applications implemented on earlier versions of WebLogic Server to the current WebLogic Server environment. See *Upgrading Oracle WebLogic Server*.

2

System Administration

System administration of WebLogic Server includes a wide range of tasks: creating WebLogic Server domains, deploying applications, migrating domains from development environments to production environments, monitoring and managing the performance of the runtime system, configuring and managing security for applications and system resources, diagnosing and troubleshooting problems, and more.

This chapter includes the following topics:

Overview of WebLogic Server System Administration

WebLogic Server provides several tools for system administrators: the browser-based WebLogic Server Administration Console and Fusion Middleware Control; the WebLogic Scripting Tool (WLST), a scripting language for automation of WebLogic system administration tasks based on Jython; a robust set of RESTful management interfaces; SNMP; the Configuration Wizard; and several command-line utilities.

Because the WebLogic Server management system is based on Java EE and other standards, it integrates with systems that are frequently used to manage other software and hardware components. In addition, WebLogic Server implements the Java EE Java Management Extension (JMX) specification, which allows programmatic access to the WebLogic Server management system. Using this API, you can create custom administration utilities or automate frequent tasks using Java classes.

The following sections provide an overview of system administration for the WebLogic Server component of your development or production environments:

- [Choosing the Appropriate Technology for Your Administrative Tasks](#)
- [Summary of System Administration Tools and APIs](#)

For information about installing WebLogic Server, see *Installing and Configuring Oracle WebLogic Server and Coherence*.

Choosing the Appropriate Technology for Your Administrative Tasks

WebLogic Server supports a wide range of technologies for performing administrative tasks, including the browser-based WebLogic Server Administration Console and Fusion Middleware Control (FMWC), the Jython-based WebLogic Scripting Tool, WebLogic RESTful management services, and several components for administering and monitoring domains, applications, server life cycle, performance, and more.

[Table 2-1](#) describes common system administration tasks and associated technologies.

Table 2-1 Choosing the Appropriate Management Technology

To do this...	Use this technology...
Create domains	<p>The Configuration Wizard guides you through the process of creating or extending a domain for your target environment. See <i>Creating WebLogic Domains Using the Configuration Wizard</i>.</p> <p>To automate the creation of domains, use the WebLogic Scripting Tool, which is a command-line scripting interface based on Jython. See <i>Creating Domains Using WLST Offline in Understanding the WebLogic Scripting Tool</i>.</p> <p>Or create domain configuration XML files that conform to the WebLogic Server schema. See <i>Domain Configuration Files in Understanding Domain Configuration for Oracle WebLogic Server</i>.</p>
Migrate domains from development environments to production environments	<p>Domain Template Builder's <code>pack</code> command archives a snapshot of a domain into a JAR file. The <code>unpack</code> command expands the archive and creates the necessary start scripts and certain security and configuration files. See <i>Creating Templates and Domains Using the Pack and Unpack Commands</i>.</p>
Track changes in a domain's configuration	<p>In environments where configuration changes to active domains are allowed, WebLogic Server automatically maintains a versioned archive of configuration files. See <i>Configuration File Archiving in Understanding Domain Configuration for Oracle WebLogic Server</i>.</p> <p>To receive real-time notifications that a domain's configuration has been modified, enable the configuration auditing feature. See <i>Configuring the WebLogic Auditing Provider in Administering Security for Oracle WebLogic Server</i>.</p> <p>For tightly controlled production environments, configure the run-time domain to be read-only (see <i>Restricting Configuration Changes in Understanding Domain Configuration for Oracle WebLogic Server</i>). You can change the read-only setting if you need to roll in changes that have been tested and approved in a staging environment, or you can modify and test your staging environment, and then use a web server to re-route requests from your production environment to the staging environment.</p>
Configure connections to databases or other systems	<p>Within individual applications, you can define your own data sources or database connections using JDBC, or connect to external systems using resource adapters. When you deploy such an application, WebLogic Server creates the data sources and connections for you. See:</p> <ul style="list-style-type: none"> • <i>Configuring WebLogic JDBC Resources in Administering JDBC Data Sources for Oracle WebLogic Server</i> • <i>Understanding Resource Adapters in Developing Resource Adapters for Oracle WebLogic Server</i> <p>If you have not defined your own data sources or connections within an application, you can use the WebLogic Server Administration Console, FMWC, or the WebLogic Scripting Tool to create the resources. See <i>Oracle WebLogic Server Administration Console Online Help</i> or <i>Using the WebLogic Scripting Tool in Understanding the WebLogic Scripting Tool</i>.</p>

Table 2-1 (Cont.) Choosing the Appropriate Management Technology

To do this...	Use this technology...
Manage the server life cycle	<p>The Node Manager is a utility for remote control of Administration Servers and Managed Servers. It runs separately from WebLogic Server and lets you start up and shut down Administration Servers and Managed Servers. While use of Node Manager is optional, it provides additional life cycle benefits if your WebLogic Server environment hosts applications with high availability requirements. See <i>Using Node Manager to Control Servers</i> in the <i>Administering Node Manager for Oracle WebLogic Server</i>.</p> <p>To start Administration Servers or Managed Servers without using Node Manager, use the WebLogic Scripting Tool or scripts that WebLogic Server installs. See <i>Starting and Stopping Servers</i> in <i>Administering Server Startup and Shutdown for Oracle WebLogic Server</i>.</p>
Configure Coherence Clusters	<p>The WebLogic Server Administration Console and FMWC provide a graphical user interface for configuring and managing Coherence clusters; configuring and managing cluster members; and deploying Coherence applications. See the <i>Oracle WebLogic Server Administration Console Online Help</i>.</p> <p>If you prefer a command-line interface, use the WebLogic Scripting Tool. See <i>Using the WebLogic Scripting Tool</i> in <i>Understanding the WebLogic Scripting Tool</i>.</p>
Modify or add services to an active domain	<p>The WebLogic Server Administration Console and FMWC provide a graphical user interface for modifying or adding services to an active domain. See the <i>Oracle WebLogic Server Administration Console Online Help</i>. You can also modify or add services to an active domain using Fusion Middleware Control. See the <i>Oracle Fusion Middleware Control Help for WebLogic Server</i>.</p> <p>If you prefer a command-line interface, use the WebLogic Scripting Tool in interactive mode. See <i>Using the WebLogic Scripting Tool</i> in <i>Understanding the WebLogic Scripting Tool</i>.</p>
Monitor application server services and resources	<p>Monitor the performance of services such as the EJB container, servlet container, and JDBC data sources from the WebLogic Server Administration Console or through Fusion Middleware Control.</p> <p>Configure policy expressions and actions in the WebLogic Diagnostics Framework to automatically notify administrators of monitoring data events or integrate automated systems through JMX or JMS. See <i>Configuring Policies and Actions</i> in <i>Configuring and Using the Diagnostics Framework for Oracle WebLogic Server</i>.</p> <p>If you use SNMP in your operations center, you can enable WebLogic Server to send SNMP notifications for run-time events that you define. See <i>Monitoring Oracle WebLogic Server with SNMP</i>.</p>

Table 2-1 (Cont.) Choosing the Appropriate Management Technology

To do this...	Use this technology...
Deploy applications	<p>The WebLogic Server Administration Console and FMWC provide a series of web-based deployment assistants that guide you through the deployment process. See <i>Oracle WebLogic Server Administration Console Online Help</i>. You can also deploy applications through Fusion Middleware Control. See <i>Oracle Fusion Middleware Control Help for WebLogic Server</i>.</p> <p>To automate the deployment of applications, use the WebLogic Scripting Tool. See Deployment Commands in <i>WLST Command Reference for WebLogic Server</i>. You can also use the deployment API to write Java programs that deploy applications. See <i>Deploying Applications with the WebLogic Deployment API</i>.</p> <p>For information about additional deployment utilities and APIs, see Deployment Tools in <i>Deploying Applications to Oracle WebLogic Server</i>.</p>
Modify applications in an active domain	<p>To modify the configuration of a deployed application, use a text editor or IDE to modify the deployment descriptor. Then either redeploy the application or use the deployment API to upload the modified deployment descriptor and cause the application container to re-read the deployment descriptor.</p> <p>See <i>Deploying Applications to Oracle WebLogic Server</i>.</p>
Monitor activity within applications	<p>Determine which data points you want to monitor and then instrument one or more beans to expose this data through JMX. See <i>Developing Manageable Applications Using JMX for Oracle WebLogic Server</i>.</p> <p>Alternatively, use the WebLogic Server Diagnostics Service to insert instrumentation code into a running application and monitor its methods or monitor transactions that involve the application. Use this technology to discover the cause of problems that cannot otherwise be discovered by scanning the available monitoring metrics. If you determine that the problem is within your application, you can prevent the problem from recurring by using JMX to expose attributes that indicate the application's health state is degrading. See <i>Configuring and Using the Diagnostics Framework for Oracle WebLogic Server</i>.</p>
Optimize the performance of your application and maintain service-level agreements.	<p>Work Managers configure how your application prioritizes the execution of its work. Based on rules you define and by monitoring actual run-time performance, WebLogic Server can optimize the performance of your application and maintain service-level agreements.</p> <p>See <i>Using Work Managers to Optimize Scheduled Work in Administering Server Environments for Oracle WebLogic Server</i>.</p>
Configure and secure administration communications	<p>You can separate administration traffic from application traffic in your domain by enabling the administration port. In production environments, separating the two forms of traffic ensures that critical administration operations (starting and stopping servers, changing a server's configuration, and deploying applications) do not compete with high-volume application traffic on the same network connection.</p> <p>The administration port only accepts communications that use SSL, and therefore secures your administrative requests. See Administration Port and Administrative Channel in <i>Administering Server Environments for Oracle WebLogic Server</i>.</p>

Table 2-1 (Cont.) Choosing the Appropriate Management Technology

To do this...	Use this technology...
Configure logging and view log files	<p>Many WebLogic Server operations generate logs of their activity. Each server has its own log as well as a standard HTTP access log. These log files can be configured and used in a variety of ways to monitor the health and activity of your servers and applications.</p> <p>By default, WebLogic Server uses the standard JDK logging APIs to filter and write the messages to log files. See Understanding WebLogic Logging Services in <i>Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server</i>.</p>

Summary of System Administration Tools and APIs

WebLogic Server includes several of its own standards-based, extensible utilities that you can use to create, manage, and monitor domains. As an alternative, you can also use WebLogic Server's management APIs to create custom management utilities.

[Table 2-2](#) describes the utilities that are included with WebLogic Server.

Table 2-2 Management Utilities

Utility	Description
WebLogic Server Administration Console	<p>The WebLogic Server Administration Console is a web application hosted by the Administration Server. Use it to manage and monitor an active domain. The management capabilities include:</p> <ul style="list-style-type: none"> • Configuring active domains • Stopping and starting servers • Monitoring server health and performance • Monitoring application performance • Viewing server logs • Control (start, stop, and restart) managed Coherence servers • Create and configure Coherence clusters <p>Through the WebLogic Server Administration Console, system administrators can easily perform all WebLogic Server management tasks without having to learn the JMX API or the underlying management architecture. The Administration Server persists changes to attributes in the <code>config.xml</code> file for the domain you are managing.</p> <p>See:</p> <ul style="list-style-type: none"> • Overview of Administration Consoles • <i>Oracle WebLogic Server Administration Console Online Help</i> (The online help is also available from the WebLogic Server Administration Console by clicking on the Help link located in the tool bar at the top of the Console.)

Table 2-2 (Cont.) Management Utilities

Utility	Description
WebLogic Scripting Tool	<p>The WebLogic Scripting Tool (WLST) is a command-line scripting interface that you use to manage and monitor active or inactive WebLogic Server domains. The WLST scripting environment is based on the Java scripting interpreter Jython. In addition to WebLogic scripting functions, you can use common features of interpreted languages, including local variables, conditional variables, and flow control statements. You can extend the WebLogic scripting language by following the Jython language syntax. See http://www.jython.org.</p> <p>See <i>Understanding the WebLogic Scripting Tool</i>.</p>
RESTful management resources	<p>WebLogic RESTful management resources provide a comprehensive public interface for configuring, monitoring, deploying and administering WebLogic Server in all supported environments. See About the WLS RESTful Management Interface in <i>Administering Oracle WebLogic Server with RESTful Management Services</i>.</p>
Configuration Wizard	<p>The Configuration Wizard creates the appropriate directory structure for a WebLogic Server domain, a <code>config.xml</code> file, and scripts you can use to start the servers in your domain. The wizard uses templates to create domains, and you can customize these templates to duplicate your own domains.</p> <p>You can also use the Configuration Wizard to add or remove services from an existing, inactive domain.</p> <p>You can run the Configuration Wizard through a graphical user interface (GUI) or in a text-based command-line environment. This command-line environment is called <i>console mode</i>—do not confuse this mode with the WebLogic Server Administration Console. You can also create user-defined domain configuration templates for use with the Configuration Wizard.</p> <p>See <i>Creating WebLogic Domains Using the Configuration Wizard</i>.</p>
Domain Template Builder	<p>The Domain Template Builder provides the capability to easily create your own domain templates, to enable, for example, the definition and propagation of a standard domain across a development project, or to enable the distribution of a domain along with an application that has been developed to run on that domain. The templates you create with the Configuration Template Builder are used as input to the Configuration Wizard as the basis for creating a domain that is customized for your target environment. See <i>Creating Domain Templates Using the Domain Template Builder</i>.</p>
Apache Ant tasks	<p>You can use two Ant tasks provided with WebLogic Server to help you perform common configuration tasks in a development environment. Ant is a Java-based build tool similar to Make. The configuration tasks let you start and stop WebLogic Server instances as well as create and configure WebLogic Server domains. When combined with other WebLogic Ant tasks, you can create powerful build scripts for demonstrating or testing your application with custom domains.</p> <p>See Using Ant Tasks to Configure a WebLogic Server Domain in <i>Developing Applications for Oracle WebLogic Server</i>.</p>

Table 2-2 (Cont.) Management Utilities

Utility	Description
SNMP Agents	<p>WebLogic Server includes the ability to communicate with enterprise-wide management systems using Simple Network Management Protocol (SNMP). WebLogic Server SNMP agents let you integrate management of WebLogic Servers into an SNMP-compliant management system that gives you a single view of the various software and hardware resources of a complex, distributed system.</p> <p>See <i>Monitoring Oracle WebLogic Server with SNMP</i>.</p>

Table 2-3 describes APIs that you can use to create your own management utilities.

Table 2-3 Management APIs

API	Description
JMX	<p>Java Management Extensions (JMX) is the Java EE solution for monitoring and managing resources on a network. Like SNMP and other management standards, JMX is a public specification and many vendors of commonly used monitoring products support it.</p> <p>The WebLogic Server Administration Console, WebLogic Scripting Tool, and other WebLogic Server utilities use the JMX APIs.</p> <p>See <i>Developing Custom Management Utilities Using JMX for Oracle WebLogic Server</i>.</p>
Java EE Management API	<p>The Java EE Management APIs (JSR-77) enable a software developer to create a single Java program that can discover and browse resources, such as JDBC connection pools and deployed applications, on any Java EE web application server. The APIs are part of the Java EE Management Specification, which requires all Java EE web application servers to describe their resources in a standard data model.</p> <p>See <i>Developing Java EE Management Applications for Oracle WebLogic Server</i>.</p>
Deployment API	<p>The WebLogic Server deployment API implements and extends the JSR-88 deployment specification. All WebLogic Server deployment tools, such as the WebLogic Server Administration Console and wldesploy Ant task, use the deployment API to configure, deploy, and redeploy applications in a domain. You can use the deployment API to build your own WebLogic Server deployment tools, or to integrate WebLogic Server configuration and deployment operations with an existing JSR-88-compliant tool.</p> <p>See <i>Deploying Applications with the WebLogic Deployment API</i>.</p>
WebLogic Diagnostic Service APIs	<p>The WebLogic Diagnostic Service includes a set of standardized APIs that enable dynamic access and control of diagnostic data, as well as improved monitoring that provides visibility into the server. The interfaces are standardized to facilitate future enhancement and integration of third-party tools, while maintaining the integrity of the server code base. The service is well suited to the server and the server's stack product components and targets operations and administrative staff as primary users.</p> <p>See <i>Configuring and Using the Diagnostics Framework for Oracle WebLogic Server</i>.</p>

Table 2-3 (Cont.) Management APIs

API	Description
Logging APIs	By default, WebLogic Server uses the standard JDK logging APIs to filter and write the messages to log files. See <i>Understanding WebLogic Logging Services in Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server</i> .

Roadmap for Administering the WebLogic Server System

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to use each of the administration tools and technologies provided by WebLogic Server.

Table 2-4 Roadmap for Administering the WebLogic Server System

Major Task	Subtasks and Additional Information
Understanding WebLogic Server system administration	<ul style="list-style-type: none"> • Overview of WebLogic Server domains • Overview of WebLogic Server clusters • Overview of WebLogic security • Overview of administration consoles • <i>Developing Custom Management Utilities Using JMX for Oracle WebLogic Server</i> • <i>Tuning Performance of Oracle WebLogic Server</i>
Installing or upgrading WebLogic Server	<ul style="list-style-type: none"> • <i>Installing and Configuring Oracle WebLogic Server and Coherence</i> • <i>Creating WebLogic Domains Using the Configuration Wizard</i> • Oracle Fusion Middleware Supported System Configurations • <i>What's New in Oracle WebLogic Server</i> • <i>Release Notes for Oracle WebLogic Server</i> • WebLogic Server compatibility • <i>Upgrading Oracle WebLogic Server</i>
Configuring a server environment	<ul style="list-style-type: none"> • Summary of system administration tools and APIs • Managing configuration changes • <i>Oracle WebLogic Server Administration Console Online Help</i> • <i>Understanding the WebLogic Scripting Tool</i> • <i>Creating Domain Templates Using the Domain Template Builder</i>
Learning about server startup and shutdown	<ul style="list-style-type: none"> • Overview of starting and stopping servers • Understanding the life cycle of WebLogic Server instances • Server startup command-line reference • Quick reference for starting and stopping servers
Starting or stopping a WebLogic Server instance	<ul style="list-style-type: none"> • Using shell scripts • Using the Administration Console • Using the WebLogic Scripting Tool (WLST) • Using Node Manager to control remote servers • Using the quick reference
Configuring Coherence clusters	<ul style="list-style-type: none"> • Configuring and managing Coherence clusters • <i>Developing Oracle Coherence Applications for Oracle WebLogic Server</i> • Securing Coherence in WebLogic Server

Table 2-4 (Cont.) Roadmap for Administering the WebLogic Server System

Major Task	Subtasks and Additional Information
Configuring security	<ul style="list-style-type: none"> • Overview of WebLogic Server security • <i>Administering Security for Oracle WebLogic Server</i> • <i>Securing a Production Environment for Oracle WebLogic Server</i> • <i>Securing Resources Using Roles and Policies for Oracle WebLogic Server</i>
Managing server and network communications	<ul style="list-style-type: none"> • Configuring network resources • Configuring web server functionality • <i>Using Oracle WebLogic Server Proxy Plug-Ins</i>
Configuring system resources	<ul style="list-style-type: none"> • <i>Administering JDBC Data Sources for Oracle WebLogic Server</i> • <i>Administering JMS Resources for Oracle WebLogic Server</i> • Configuring WebLogic transactions • Configuring the WebLogic Tuxedo Connector • Configuring the persistent store
Configuring and deploying applications	<ul style="list-style-type: none"> • <i>Deploying Applications to Oracle WebLogic Server</i> • Configuring web applications • Configuring XML resources • Configuring resource adapters • <i>Understanding WebLogic Web Services for Oracle WebLogic Server</i>
Monitoring your domain	<ul style="list-style-type: none"> • <i>Configuring and Using the Diagnostics Framework for Oracle WebLogic Server</i> • <i>Monitoring Oracle WebLogic Server with SNMP</i> • <i>Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server</i> • <i>Developing Java EE Management Applications for Oracle WebLogic Server</i> • Using the Monitoring Dashboard
Configuring server environments for high availability	<ul style="list-style-type: none"> • Understanding cluster architectures • Setting up WebLogic Server clusters • Using session replication across clusters • Using Work Managers to prioritize application execution • Avoiding and managing overload
Understanding the WebLogic persistent store	<ul style="list-style-type: none"> • Using the WebLogic persistent store • Configuring custom persistent stores • Tuning the WebLogic persistent store
Troubleshooting	<ul style="list-style-type: none"> • Viewing the WebLogic Server error message catalog • <i>Tuning Performance of Oracle WebLogic Server</i> • Troubleshooting common problems with clustering • <i>Administering Node Manager for Oracle WebLogic Server</i>
Reference	<ul style="list-style-type: none"> • <i>Administration Console Accessibility Notes for Oracle WebLogic Server</i> • <i>Command Reference for Oracle WebLogic Server</i> • SNMP MIB for Oracle WebLogic Server • <i>WLST Command Reference for WebLogic Server</i> • <i>MBean Reference for Oracle WebLogic Server</i>

3

Overview of Administration Consoles

This chapter includes the following topic:

Using the WebLogic Server Administration Console

The WebLogic Server Administration Console is a web browser-based, graphical user interface that you use to manage a WebLogic Server domain. It provides the starting point for essential operations, administration, automation, and management.

This topic includes the following sections:

About the WebLogic Server Administration Console

The WebLogic Server Administration Console is a web browser-based, graphical user interface that you use to manage a WebLogic Server domain. A WebLogic Server domain is a logically related group of WebLogic Server resources that you manage as a unit. A domain includes one or more WebLogic Servers and may also include WebLogic Server clusters. Clusters are groups of WebLogic Servers instances that work together to provide scalability and high availability for applications. You deploy and manage your applications as part of a domain.

One instance of WebLogic Server in each domain is configured as an Administration Server. The Administration Server provides a central point for managing a WebLogic Server domain. All other WebLogic Server instances in a domain are called Managed Servers. In a domain with only a single WebLogic Server instance, that server functions both as Administration Server and Managed Server. The Administration Server hosts the WebLogic Server Administration Console, which is a web application accessible from any supported web browser with network access to the Administration Server. Managed Servers host applications.

Use the WebLogic Server Administration Console to:

- Configure, start, and stop WebLogic Server instances
- Configure WebLogic Server clusters
- Configure WebLogic Server services, such as database connectivity (JDBC) and messaging (JMS)
- Configure security parameters, including managing users, groups, and roles
- Configure and deploy your applications
- Monitor server and application performance
- View server and domain log files
- View application deployment descriptors
- Edit selected run-time application deployment descriptor elements
- Control (start, stop, and restart) managed Coherence servers
- Create and configure Coherence clusters

WebLogic Server Administration Console Online Help

The WebLogic Server Administration Console includes a complete help system. It has two parts:

- How do I...?, which documents procedures for tasks you can perform using the WebLogic Server Administration Console.
- WebLogic Server Administration Console Reference, which provides reference information for each page in the WebLogic Server Administration Console, including descriptions of the attributes you can set using the WebLogic Server Administration Console.

You can access the WebLogic Server Administration Console online help either through the WebLogic Server Administration Console itself, or in [Oracle WebLogic Server Administration Console Online Help](#).

Console Errors

Messages (including information, warning, and error messages) can be generated and logged in the course of using the WebLogic Server Administration Console. You can view WebLogic Server logs from the **Diagnostics > Log Files** page of the WebLogic Server Administration Console.

Starting the WebLogic Server Administration Console

This section contains instructions for starting the WebLogic Server Administration Console.

To use the WebLogic Server Administration Console, use one of the supported web browsers for your environment. See Supported Configurations in *What's New in Oracle WebLogic Server*. If your web browser is not a supported browser, you may experience functional or formatting problems when using the WebLogic Server Administration Console.

To start the WebLogic Server Administration Console:

1. Start an Administration Server.
2. Open one of the supported web browsers to the following URL:

```
http://hostname:port/console
```

where *hostname* is the DNS name or IP address of the Administration Server and *port* is the listen port on which the Administration Server is listening for requests (port 7001 by default). If you have configured a domain-wide administration port, use that port number. If you configured the Administration Server to use Secure Socket Layer (SSL) you must add *s* after `http` as follows:

```
https://hostname:port/console
```

Note:

A domain-wide administration port always uses SSL.

3. When the login page appears, enter the user name and the password you used to start the Administration Server (you may have specified this user name and password during the installation process) or enter a user name that belongs to one of the following security groups: Administrators, Operators, Deployers, or Monitors. These groups provide various levels of access to system administration functions in the WebLogic Server Administration Console.

Using the security system, you can add or delete users to one of these groups to provide controlled access to the Console.

 **Note:**

If you have your browser configured to send HTTP requests to a proxy server, then you may need to configure your browser to not send Administration Server HTTP requests to the proxy. If the Administration Server is on the same machine as the browser, then ensure that requests sent to localhost or 127.0.0.1 are not sent to the proxy.

Enabling the WebLogic Server Administration Console

By default, the WebLogic Server Administration Console is enabled. If you disable it, you can reenable it using the WebLogic Scripting Tool (WLST). Start the Administration Server, then invoke WLST and use the following commands:

Example 3-1 Using WLST to Reenable the Console

```
connect("username","password")
edit()
startEdit()
cmo.setConsoleEnabled(true)
save()
activate()
The following attribute(s) have been changed on MBeans which require server re-start.
MBean Changed : com.bea:Name=mydomain,Type=Domain Attributes changed :
ConsoleEnabled
Activation completed
disconnect()
exit()
```

For information about using WLST, see *Understanding the WebLogic Scripting Tool*.

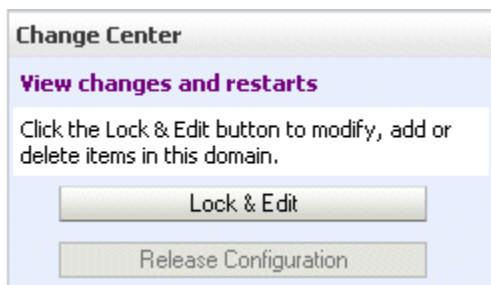
Elements of the WebLogic Server Administration Console

The WebLogic Server Administration Console user interface includes the following panels:

Change Center

This is the starting point for using the WebLogic Server Administration Console to make changes in WebLogic Server. See [Using the Change Center](#).

Figure 3-1 Change Center



Domain Structure

This panel contains a tree structure you can use to navigate to pages in the WebLogic Server Administration Console. Select any of the nodes in the Domain Structure tree to view that page. Click a + (plus) icon in the Domain Structure to expand a node and a - (minus) icon to collapse the node.

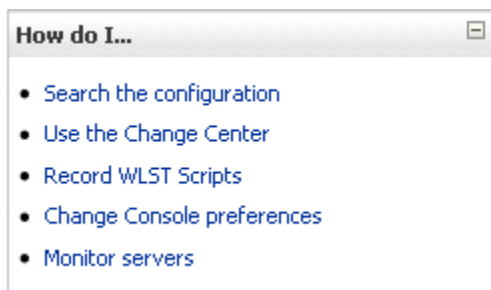
Figure 3-2 Domain Structure



How Do I...

This panel includes links to online help tasks that are relevant to the current Console page.

Figure 3-3 How do I...



Tool Bar

The tool bar at the top of the Console includes the following elements:

Tool Bar Element	Description
Welcome message	Indicates user name with which you have logged into the Console.
Connected to:	The IP address and port you used to connect to the Console.
Home	A link to the top page of the Console.
Log Out	Click to log out of the Console.
Preferences	A link to a page where you can change some Console behavior.
Record	Starts recording your configuration actions as a series of WebLogic Scripting Tool (WLST) commands. Writes the commands to a separate file that you can replay in WLST. See Record WLST Scripts in <i>Oracle WebLogic Server Administration Console Online Help</i> .
Help	A link to the WebLogic Server Administration Console Online Help.
Search	A text field in which you can enter a string to find any WebLogic Server Configuration MBeans that contain the string you specified in their name.

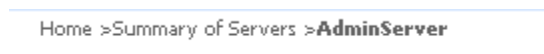
Figure 3-4 Tool Bar



Breadcrumb Navigation

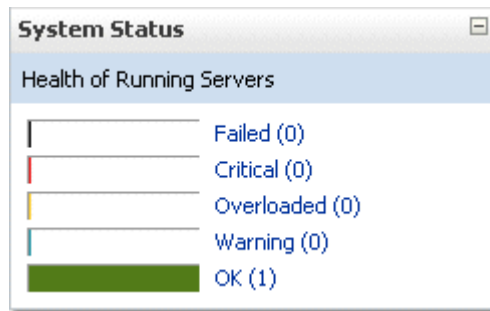
A series of links that show the path you have taken through the WebLogic Server Administration Console's pages. You can click on any of the links to return to a previously-visited page.

Figure 3-5 Breadcrumb Navigation



System Status

The System Status panel reports on the number of information, error, and warning messages that have been logged. You can view these messages in the server log files, which you can access from the WebLogic Server Administration Console at **Diagnostics > Log Files**.

Figure 3-6 System Status

Using the Change Center

The starting point for using the WebLogic Server Administration Console to make changes in your WebLogic Server domain is the Change Center. The Change Center provides a way to lock a domain configuration so you can make changes to the configuration while preventing other accounts from making changes during your edit session.

The domain configuration locking feature is always enabled in production domains. It can be enabled or disabled in development domains. It is disabled by default when you create a new development domain. See [Enable and disable the domain configuration lock](#) in *Oracle WebLogic Server Administration Console Online Help*.

To change a production domain's configuration, you must:

1. Locate the Change Center in the upper left of the WebLogic Server Administration Console screen.
2. Click the **Lock & Edit** button to lock the configuration edit hierarchy for the domain.
3. Make the changes you desire on the relevant page of the Console. Click **Save** on each page where you make a change.
4. When you have finished making all the desired changes, click **Activate Changes** in the Change Center.

As you make configuration changes using the WebLogic Server Administration Console, you click **Save** (or in some cases **Finish**) on the appropriate pages. This does not cause the changes to take effect immediately. The changes take effect when you click **Activate Changes** in the Change Center. At that point, the configuration changes are distributed to each of the servers in the domain. If the changes are acceptable to each of the servers, then they take effect. If any server cannot accept a change, then all of the changes are rolled back from all of the servers in the domain. The changes are left in a pending state; you can then either edit the pending changes to resolve the problem or revert the pending changes.

Undoing Changes

You can revert any pending (saved, but not yet activated) changes by clicking **Undo All Changes** in the Change Center. You can revert any individual change by going to the appropriate page in the WebLogic Server Administration Console and restoring the attribute to its previous value.

Releasing the Configuration Lock

You release the configuration lock as follows:

- Before you make changes, click **Release Configuration** in the Change Center to release the lock explicitly.
- After you save changes, click **Activate Changes** or **Undo All Changes** in the Change Center to release the lock implicitly.

Stopping the Administration Server does not release the configuration lock. When the Administration Server starts again, the configuration lock is in the same state it was in when the Administration Server was shut down, and any pending changes are preserved.


How Change Management Works

To provide a secure, predictable means for distributing configuration changes in a domain, WebLogic Server imposes a change management process that loosely resembles a database transaction. The configuration of a domain is represented on the file system by a set of XML configuration files, centralized in the `config.xml` file, and at run time by a hierarchy of Configuration MBeans. When you edit the domain configuration, you edit a separate hierarchy of Configuration MBeans that resides on the Administration Server. To start the edit process, you obtain a lock on the edit hierarchy to prevent other people from making changes. When you finish making changes, you save the changes to the edit hierarchy. The changes do not take effect, however, until you activate them, distributing them to all server instances in the domain. When you activate changes, each server determines whether it can accept the change. If all servers are able to accept the change, they update their working configuration hierarchy and the change is completed.

For more information about change management, see *Managing Configuration Changes in Understanding Domain Configuration for Oracle WebLogic Server*.

Dynamic and Non-Dynamic Changes

Some changes you make in the WebLogic Server Administration Console take place immediately when you activate them. Other changes require you to restart the server or module affected by the change. These latter changes are called *non-dynamic changes*. Non-dynamic changes are indicated in the WebLogic Server Administration Console with this

warning icon: 

Changes to dynamic configuration attributes become available once they are activated, without restarting the affected server or system restart. These changes are made available to the server and run-time hierarchies once they are activated. Changes to non-dynamic configuration attributes require that the affected servers or system resources be restarted before they become effective.

If a change is made to a non-dynamic configuration setting, no changes to dynamic configuration settings will take effect until after restart. This is to assure that a batch of updates having a combination of dynamic and non-dynamic attribute edits will not be partially activated.

Note that WebLogic Server's change management process applies to changes in domain and server configuration data, not to security or application data.

Viewing Changes

You can view any changes that you have saved, but not yet activated, by clicking the **View Changes** and **Restarts** link in the Change Center. The **View Changes** and **Restarts** link presents two tabs, **Change List** and **Restart Checklist**:

- The Change List page presents all changes that have been saved, but not yet activated.
- The Restart Checklist lists all servers for which non-dynamic changes have been activated, but which require restarts before the changes become effective.

4

WebLogic Server Domains

A WebLogic Server domain is a logically related group of WebLogic Server instances, and the resources running on and connected to them, that can be managed as a single administrative unit.

This chapter includes the following topics:

Understanding Domains

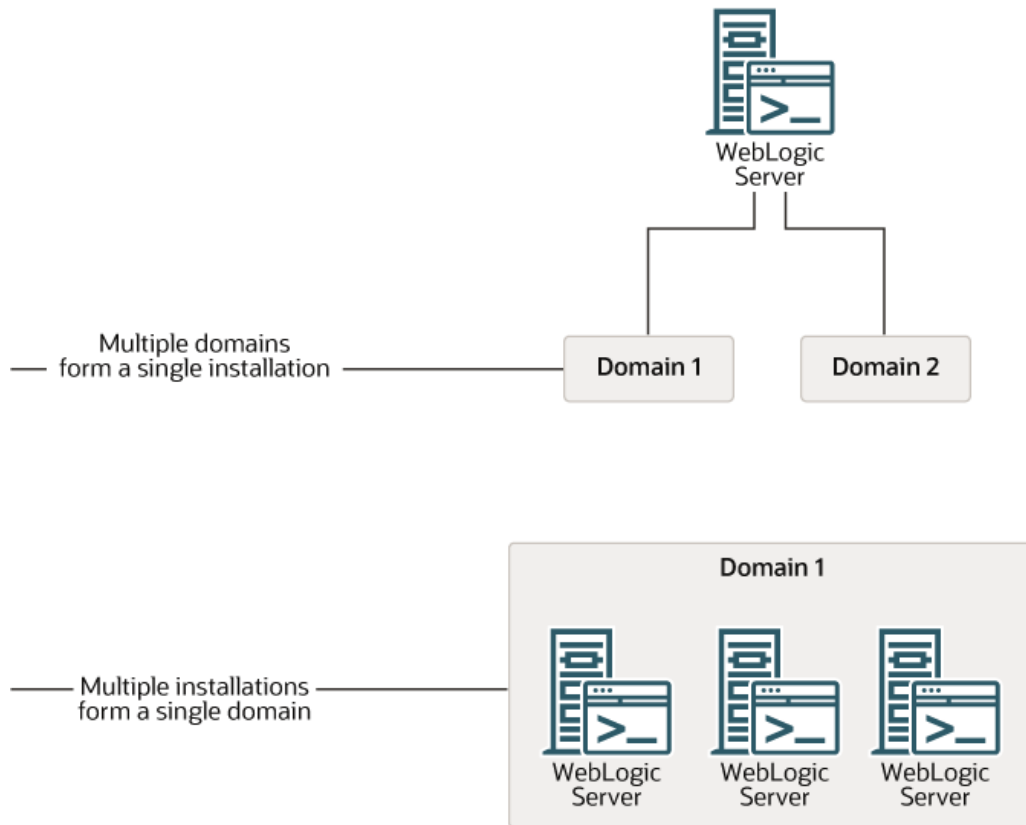
The core of a WebLogic domain consists of the Administration Server, which is the central point from which you configure and manage all resources in the domain.

Usually, you configure a domain to include additional WebLogic Server instances called **Managed Servers**. You deploy web applications, EJBs, web services, and other resources onto the Managed Servers and use the **Administration Server** for configuration and management purposes only.

Organizing Domains

A WebLogic domain is distinct from a WebLogic Server installation. You create and run multiple domains using a single WebLogic Server installation, or you can create a single domain that encompasses multiple installations. How you organize your domains is based on your needs.

[Figure 4-1](#) shows the relationship between a WebLogic Server installation and a WebLogic domain.

Figure 4-1 Oracle WebLogic Server Installations and Domains

How you organize your Oracle WebLogic Server installations into domains depends on your business needs. You can define multiple domains based on different system administrators' responsibilities, application boundaries, or geographical locations of the machines on which servers run. Conversely, you might decide to use a single domain to centralize all Oracle WebLogic Server administration activities.

Depending on your particular business needs and system administration practices, you might decide to organize your domains based on criteria such as:

- Logical divisions of applications. For example, you might have one domain devoted to end-user functions such as shopping carts and another domain devoted to back-end accounting applications.
- Physical location. You might establish separate domains for different locations or branches of your business. Each physical location requires its own Oracle WebLogic Server installation.
- Size. You might find that domains organized in small units can be managed more efficiently, perhaps by different system administrators. Contrarily, you might find that maintaining a single domain or a small number of domains makes it easier to maintain a consistent configuration.

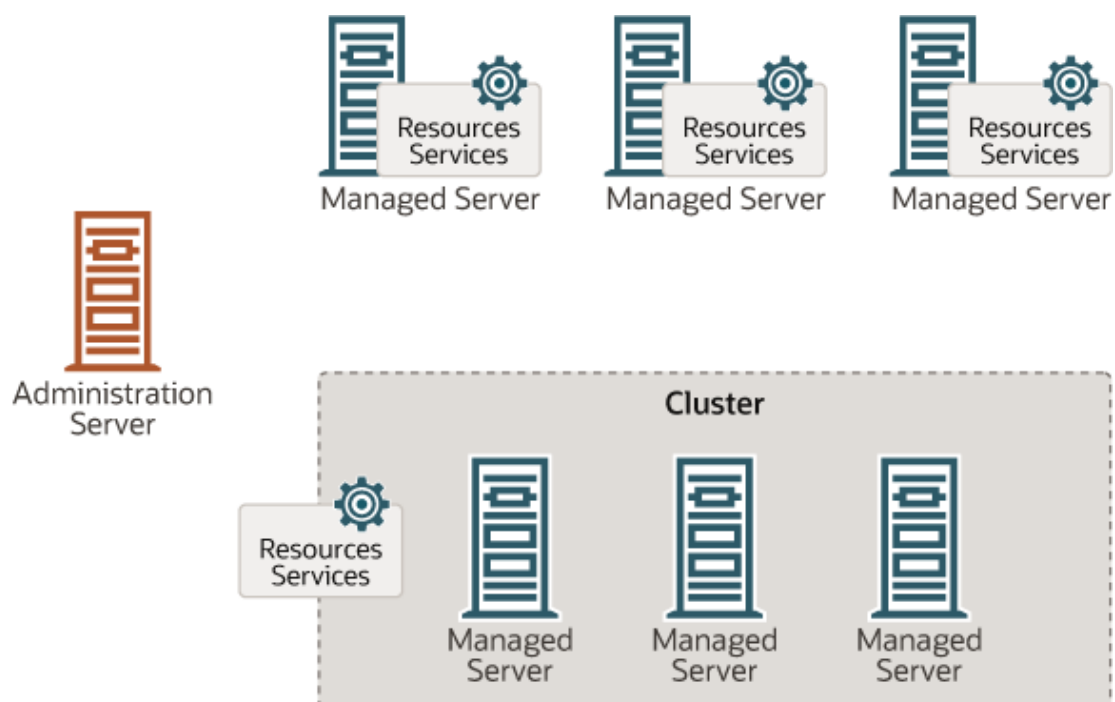
You can create a simple domain that consists of a single server instance. This single instance acts as an Administration Server and hosts the applications that you are developing. Although a single server domain is typically used for development and test environments, this domain type is fully supported for production use and may be

appropriate for light-load applications. The `wl_server` domain that you can install with Oracle WebLogic Server is an example of this type of domain.

Contents of a Domain

A domain consists of an Administration Server and optionally one or more Managed Servers. A domain also contains a variety of resources and services used by the those server instances. [Figure 4-2](#) shows a production environment that contains an Administration Server, three stand-alone Managed Servers, and a cluster of three Managed Servers. Although the scope and purpose of a domain can vary significantly, most Oracle WebLogic Server domains contain the components described in this section.

Figure 4-2 Contents of a Domain



The following topics describe the primary components of a domain, which may optionally include managed Coherence servers and Coherence clusters:

Administration Server

The Administration Server operates as the central control entity for the configuration of the entire domain. It maintains the domain's configuration documents and distributes changes in the configuration documents to Managed Servers. You can also use the Administration Server as a central location from which to monitor all resources in a domain.

To interact with the Administration Server, you can use any of the administration tools listed in [Summary of System Administration Tools and APIs](#). See [System Administration](#) for information about modifying the domain's configuration.

Each Oracle WebLogic Server domain must have one server instance that acts as the Administration Server.

For more information about the Administration Server and its role in the Oracle WebLogic Server JMX management system, see [System Administration](#).

Managed Servers and Managed Server Clusters

Managed Servers host business applications, application components, web services, and their associated resources. To optimize performance, Managed Servers maintain a read-only copy of the domain's configuration document. When a Managed Server starts up, it connects to the domain's Administration Server to synchronize its configuration document with the document that the Administration Server maintains.

For production environments that require increased application performance, throughput, or high availability, you can configure two or more Managed Servers to operate as a **cluster**. A cluster is a collection of multiple Oracle WebLogic Server instances running simultaneously and working together to provide increased scalability and reliability. In a cluster, most resources and services are deployed identically to each Managed Server (as opposed to a single Managed Server), enabling failover and load balancing. A single domain can contain multiple Oracle WebLogic Server clusters, as well as multiple Managed Servers that are not configured as clusters. The key difference between clustered and non-clustered Managed Servers is support for failover and load balancing. These features are available only in a cluster of Managed Servers. For more information about the benefits and capabilities of an Oracle WebLogic Server cluster, see *Understanding WebLogic Server Clustering in Administering Clusters for Oracle WebLogic Server*.

Managed Coherence Servers and Coherence Clusters

Managed Coherence servers provide in-memory distributed caching for applications. A Managed Server that is configured to be a Coherence cluster member is a managed Coherence server. Coherence is integrated within WebLogic Server as a container subsystem. The use of a container aligns the lifecycle of a Coherence member with the lifecycle of a Managed Server: starting or stopping a server JVM starts and stops a Coherence cluster member.

A domain can contain a single Coherence cluster that can be associated with multiple WebLogic Server clusters. Managed Coherence servers that are part of a WebLogic Server cluster inherit their Coherence settings from the WebLogic Server cluster. WebLogic Server clusters are typically used to setup Coherence tiers that organize managed Coherence servers based on their role in the Coherence cluster.

For details on configuring and managing Coherence clusters, see *Configuring and Managing Coherence Clusters in Administering Clusters for Oracle WebLogic Server*.

Resources and Services

In addition to the Administration Server and Managed Servers, a domain also contains the resources and services that Managed Servers and deployed applications require.

Managed Servers can use the following resources:

- Machine definitions that identify a particular, physical piece of hardware. A machine definition is used to associate a computer with the Managed Servers it hosts. This information is used by Node Manager in restarting a failed Managed Server, and by a clustered Managed Server in selecting the best location for storing replicated session data. For more information about Node Manager, see

Node Manager Overview in the *Administering Node Manager for Oracle WebLogic Server*.

- Network channels that define default ports, protocols, and protocol settings that a Managed Server uses to communicate with clients. After creating a network channel, you can assign it to any number of Managed Servers and clusters in the domain. See *Configuring Network Resources in Administering Server Environments for Oracle WebLogic Server*.
- Virtual hosting, which defines a set of host names to which Oracle WebLogic Server instances (servers) or clusters respond. When you use virtual hosting, you use DNS to specify one or more host names that map to the IP address of a server or cluster. You also specify which web applications are served by each virtual host.

Applications can use the following resources and services:

- Security providers, which are modular components that handle specific aspects of security, such as authentication and authorization.
- Resource adapters, which are system libraries specific to Enterprise Information Systems (EIS) and provide connectivity to an EIS.
- Diagnostics and monitoring services.
- JDBC data sources, which enable applications to connect to databases.
- Mail sessions.
- XML entity caches and registry of XML parsers and transformer factories.
- Messaging services such as JMS servers and store-and-forward services.
- Persistent store, which is a physical repository for storing data, such as persistent JMS messages. It can be either a JDBC-accessible database or a disk-based file.
- Startup classes, which are Java programs that you create to provide custom, system-wide services for your applications.
- Work Managers, which determine how an application prioritizes the execution of its work based on rules you define and by monitoring actual run-time performance. You can create Work Managers for entire Oracle WebLogic Server domains or for specific application components.
- Work Contexts, which enable applications to pass properties to a remote context without including the properties in a remote call.

Roadmap for Understanding WebLogic Server Domains

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to create, configure, and manage WebLogic domains.

Table 4-1 Roadmap for Understanding WebLogic Server Domains

Major Task	Subtasks and Additional Information
Learning more about WebLogic Server domains	<ul style="list-style-type: none"> • What to do if the Administration Server fails • Domain restrictions • Domain configuration files • Overview of change management • System Administration

Table 4-1 (Cont.) Roadmap for Understanding WebLogic Server Domains

Major Task	Subtasks and Additional Information
Creating domains	<ul style="list-style-type: none">• <i>Creating WebLogic Domains Using the Configuration Wizard</i>• Overview of the Configuration Wizard• Extending WebLogic domains• <i>Creating Templates and Domains Using the Pack and Unpack Commands</i>• Creating WebLogic domains using WLST offline
Configuring domains	<ul style="list-style-type: none">• Configuring existing WebLogic domains• <i>Understanding Domain Configuration for Oracle WebLogic Server</i>• Managing configuration changes
Working with domain templates	<ul style="list-style-type: none">• <i>Creating Domain Templates Using the Domain Template Builder</i>• Creating and using a domain template (offline)
Examples	<ul style="list-style-type: none">• WLST offline sample scripts <p>In addition, sample scripts are provided that configure WebLogic domain resources using WLST offline and online on the Oracle Technology Network site.</p>
Reference	<ul style="list-style-type: none">• <i>Domain Template Reference</i>• Domain configuration schema• Domain security schema

5

WebLogic Server Clustering

The foundation of high availability in WebLogic Server is the cluster. A WebLogic Server cluster is a group of WebLogic Server instances running simultaneously and working together to provide increased scalability and reliability.

This chapter includes the following topics:

Overview of WebLogic Server Clusters

The server instances that constitute a cluster can run on the same machine, or be distributed across multiple machines. A cluster appears to clients to be a single WebLogic Server instance. You can increase a cluster's capacity by adding additional server instances to the cluster on an existing machine, or you can add machines to the cluster to host the incremental server instances. Each server instance in a cluster must run the same version of WebLogic Server.

Relationship Between Clusters and Domains

A cluster is part of a particular WebLogic domain. A domain includes one or more WebLogic Server instances. In a domain with multiple server instances, those servers can be clustered, nonclustered, or a combination of clustered and nonclustered instances.

A domain can include multiple clusters. A domain also contains the application components deployed in the domain, and the resources and services required by those application components and the server instances in the domain. Examples of the resources and services used by applications and server instances include machine definitions, optional network channels, connectors, and startup classes.

You can use a variety of criteria for organizing WebLogic Server instances into domains. For instance, you might choose to allocate resources to multiple domains based on logical divisions of the hosted application, geographical considerations, or the number or complexity of the resources under management. For additional information about domains see *Understanding Oracle WebLogic Server Domains* in *Understanding Domain Configuration for Oracle WebLogic Server*.

In each domain, one WebLogic Server instance acts as the Administration Server—the server instance which configures, manages, and monitors all other server instances and resources in the domain. Each Administration Server manages one domain only. If a domain contains multiple clusters, each cluster in the domain has the same Administration Server. All server instances in a cluster must reside in the same domain; you cannot "split" a cluster over multiple domains. Similarly, you cannot share a configured resource or subsystem between domains.

Clustered WebLogic Server instances behave similarly to nonclustered instances, except that they provide failover and load balancing. The process and tools used to configure clustered WebLogic Server instances are the same as those used to configure nonclustered instances. However, to achieve the load balancing and failover benefits that clustering enables, you must adhere to certain guidelines for cluster configuration.

Relationship Between Coherence and WebLogic Server Clusters

Similar to WebLogic Server clusters, Coherence clusters consist of multiple managed Coherence server instances that work together to distribute data in-memory to increase application scalability, availability, and performance. However, Coherence clusters use different clustering protocols and are configured separately from WebLogic Server clusters.

With Coherence clusters, a client interacts with the data in a local cache, and the distribution and backup of the data is automatically performed across cluster members.

A WebLogic Server domain can contain a single Coherence cluster. Multiple WebLogic Server clusters can be associated with a Coherence cluster.

For details on configuring and managing Coherence clusters, see *Configuring and Managing Coherence Clusters* in *Configuring and Managing Coherence Clusters*.

Benefits of Clustering

Clustering provides two key benefits: scalability and high availability. A WebLogic Server cluster provides the following benefits:

- **Scalability**
The capacity of an application deployed on a WebLogic Server cluster can be increased dynamically to meet demand. You can add server instances to a cluster without interruption of service—the application continues to run without impact to clients and end users.
- **High availability**
In a WebLogic Server cluster, application processing can continue when a server instance fails. You *cluster* application components by deploying them on multiple server instances in the cluster—so, if a server instance on which a component is running fails, then another server instance on which that component is deployed can continue application processing.

Key Capabilities of Clusters

A WebLogic cluster has three key capabilities that enable its primary benefits: failover, migration, and load balancing. The key clustering capabilities that enable scalability and high availability include the following:

- **Application failover**
If an application component that is doing a particular set of operations becomes unavailable for any reason, then a copy of the failed application component finishes those operations.
- **Migration**
WebLogic Server supports automatic and manual migration of a clustered server instance from one machine to another. A Managed Server that can be migrated is

referred to as a migratable server. This feature is designed for environments with requirements for high availability.

- Load balancing

Load balancing is the even distribution of jobs and associated communications across the computing and networking resources in your environment.

Objects That Can Be Clustered

A clustered application or application component is one that is available on multiple WebLogic Server instances in a cluster. Knowing what objects can be clustered is key to understanding how objects should be deployed throughout your domain. If an object is clustered, failover and load balancing for that object is available. To simplify cluster administration, maintenance, and troubleshooting, make sure that you deploy objects homogeneously; that is, to every server instance in your cluster.

Web applications can consist of different types of objects, including Enterprise Java Beans (EJBs), servlets, and Java Server Pages (JSPs). Each object type has a unique set of behaviors related to control, invocation, and how it functions within an application. For this reason, the methods that WebLogic Server uses to support clustering—and hence to provide load balancing and failover—can vary for different types of objects. The following types of objects can be clustered in a WebLogic Server deployment:

- Servlets
- JSPs
- EJBs
- Remote Method Invocation (RMI) objects
- Java Messaging Service (JMS) destinations
- Coherence cluster and managed Coherence servers
- Timer services
- Batch applications

About Dynamic Clusters

Dynamic clusters consist of server instances that can be dynamically scaled up to meet the resource needs of your application. A dynamic cluster uses a single server template to define configuration for a specified number of generated (dynamic) server instances.

When you create a dynamic cluster, the dynamic servers are preconfigured and automatically generated for you, enabling you to easily scale up the number of server instances in your dynamic cluster when you need additional server capacity. You can simply start the dynamic servers without having to first manually configure and add them to the cluster.

If you need additional server instances on top of the number you originally specified, you can increase the maximum number of servers instances (dynamic) in the dynamic cluster configuration or manually add configured server instances to the dynamic cluster. A dynamic cluster that contains both dynamic and configured server instances is called a mixed cluster.

The following table defines terminology associated with dynamic clusters:

Term	Definition
dynamic cluster	A cluster that contains one or more generated (dynamic) server instances that are based on a single shared server template.
configured cluster	A cluster in which you manually configure and add each server instance.
dynamic server	A server instance that is generated by WebLogic Server when creating a dynamic cluster. Configuration is based on a shared server template.
configured server	A server instance for which you manually configure attributes.
mixed cluster	A cluster that contains both dynamic and configured server instances.
server template	A prototype server definition that contains common, non-default settings and attributes that can be assigned to a set of server instances, which then inherit the template configuration. For dynamic clusters, the server template is used to generate the dynamic servers. See Server Templates in <i>Understanding Domain Configuration for Oracle WebLogic Server</i> .

For more information about dynamic clusters, see Dynamic Clusters in *Administering Clusters for Oracle WebLogic Server*.

Roadmap for Clustering in WebLogic Server

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to configure and manage WebLogic clusters.

Table 5-1 Roadmap for Clustering in WebLogic Server

Major Task	Subtasks and Additional Information
Learning more about WebLogic Server clustering	<ul style="list-style-type: none"> Clustering servlets and JSPs Clustering EJBs and RMI objects JMS and clustering Coherence clustering Dynamic clusters
Configuring a cluster	<ul style="list-style-type: none"> Understanding cluster configuration Communications in a cluster Cluster architectures Setting up WebLogic Server clusters Clustering best practices Setting up Coherence clusters
Configuring elasticity for a dynamic cluster	<ul style="list-style-type: none"> Performing on-demand scaling Configuring elastic actions Configuring calendar-based scaling Configuring policy-based scaling
Learning more about load balancing and failover in a cluster	<ul style="list-style-type: none"> Load balancing in a cluster Failover and replication in a cluster Configuring BIG-IP hardware with clusters Configuring F5 load balancers for MAN/WAN failover Configuring Radware load balancers for MAN/WAN failover

Table 5-1 (Cont.) Roadmap for Clustering in WebLogic Server

Major Task	Subtasks and Additional Information
Migrating servers and services in a cluster	<ul style="list-style-type: none">• Whole server migration• Service migration
Troubleshooting	<ul style="list-style-type: none">• Troubleshooting common problems• Troubleshooting multicast configuration
Reference	<ul style="list-style-type: none">• The WebLogic cluster API

6

Developing Applications in WebLogic Server

WebLogic Server implements Java Platform, Enterprise Edition (Java EE) Version 8.0 technologies. Java EE is the standard platform for developing multitier enterprise applications based on the Java programming language.

This chapter includes the following topics:

WebLogic Server and the Java EE Platform

With Java EE, development of Java enterprise applications has never been easier or faster. The aim of the Java EE platform is to provide developers with a powerful set of APIs while shortening development time, reducing application complexity, and improving application performance. The technologies that make up Java EE were developed collaboratively by several software vendors. For background information on Java EE application development, refer to the Java EE Tutorial at: <https://javaee.github.io/tutorial/toc.html>.

An important aspect of the Java EE programming model is the introduction of metadata annotations. Annotations simplify the application development process by allowing a developer to specify within the Java class itself how the application component behaves in the container, requests for dependency injection, and so on. Annotations are an alternative to deployment descriptors that were required by older versions of enterprise applications (Java EE 1.4 and earlier).

Starting in Java EE 5 and continuing in Java EE 8, the focus has been ease of development. There is less code to write – much of the boilerplate code has been removed, defaults are used whenever possible, and annotations are used extensively to reduce the need for deployment descriptors.

- EJB 3.2 provides simplified programming and packaging model changes. The mandatory use of Java interfaces from previous versions has been removed, allowing plain old Java objects to be annotated and used as EJB components. The simplification is further enhanced through the ability to place EJB modules directly inside web applications, removing the need to produce archives to store the web and EJB components and combine them together in an EAR file.
- Java EE 8 continues the focus on modern web applications of Java EE 7 and broadening the range of such applications. The key goals of the Java EE 8 platform are to modernize the infrastructure for enterprise Java for the cloud and microservices environments, emphasize HTML5 and HTTP/2 support, and enhance the ease of development through new Contexts and Dependency Injection (CDI) features, and further enhance the security and reliability of the platform.
- Constructing web applications is made easier with JavaServer Faces (JSF) technology and the JSP Standard Tag Library (JSTL). Java EE 8 supports rich thin-client technologies such as AJAX, for building applications for Web 2.0.

WebLogic Server Java EE applications are based on standardized, modular components. WebLogic Server provides a complete set of services for those modules and handles many details of application behavior automatically, without requiring programming. Java EE defines module behaviors and packaging in a generic, portable way, postponing runtime configuration until the module is actually deployed on an application server.

Java EE includes deployment specifications for web applications, EJB modules, web services, enterprise applications, client applications, and connectors. Java EE does not specify *how* an application is deployed on the target server—only how a standard module or application is packaged. For each module type, the specifications define the files required and their location in the directory structure.

Java is platform independent, so you can edit and compile code on any platform, and test your applications on development WebLogic Servers running on other platforms. For example, it is common to develop WebLogic Server applications on a PC running Windows or Linux, regardless of the platform where the application is ultimately deployed.

For more information, refer to the Java EE specification at: <https://www.oracle.com/technetwork/java/javaee/overview/index-jsp-135147.html>.

Overview of Java EE Applications and Modules

At runtime, a Java EE application is a type of module. A WebLogic Server Java EE application consists of one of the following modules or applications running on WebLogic Server:

- Web application modules—HTML pages, servlets, JavaServer Pages, and related files. See Web Application Modules in *Developing Applications for Oracle WebLogic Server*.
- Enterprise Java Beans (EJB) modules—entity beans, session beans, and message-driven beans. See Enterprise JavaBean Modules in *Developing Applications for Oracle WebLogic Server*.
- Connector modules—resource adapters. See Connector Modules in *Developing Applications for Oracle WebLogic Server*.
- Enterprise applications—web application modules, EJB modules, resource adapters and web services packaged into an application. See Enterprise Applications in *Developing Applications for Oracle WebLogic Server*.
- Web services—See WebLogic Web Services in *Developing Applications for Oracle WebLogic Server*.

A WebLogic application can also include the following WebLogic-specific modules:

- JDBC and JMS modules—See JMS and JDBC Modules in *Developing Applications for Oracle WebLogic Server*.
- Coherence Grid modules—See Packaging Coherence Applications in *Developing Oracle Coherence Applications for Oracle WebLogic Server*.
- WebLogic Diagnostic FrameWork (WLDF) modules—See WebLogic Diagnostic Framework Modules in *Developing Applications for Oracle WebLogic Server*.

Roadmap for Developing Applications in WebLogic Server

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to develop Java EE applications on WebLogic Server.

Table 6-1 Roadmap for Developing Applications in WebLogic Server

Major Task	Subtasks and Additional Information
Learning more about application development	<ul style="list-style-type: none"> • XML deployment descriptors • Deployment plans • Best practices for developing WebLogic Server applications • Understanding application life cycle events • Understanding production redeployment • Understanding WebLogic Server application classloading • Overview of shared Java EE libraries and optional packages
Setting up your development environment	<ul style="list-style-type: none"> • Starting and stopping WebLogic Server • Use the "split development directory" to develop your applications
Designing your application	<ul style="list-style-type: none"> • Using shared Java EE libraries and optional packages to share code among deployed applications • Programming JSF and JSTL applications • Using life cycle listeners • Using the HTTP publish-subscribe server • Using Coherence to cache data • Using Coherence to cache HTTP session data • <i>Developing Applications with the WebLogic Security Service</i> • Internationalize or localize your application • Using threads in WebLogic Server • Using WebSockets in WebLogic Server • <i>Adding WebLogic Logging Services to Applications Deployed on Oracle WebLogic Server</i> • <i>Developing Standalone Clients for Oracle WebLogic Server</i> • Designing manageable applications
Building your application	<ul style="list-style-type: none"> • <i>Developing Applications for Oracle WebLogic Server</i> • Deploying your "split development directory" application on WebLogic Server • Using Ant tasks to compile Java code
Using development tools	<ul style="list-style-type: none"> • Development software • Ant • <i>Oracle WebLogic Server Administration Console Online Help</i> • <i>Command Reference for Oracle WebLogic Server</i> • <i>Creating WebLogic Domains Using the Configuration Wizard</i> • <i>Creating Domain Templates Using the Domain Template Builder</i> • <i>Understanding the WebLogic Scripting Tool</i>
Moving your application to a production environment	<ul style="list-style-type: none"> • Preparing your application or module for deployment • Configuring your application for production deployment • Updating your deployed application (production redeployment)
Application examples	<ul style="list-style-type: none"> • Java EE 8 Examples • Java EE 7 Examples • Additional API Examples • Avitek Medical Records <p>A complete and functional Java EE application including source code. The MedRec (Spring) sample application demonstrates Spring 3.0.x application development practices.</p>

Table 6-1 (Cont.) Roadmap for Developing Applications in WebLogic Server

Major Task	Subtasks and Additional Information
Java EE API programming guides	<ul style="list-style-type: none"> • <i>Developing Custom Management Utilities Using JMX for Oracle WebLogic Server</i> • <i>Developing Manageable Applications Using JMX for Oracle WebLogic Server</i> • <i>Developing Security Providers for Oracle WebLogic Server</i> • <i>Solution Guide for Oracle TopLink</i> • <i>Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server</i> • <i>Developing Java EE Management Applications for Oracle WebLogic Server</i> • <i>Developing Enterprise JavaBeans for Oracle WebLogic Server</i> • <i>Developing JDBC Applications for Oracle WebLogic Server</i> • <i>Developing JMS Applications for Oracle WebLogic Server</i> • <i>Developing JNDI Applications for Oracle WebLogic Server</i> • <i>Developing JTA Applications for Oracle WebLogic Server</i> • <i>Developing Resource Adapters for Oracle WebLogic Server</i> • <i>Developing RMI Applications for Oracle WebLogic Server</i> • <i>Developing XML Applications for Oracle WebLogic Server</i> • <i>Developing Standalone Clients for Oracle WebLogic Server</i> • <i>Deploying Applications with the WebLogic Deployment API</i> • <i>Developing JSP Tag Extensions for Oracle WebLogic Server</i> • <i>Developing Applications with the WebLogic Security Service</i> • <i>Developing JAX-WS Web Services for Oracle WebLogic Server</i> • <i>Developing CommonJ Applications for Oracle WebLogic Server</i> • <i>Adding WebLogic Logging Services to Applications Deployed on Oracle WebLogic Server</i> • <i>Administering Clusters for Oracle WebLogic Server</i> • <i>Developing Oracle WebLogic Tuxedo Connector Applications for Oracle WebLogic Server</i>
Javadoc and API reference	<ul style="list-style-type: none"> • Java Platform, Enterprise Edition (Java EE) Version 8 • Java Platform, Standard Edition (Java SE) Version 8 • Java Platform, Standard Edition (Java SE) Version 11 • <i>JMS C API Reference for Oracle WebLogic Server</i> • <i>Java API Reference for Oracle WebLogic Server</i> • <i>Microsoft .NET Messaging API for Oracle WebLogic Server</i>
General reference	<ul style="list-style-type: none"> • XML deployment descriptors • WebLogic JSP cache, process, and repeat tags • WebLogic JSP form validation tags • <i>Command Reference for Oracle WebLogic Server</i> • <i>MBean Reference for Oracle WebLogic Server</i> • <i>WebLogic Server Error Message Catalog</i>

7

Deploying Applications in WebLogic Server

Application deployment refers to the process of making an application or module available for processing client requests in a WebLogic Server domain.

This chapter includes the following topics:

Overview of the Deployment Process

Application deployment encompasses several discrete tasks, such as preparing and configuring applications for deployment, exporting and redeploying applications to new environments, and managing deployed applications.

See the following topics in *Deploying Applications to Oracle WebLogic Server*:

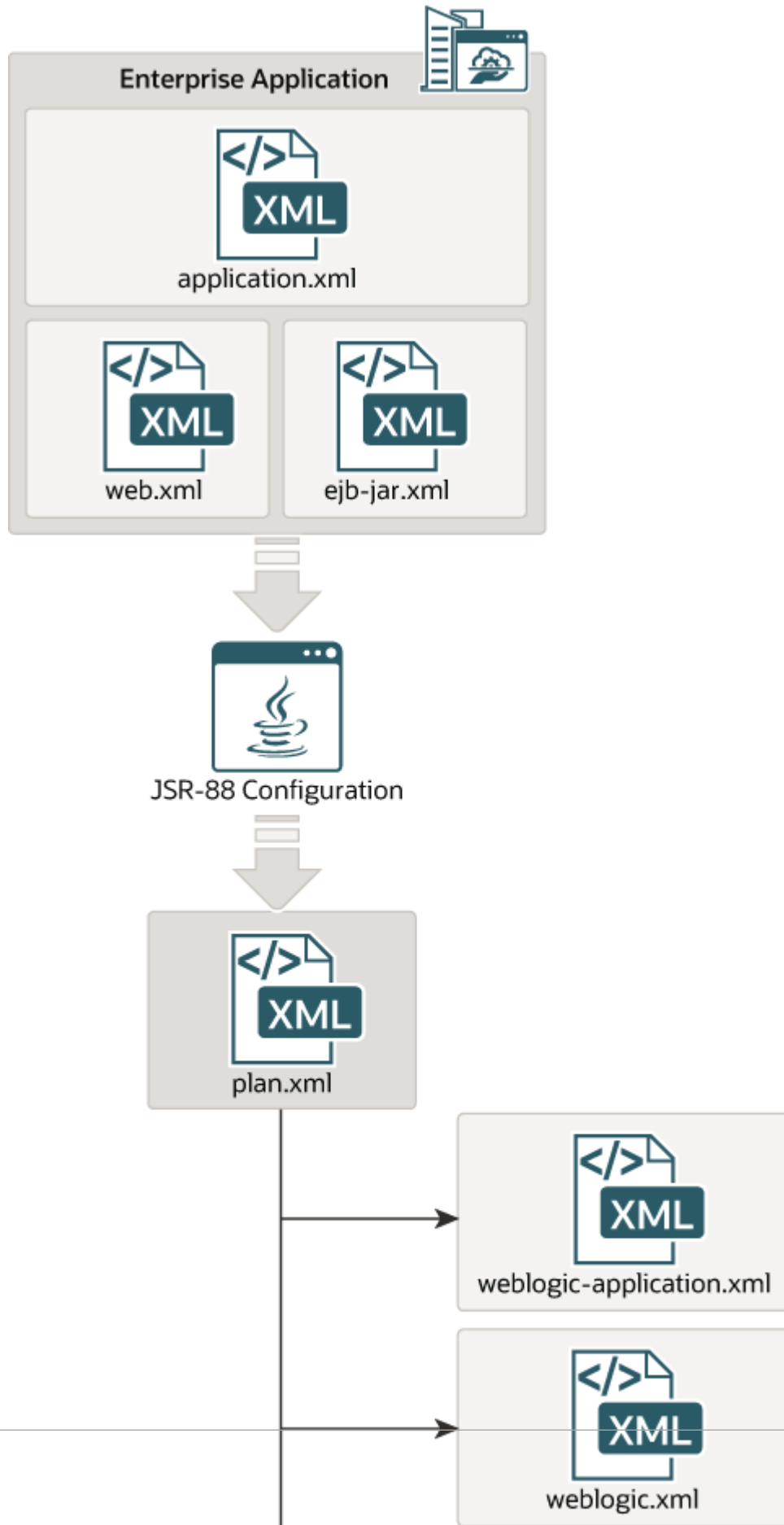
- Preparing Applications and Modules for Deployment
- Configuring Applications for Production Deployment
- Exporting an Application for Deployment to New Environments
- Deploying Applications and Modules with `weblogic.Deployer`
- Redeploying Applications in a Production Environment
- Managing Deployed Applications

Java EE Deployment Implementation

WebLogic Server implements the Java EE specification which includes a deployment specification, JSR-88, that describes a standard API used by deployment tools and application server providers to configure and deploy applications to an application server.

WebLogic Server implements both the JSR-88 Service Provider Interface (SPI) plug-in and model plug-in to comply with the Java EE deployment specification. You can use a basic Java EE deployment API deployment tool with the WebLogic Server plug-ins (without using WebLogic Server extensions to the API) to configure, deploy, and redeploy Java EE applications and modules to WebLogic Server. The WebLogic Server configuration generated by a Java EE deployment API configuration process is stored in a deployment plan and one or more generated WebLogic Server deployment descriptor files, as shown in [Figure 7-1](#).

Figure 7-1 Configuring Applications with the Java EE Deployment API



WebLogic Server deployment descriptors are generated as needed to store WebLogic Server configuration data.

The WebLogic Server deployment plan generated by a Java EE deployment API deployment tool identifies the WebLogic Server deployment descriptors that were generated for the application during the configuration session.

Although the Java EE deployment API provides a simple, standardized way to configure applications and modules for use with a Java EE-compliant application server, the specification does not address many deployment features that were available in previous WebLogic Server releases. For this reason, WebLogic Server provides important extensions to the Java EE deployment API specification to support capabilities described in WebLogic Server Deployment Features in *Deploying Applications to Oracle WebLogic Server*.

Fast Track Deployment Guide

To support application development environments, WebLogic Server provides a robust set of utilities and tools that you can use to deploy your applications quickly. The following topics provide basic instructions for quickly deploying Java EE applications and modules, JSP and HTML files, and Coherence modules. They also provide pointers to tools for system administrators. These deployment procedures are recommended for use in development environments only; the procedures are not recommended for use in production environments. For additional information about developing and deploying applications on WebLogic Server, see *Developing Applications for Oracle WebLogic Server* and *Deploying Applications to Oracle WebLogic Server*.

Complete *Installing and Configuring Oracle WebLogic Server and Coherence* before using the Fast Track procedures described in the following sections:

Java EE Deployment

To deploy a Java EE application or module:

1. Make sure that the Java EE application or module does not require additional resources such as named JDBC data sources or JMS queues. If the application requires external resources, you must configure them in the target WebLogic Server domain before deploying the application.
2. Copy the archive file or exploded archive directory for the Java EE application or module into the `/autodeploy` directory of the examples server domain directory, `ORACLE_HOME/user_projects/domains/wl_server/autodeploy`.
3. Start the Examples WebLogic Server instance.
4. Access the application using either a Java client or the configured URI for the application.

Auto-Deployment

When running in development mode, WebLogic Server automatically deploys applications copied into the `/autodeploy` subdirectory of the domain directory. Auto-deployment is a simple and quick method of deploying an application for testing or evaluation. See *Auto-Deploying Applications in Development Domains* in *Deploying Applications to Oracle WebLogic Server*.

Deploying Multiple Applications

When you use the WebLogic Server Administration Console to deploy multiple applications, upon installing the applications, they are listed in the Console's Deployments page in the "distribute Initializing" state. After activating changes, they are listed in the "Prepared" state. To deploy the applications, select the application names on the Deployments page and click **Start**.

System Administrator Tools

System Administrators can use the following tools to get started:

- WebLogic Server Administration Console

The WebLogic Server Administration Console is a browser-based web application that allows you to configure and monitor your WebLogic Server domain, server instances, and running applications and their associated resources. You can also use the WebLogic Server Administration Console to create new server instances and clusters and tune application descriptors. See *Oracle WebLogic Server Administration Console Online Help*.

After you log into the Console using the credentials you provided during installation, click the **Help** button or **How do I ...?** links for additional information.

- Configuration Wizard

Use the WebLogic Server Configuration Wizard to create new domains, and to create templates for automating domain configuration. See *Creating WebLogic Domains Using the Configuration Wizard*.

JSP/HTML Deployment

To deploy a simple JSP or HTML file:

1. Make sure your JSP file does not reference a tag library or other external resources—such resources require additional deployment steps that are beyond the scope of these Fast Track procedures. HTML files do not have this restriction.
2. Copy your JSP or HTML file into the `EXAMPLES_HOME/wl_server/examples/build/mainWebApp` directory, where `EXAMPLES_HOME` represents the directory in which the WebLogic Server code examples are configured. By default, this directory is `ORACLE_HOME/wlserver/samples/server`.
3. Start the Examples WebLogic Server instance.
4. In a web browser, request the JSP or HTML file using the following URL:

```
http://localhost:port/myFile
```

where:

`localhost` is the host name of the machine running WebLogic Server.

`port` is the port number where WebLogic Server is listening for requests (7001 by default).

`myFile` is the full name, including the `.jsp` or `.html` extension, of the JSP or HTML file you copied in step 2.

The JSP or HTML file has been automatically deployed from a directory preconfigured to target the Examples Server. `mainWebApp` is deployed by default and you can place your own JSP and HTML files into the `mainWebApp` exploded directory in order to quickly view or test them.

Coherence Deployment

WebLogic Server supports the deployment of Coherence applications that are packaged as Grid ARchive (GAR) modules. GAR modules contain the artifacts that are required for a Coherence application. GAR modules are deployed as standalone modules, packaged within enterprise applications, and as shared libraries. For details on packaging and deploying Coherence applications, see *Deploying Coherence Applications in Developing Oracle Coherence Applications for Oracle WebLogic Server*.

Roadmap for Deploying Applications in WebLogic Server

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to deploy applications in the WebLogic Server environment.

Table 7-1 Roadmap for Deploying Applications in WebLogic Server

Major Task	Subtasks and Additional Information
Learning more about application deployment	<ul style="list-style-type: none"> • Deployment terminology • Java EE deployment implementation • WebLogic Server deployment features • Understanding the deployment configuration process • Overview of the export process • Best practices for deploying applications
Packaging applications	<ul style="list-style-type: none"> • Preparing applications and modules for deployment • Archive file and exploded archive deployments • Using the <code>wlpackage</code> Ant task • Preparing Coherence applications for deployment
Using deployment tools	<ul style="list-style-type: none"> • Overview of deployment tasks • <code>weblogic.Deployer</code> utility • WebLogic.Plan generator command-line reference • WebLogic Maven plug-in for deployment • <code>wldeploy</code> Ant task
Advanced topics	<ul style="list-style-type: none"> • Overview of common deployment scenarios • Configuring applications for deployment • Redeploying a production application • <i>Deploying Applications with the WebLogic Deployment API</i> • Exporting an application for deployment to new environments • Distributing an application to a production environment • Changing the deployment order • Taking an application offline • Managing deployed applications
Reference	<ul style="list-style-type: none"> • Understanding the WebLogic deployment API

8

WebLogic Server Data Sources

In WebLogic Server, you can configure database connectivity by configuring Java Database Connectivity (JDBC) data sources and then targeting or deploying those resources to servers or clusters in your WebLogic domain.

This chapter includes the following topics:

Understanding JDBC Data Sources

A data source is a pool of database connections that are created when the data source instance is created, which can occur when the data source is deployed, when it is targeted, or when the host WebLogic Server instance is started.

Oracle WebLogic Server provides five types of data sources:

- Generic data sources—Generic data sources and their connection pools provide connection management processes that help keep your system running efficiently. You can set options in the data source to suit your applications and your environment.
- Active GridLink data sources—An event-based data source that adaptively responds to state changes in an Oracle RAC instance.
- Multi data sources—An abstraction around a group of generic data sources that provides load balancing or failover processing.
- Proxy data sources—Data sources that provide the ability to switch between databases in a WebLogic Server Multitenant environment.
- Universal Connection Pool (UCP) data sources—Data sources provided as an option for users who wish to use Oracle Universal Connection Pooling (UCP) to connect to Oracle Databases. UCP provides an alternative connection pooling technology to Oracle WebLogic Server connection pooling.

WebLogic Server also supports Java EE `DataSource` objects, which can be programmatically defined for a more flexible and portable method of database connectivity. For more information on Java EE `DataSource` objects, see *Using DataSource Resource Definitions in Developing JDBC Applications for Oracle WebLogic Server*.

Understanding Generic Data Sources

Generic data sources and their connection pools provide database access and database connection management processes that help keep your system running efficiently. Each generic data source contains a pool of database connections that are created when the data source is created and at server startup. Applications reserve a database connection from the data source by looking up the data source on the JNDI tree or in the local application context and then calling `getConnection()`. When finished with the connection, the application should call `connection.close()` as early as possible, which returns the database connection to the pool for other applications to use.

Understanding Active GridLink Data Sources

A single Active GridLink (AGL) data source provides connectivity between WebLogic Server and an Oracle Database service, which may include multiple Oracle RAC clusters. An AGL data source uses the Oracle Notification Service (ONS) to adaptively respond to state changes in an Oracle RAC instance. An Oracle Database service represents a workload with common attributes that enables administrators to manage the workload as a single entity. You scale the number of AGL data sources as the number of services increases in the database, independent of the number of nodes in the cluster.

An AGL data source includes the features of generic data sources plus the following support for Oracle RAC as described in *Administering JDBC Data Sources for Oracle WebLogic Server*:

- Fast Connection Failover
- Runtime Connection Load Balancing
- Generic Data Source Handling for Oracle RAC Outages
- GridLink Affinity
- SCAN Addresses
- Secure Communication using Oracle Wallet

Understanding JDBC Multi Data Sources

Conceptually, a multi data source can be regarded as a pool of generic data sources. Multi data sources are best used for failover or load balancing between nodes of a highly available database system, such as redundant databases or Oracle Real Application Clusters (Oracle RAC). A multi data source is bound to the JNDI tree or local application context, in the same way that generic data sources are bound to the JNDI tree. Applications look up a multi data source on the JNDI tree or in the local application context (`java:comp/env`), just as they do for data sources, and then request a database connection. The multi data source determines the data source to use that can satisfy the request depending upon the algorithm selected in the multi data source configuration: load balancing or failover.

Understanding Universal Connection Pool Data Sources

A Universal Connection Pool (UCP) data source enables the use of Oracle Universal Connection Pooling (UCP) for connecting to Oracle Database. A UCP data source is available as an option for using UCP, which is an alternative connection pooling technology to WebLogic Server connection pooling.

Note:

Oracle generally recommends the use of Active GridLink data sources, multi data sources, or generic data sources, and also the Oracle WebLogic Server connection pooling included in these data source implementations to establish connectivity with Oracle Database.

The implementations of UCP data sources are loosely coupled, allowing the swapping of the `ucp.jar` file to support the use of new UCP features by the applications. UCP data sources are not supported in an application-scoped, application-packaged, or standalone module environment. See *Using Universal Connection Pool Data Sources in Administering JDBC Data Sources for Oracle WebLogic Server*.

Roadmap for WebLogic Server Data Sources

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to configure and use WebLogic Server data sources.

Table 8-1 Roadmap for WebLogic Server Data Sources

Major Task	Subtasks and Additional Information
Learning more about WebLogic Server data source	<ul style="list-style-type: none"> About WebLogic JDBC resources JDBC resources JMX and WLST access for JDBC resources Overview of clustered JDBC resources Multi data source features Using WebLogic JDBC in an application
Configuring JDBC	<ul style="list-style-type: none"> Configuring JDBC data sources Using GridLink data sources Configuring JDBC multi data sources Advanced configuration for Oracle drivers JDBC data source transaction options Using roles and policies to secure JDBC data sources
Java EE DataSourcees	<ul style="list-style-type: none"> Using DataSource resource definitions
Managing JDBC	<ul style="list-style-type: none"> Managing data sources Monitoring data sources Monitoring GridLink JDBC resources
Performance and tuning	<ul style="list-style-type: none"> Tuning JDBC applications Tuning data source connection pools
Using WebLogic Server with Oracle RAC	<ul style="list-style-type: none"> Using WebLogic Server with Oracle RAC Using multi data sources with Oracle RAC Using fast connection failover with Oracle RAC
Using JDBC drivers	<ul style="list-style-type: none"> Overview of third-party JDBC drivers Derby <p>Derby is an all-Java DBMS product included in the WebLogic Server distribution that is intended solely to support demonstration of WebLogic Server examples. Documentation is not shipped with the product; it is available at http://db.apache.org/derby/manuals/index.html. For more information about Derby, see http://db.apache.org/derby.</p>

9

WebLogic Server Messaging

The Java Message Service (JMS) is a standard API for accessing enterprise messaging systems. JMS simplifies application development by providing a standard interface for creating, sending, and receiving messages.

This chapter includes the following topics:

Overview of JMS and WebLogic Server

The WebLogic Server implementation of JMS is an enterprise-class messaging system that is tightly integrated into the WebLogic Server platform and fully supports the JMS 2.0 Specification. The JMS 2.0 Specification is available at <http://www.oracle.com/technetwork/java/jms/index.html>. WebLogic JMS provides numerous [WebLogic JMS Extensions](#) that go beyond the standard JMS APIs.

Java Message Service

An enterprise messaging system enables applications to asynchronously communicate with one another through the exchange of messages. A message is a request, report, and/or event that contains information needed to coordinate communication between different applications. A message provides a level of abstraction, allowing you to separate the details about the destination system from the application code.

The Java Message Service (JMS) is a standard API for accessing enterprise messaging systems that is implemented by industry messaging providers. Specifically, JMS:

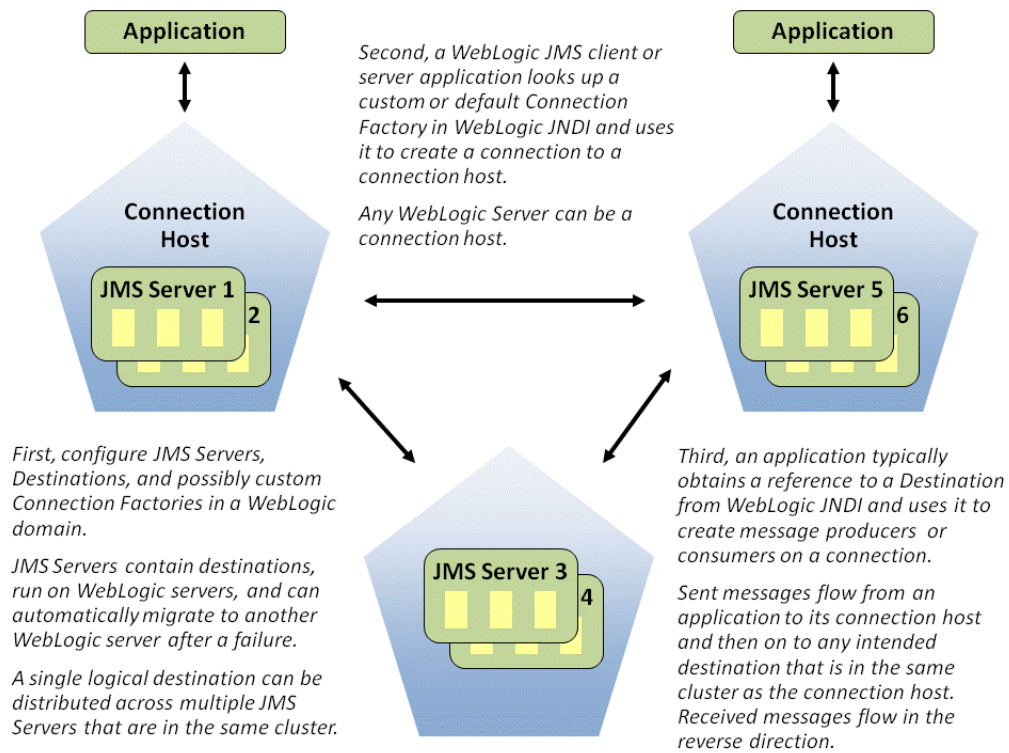
- Enables Java applications that share a messaging system to exchange messages
- Simplifies application development by providing a standard interface for creating, sending, and receiving messages

WebLogic JMS supports both client and server applications; in addition to Java, it has client libraries for C APIs and Microsoft .NET. WebLogic JMS accepts messages from *producer* applications and delivers them to *consumer* applications. For more information on JMS API programming with WebLogic Server, see *Developing JMS Applications for Oracle WebLogic Server*. For information about JMS API programming for WebLogic Server hosted consumer applications, see *Developing Message-Driven Beans for Oracle WebLogic Server*.

WebLogic JMS Message Workflow

The following figure illustrates the WebLogic message workflow.

Figure 9-1 WebLogic JMS Message Workflow



For information on the major components of the WebLogic JMS architecture, see WebLogic JMS Architecture in *Developing JMS Applications for Oracle WebLogic Server*.

Roadmap for WebLogic Server Messaging

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to develop and administer WebLogic JMS resources.

Table 9-1 Roadmap for WebLogic Server Messaging

Major Task	Subtasks and Additional Information
Learning more about WebLogic Server messaging	<ul style="list-style-type: none"> • WebLogic JMS architecture and environment • JMS configuration resources • Overview of JMS servers • Overview of JMS modules • Environment-related system resources for WebLogic JMS • Understanding the messaging models • Understanding the JMS API • Value-added public JMS API extensions

Table 9-1 (Cont.) Roadmap for WebLogic Server Messaging

Major Task	Subtasks and Additional Information
Getting started with WebLogic JMS	<ul style="list-style-type: none"> • Overview of JMS programming • Best practices for JMS beginners and advanced users • Developing a basic JMS application • Overview of JMS resource configuration • Value-added WebLogic Server JMS features • Integrating remote and foreign JMS providers • Sample Applications and Code Examples • Troubleshooting WebLogic JMS
Using new WebLogic JMS features	<ul style="list-style-type: none"> • Developing advanced pub/sub applications • Interoperating with Oracle advanced queueing • <i>Developing JMS .NET Client Applications for Oracle WebLogic Server</i>
Programming WebLogic messaging	<ul style="list-style-type: none"> • <i>Developing JMS Applications for Oracle WebLogic Server</i> • Developing advanced pub/sub applications • <i>Developing Message-Driven Beans for Oracle WebLogic Server</i>
Understanding clients for WebLogic messaging	<ul style="list-style-type: none"> • Understanding JMS clients • WebLogic Server client types and features
Configuring WebLogic messaging	<ul style="list-style-type: none"> • Best practices for JMS beginners and advanced users • <i>Administering JMS Resources for Oracle WebLogic Server</i> • Integrating remote and foreign JMS providers • <i>Administering the Store-and-Forward Service for Oracle WebLogic Server</i> • <i>Administering the WebLogic Messaging Bridge for Oracle WebLogic Server</i> • <i>Administering the WebLogic Persistent Store</i>
Using the WebLogic Server Administration Console to configure WebLogic messaging	<ul style="list-style-type: none"> • Configuring JMS servers • Configuring JMS system modules and resources • Configuring store-and-forward for JMS messages • Configuring and managing messaging bridges
Performance and tuning	<ul style="list-style-type: none"> • Tuning WebLogic JMS • Tuning WebLogic JMS store-and-forward • Tuning WebLogic messaging bridge • Tuning message-driven beans • Tuning logging last resource • Tuning the WebLogic Persistent Store
Reference	<ul style="list-style-type: none"> • Javadoc for WebLogic JMS extensions • MBean reference • JMS schema • Java Message Service Specification • <i>JMS C API Reference for Oracle WebLogic Server</i>

Understanding WebLogic Server Security

WebLogic Server includes a security architecture that provides a unique and secure foundation for applications. By taking advantage of the security features in WebLogic Server, enterprises benefit from a comprehensive, flexible security infrastructure designed to address the security challenges of making applications available on the web.

This chapter includes the following topics:

 **Note:**

For a complete checklist of critical tasks for locking down WebLogic Server, including specific tasks recommended for configuring a secure domain, securing the network, files, and databases used by WebLogic Server, see *Securing a Production Environment for Oracle WebLogic Server*.

JSR 375 Security API Support in WebLogic Server

WebLogic Server 14.1.1.0.0 provides full support for the JSR 375 Java EE Security API 1.0, a new security feature of Java EE 8.

The Java EE Security API defines portable authentication mechanisms (such as `HttpAuthenticationMechanism` and `IdentityStore`), and an access point for programmatic security using the `SecurityContext` interface. These portable, plug-in authentication and identity store interfaces provide an advantage over container-provided implementations because they allow the application to control the authentication process and the identity stores used for that authentication in a standard and portable way. Bundling the security configuration in the application, instead of configuring it externally, improves the management of the application's lifecycle, especially in a world of microservices that are distributed in containers. You can use the built-in implementations of these APIs, or define custom implementations.

In WebLogic Server, these authentication mechanisms are supported in the web container, and the `SecurityContext` interfaces are supported in the Servlet and EJB containers. WebLogic Server also supports the requirement in the Security API that group principal names are mapped to roles of the same name by default.

The Java EE Security API 1.0 (JSR 375) is defined in the specification at <https://www.jcp.org/en/jsr/detail?id=375?id=375>.

For more information about the Java EE Security API (JSR 375) in WebLogic Server, see Using the Java EE Security API in *Developing Applications with the WebLogic Security Service*.

Overview of the WebLogic Server Security Service

WebLogic Server includes a security architecture that provides a unique and secure foundation for applications that are available through the Internet. By taking advantage of the security features in WebLogic Server, enterprises benefit from a comprehensive, flexible security infrastructure designed to address the security challenges of making applications publicly available.

WebLogic security can be used standalone to secure WebLogic Server applications or as part of an enterprise-wide, security management system that represents a best-in-breed, security management solution. The key features of the WebLogic Security Service include:

- A comprehensive and standards-based design.
- End-to-end security for WebLogic Server-hosted applications, from the mainframe to the web browser.
- Legacy security schemes that integrate with WebLogic Server security, allowing companies to leverage existing investments.
- Security tools that are integrated into a flexible, unified system to ease security management across the enterprise.
- Easy customization of application security to business requirements through mapping of company business rules to security policies.
- A consistent model for applying security policies to Java EE and application-defined resources.
- Easy updates to security policies. This release includes usability enhancements to the process of creating security policies as well as additional expressions that control access to WebLogic resources.
- Easy adaptability for customized security solutions.
- A modularized architecture, so that security infrastructures can change over time to meet the requirements of a particular company.
- Support for configuring multiple security providers, as part of a transition scheme or upgrade path.
- A separation between security details and application infrastructure, making security easier to deploy, manage, maintain, and modify as requirements change.
- Default WebLogic security providers that provide you with a working security scheme out of the box. This release supports additional authentication stores such as databases and gives the option to configure an external RDBMS system as a datastore to be used by select security providers.
- Customization of security schemes using custom security providers.
- Unified management of security rules, security policies, and security providers through the WebLogic Server Administration Console.
- Support for standard Java EE security technologies such as the Java Authentication and Authorization Service (JAAS), Java Secure Sockets Extensions (JSSE), Java Cryptography Extensions (JCE), Java Authorization Contract for Containers (JACC), Java Authentication Service Provider Interface for Containers (JASPIC), and the Java EE Security API (JSR 375).

- A foundation for web services security including support for Security Assertion Markup Language (SAML) 1.1 and 2.0.
- Capabilities which allow WebLogic Server to participate in single sign-on (SSO) with web sites, web applications, and desktop clients
- A framework for managing public keys which includes a certificate lookup, verification, validation, and revocation as well as a certificate registry.

WebLogic Server Security Service Architecture

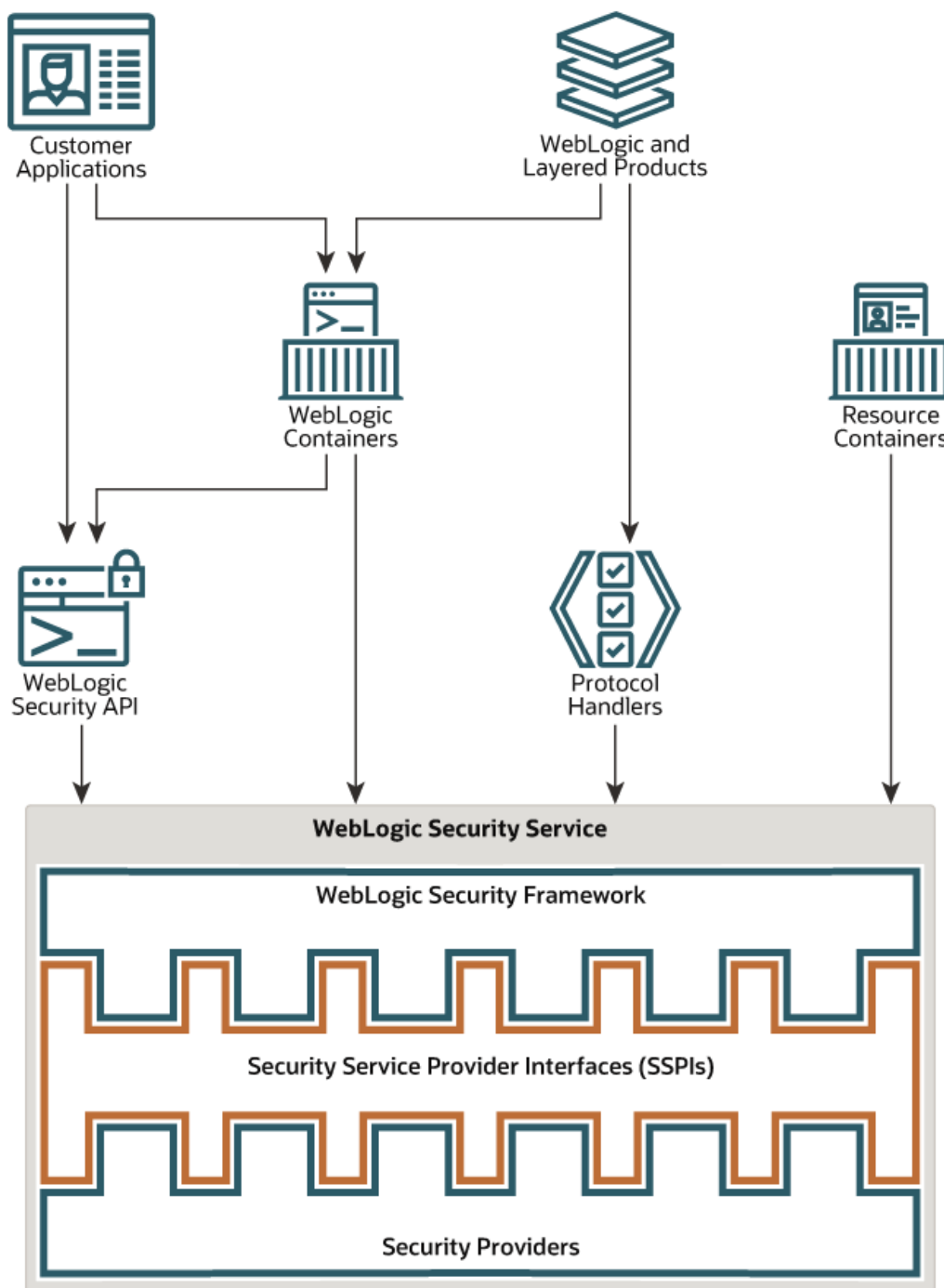
The WebLogic Server Security Service features a comprehensive and standards-based design that delivers end-to-end security for WebLogic Server-hosted applications from the mainframe to the web browser, easy customization of application security that can map of company business rules to security policies, a consistent model for applying security policies to Java EE and application-defined resources, and more.

This section provides a description of the architecture of the WebLogic Security Service. The architecture encompasses the following major components:

WebLogic Security Framework

[Figure 10-1](#) shows a high-level view of the WebLogic Security Framework. The framework comprises interfaces, classes, and exceptions in the `weblogic.security.service` package.

Figure 10-1 WebLogic Security Service Architecture



The primary function of the WebLogic Security Framework is to provide a simplified application programming interface (API) that can be used by security and application developers to define security services. Within that context, the WebLogic Security Framework also acts as an intermediary between the WebLogic containers (web and EJB), the resource containers, and the security providers.

Single Sign-on with the WebLogic Server Security Framework

Single sign-on (SSO) is the ability to require a user to sign on to an application only once and gain access to many different application components, even though these components may have their own authentication schemes. SSO enables users to log in securely to all their applications, web sites, and mainframe sessions with just one identity. The Security Assertion Markup Language (SAML) and Windows Integrated Authentication features provide web-based SSO functionality for WebLogic Server applications.

SAML Token Profile Support in WebLogic Web Services

The WebLogic web services and the WebLogic Security Framework support the generation, consumption, and validation of SAML 1.1 and 2.0 assertions. When using SAML assertions, a web service passes a SAML assertion and the accompanying proof material to the WebLogic Security Framework. If the SAML assertion is valid and trusted, the framework returns an authenticated Subject with a trusted principal back to the web service. WebLogic web services and the WebLogic Security Framework support the following SAML assertions:

- Sender-Vouches - The asserting party (different from the subject) vouches for the verification of the subject. The receiver must have a trust relationship with the asserting party.
- Holder-of-Key - The purpose of SAML token with "holder-of-key" subject confirmation is to allow the subject to use an X.509 certificate that may not be trusted by the receiver to protect the integrity of the request messages.

Conceptually, the asserting party inserts an X.509 public certificate (or other key info) into a SAML assertion. (More correctly, the asserting party binds a key to a subject.) In order to protect this embedded certificate, the SAML assertion itself must be signed by the asserting entity. For WebLogic Server, the web service client signs the SAML assertion with its private key. That is, the signature on the assertion is the signature of the SAML authority, and is not based on the certificate contained in, or identified by, the assertion.

- Bearer - The subject of the assertion is the bearer of the assertion, subject to optional constraints on confirmation using attributes that may be included in the `<SubjectConfirmationData>` element of the assertion.

The Security Service Provider Interfaces (SSPIs)

Security in WebLogic Server is based on a set of Security Service Provider Interfaces (SSPIs). The SSPIs can be used by developers and third-party vendors to develop security providers for the WebLogic Server environment. SSPIs are available for Adjudication, Auditing, Authentication, Authorization, Credential Mapping, Identity Assertion, Role Mapping, and Certificate Lookup and Validation.

The SSPIs allow customers to use custom security providers for securing WebLogic Server resources. Customers can use the SSPIs to develop custom security providers or they can purchase customer security providers from third-party vendors.

For more information on developing custom security providers, see *Developing Security Providers for Oracle WebLogic Server*.

WebLogic Security Providers

Security providers are modules that "plug into" a WebLogic Server security realm to provide security services to applications. They call into the WebLogic Security Framework on behalf of applications.

If the security providers supplied with the WebLogic Server product do not fully meet your security requirements, you can supplement or replace them with custom security providers. You develop a custom security provider by:

- Implementing the appropriate security service provider interfaces (SSPIs) from the `weblogic.security.spi` package to create runtime classes for the security provider.
- Creating an MBean Definition File (MDF) and using the WebLogic MBeanMaker utility to generate an MBean type, which is used to configure and manage the security provider.

See *Developing Security Providers for Oracle WebLogic Server*.

Managing WebLogic Server Security

When you manage WebLogic Server security, you focus primarily on two tasks: configuring security realms, and creating security policies.

Security Realms

A security realm comprises mechanisms for protecting WebLogic resources. Each security realm consists of a set of configured security providers, users, groups, security roles, and security policies. A user must be defined in a security realm to be able to access any WebLogic resources defined in that realm. When a user attempts to access a WebLogic resource, WebLogic Server authenticates the user, and then authorizes access by determining whether the user can be mapped to the security role that is defined in the security realm and designated in the security policy for that WebLogic resource.

Security Policies

A security policy answers the question, "Who has access to this WebLogic resource?" A security policy is an association between a WebLogic resource and one or more users, groups, or security roles that protects the WebLogic resource against unauthorized access. When creating a security policy, you can optionally define date and time constraints. A WebLogic resource has no protection until you assign it a security policy.

You assign security policies to any of the defined WebLogic resources (for example, an EJB resource or a JNDI resource) or to attributes or operations of a particular instance of a WebLogic resource (an EJB method or a servlet within a web application). If you assign a security policy to a type of WebLogic resource, all new instances of that resource inherit that security policy. Security policies assigned to individual resources or attributes override security policies assigned to a type of WebLogic resource.

Security for Coherence

Three key security features are available for use when you deploy Oracle Coherence within an Oracle WebLogic Server domain: Oracle Coherence access controllers, Oracle WebLogic Server authorization, and Oracle Coherence identity tokens.

Coherence is secured using both WebLogic Server security components and Coherence-specific security components. The components include:

- SSL for authentication between Coherence cluster members
- SSL for authentication between extend clients (external to WebLogic Server) and a Coherence cluster
- WebLogic Server policies and roles for authorizing Coherence services and caches
- Identity assertion between extend clients and Coherence clusters

For details about configuring Coherence security, see *Securing Coherence in WebLogic Server* in the *Securing Oracle Coherence*.

Roadmap for Securing WebLogic Server

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to use the WebLogic Security Service.

Table 10-1 Roadmap for Securing WebLogic Server

Major Task	Subtasks and Additional Information
Learning more about fundamental security concepts	<ul style="list-style-type: none">• Auditing• Authentication• Security Assertion Markup Language (SAML)• Single sign-on (SSO)• Authorization• Identity and trust• Secure Sockets Layer (SSL)• WebLogic security framework• Single sign-on with the WebLogic Server security framework• SAML token support in WebLogic web services• Security Service Provider Interfaces (SSPIs)• WebLogic security providers

Table 10-1 (Cont.) Roadmap for Securing WebLogic Server

Major Task	Subtasks and Additional Information
Administering WebLogic Server security	<ul style="list-style-type: none"> • Security management concepts • Customizing the default security configuration • Migrating security data • Managing the embedded LDAP server • Managing the RDBMS security store • Configuring keystores • Configuring SSL • Configuring cross-domain security • <i>Securing Resources Using Roles and Policies for Oracle WebLogic Server</i> • Exploring security options for cluster architectures • Configuring Security for Coherence
Authenticating users	<ul style="list-style-type: none"> • Authenticating users defined in an LDAP server • Authenticating against an RDBMS system • Authenticating against a Windows NT domain • Authenticating a remote user • Using SAML • Configuring single sign-on • Configuring single sign-on with Microsoft clients • Configuring single sign-on with web browsers and HTTP clients • Using Kerberos • Using multiple authentication providers • Configuring password composition rules • Managing users and groups • Using Java Authentication SPI for Containers (JASPIC) • Using the Java EE Security API
Configuring SSL	<ul style="list-style-type: none"> • Setting up SSL: main steps • Configuring keystores • Creating a keystore: example • X.509 certificate revocation checking
Configuring authorization	<ul style="list-style-type: none"> • <i>Securing WebLogic Resources Using Roles and Policies</i> • Configuring an authorization provider • Using multiple authorization providers • Using JAAS authorization • Configuring a role mapping provider • Using Java Authorization Contract Containers (JACC)
Learning more about security realms	<ul style="list-style-type: none"> • Introduction to security realms • Users • Groups • Security roles • Security policies • Security providers
Programming applications for security	<ul style="list-style-type: none"> • <i>Developing Applications with the WebLogic Security Service</i> • Configuring resource adapter security • WebLogic web service security topics

Table 10-1 (Cont.) Roadmap for Securing WebLogic Server

Major Task	Subtasks and Additional Information
Guidelines for locking down your WebLogic Server production environment	<ul style="list-style-type: none"><li data-bbox="646 338 1377 363">• <i>Securing a Production Environment for Oracle WebLogic Server</i>

WebLogic Server Web Services

A web service is a self-contained application that can be described, published, and invoked over a network, such as a corporate intranet or the Internet. Because you access web services using standard web protocols such as Extensible Markup Language (XML) and HTTP, the diverse and heterogeneous applications on the web (which typically already understand XML and HTTP) can access web services and communicate with each other automatically.

Major benefits of web services include:

- Interoperability among distributed applications that span diverse hardware and software platforms
- Easy, widespread access to applications through firewalls using web protocols
- A cross-platform, cross-language data model (XML) that facilitates developing heterogeneous distributed applications



Note:

The JAX-RPC API is deprecated in Oracle WebLogic Server as of version 12.2.1 and will be removed in a future release. Oracle recommends developing your web services using JAX-WS or JAX-RS.

This chapter includes the following topics:

Anatomy of a Web Service

Web services are characterized by three factors: what they do (the business functionality they expose), how they can be accessed (the set of published interfaces necessary to use the exposed functionality), and where they are (the web site which exposes that functionality).

What the web service can do is described in a standard XML vocabulary called Web Services Description Language (WSDL). For example, a banking web service may implement functions to check an account, print a statement, and deposit and withdraw funds. These functions are described in a WSDL file that any consumer can invoke to access the banking web service. As a result, a consumer does not have to know anything more about a web service than the WSDL file that describes what it can do.

A web service client (or consumer)--such as, a desktop application or a Java Platform, Enterprise Edition portlet-- invokes a web service by submitting a request in the form of an XML document to the web service. The web service processes the request and returns the result to the web service client in an XML document.

The web service client can send a request in the form of a Simple Object Access Protocol (SOAP) message. SOAP is an XML messaging framework designed to allow heterogeneous applications to exchange structured information in a distributed environment. In turn, the web service processes the request and returns the response in a SOAP message.

You can also develop Representational State Transfer (REST) web services, or "RESTful" web services. REST describes any simple interface that transmits data over a standardized interface (such as HTTP) without an additional messaging layer, such as SOAP. REST provides a set of design rules for creating stateless services that are viewed as resources, or sources of specific information, and can be identified by their unique URIs. A client accesses the resource using the URI, a standardized fixed set of methods, and a representation of the resource is returned. The client is said to transfer state with each new resource representation.

To secure the message exchange, the web service may require credentials to access the service, for example a username and a password, or encrypt the response.

Web Service Standards

Many specifications that define web service standards are written so as to allow for broad use of the specification throughout the industry. The implementation of a particular specification in Oracle WebLogic Server might not cover all possible usage scenarios covered by the specification. However, Oracle considers interoperability of web service platforms to be more important than providing support for all possible edge cases of the web service specifications.

Web services rely on a set of XML-based industry standards, including the following:

- XML, a data format that allows uniform communication between services consumers and services providers
- XML Schema, a framework that describes XML vocabularies used in business transactions
- SOAP, a protocol for exchanging structured information in the implementation of services
- WSDL, an XML-based language providing a model for describing services
- WS-Policy, a framework that provides a flexible and extensible grammar for describing the capabilities, requirements, and general characteristics of services using policies

Roadmap for Web Services

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to develop and manage WebLogic web services.

Table 11-1 Roadmap for Web Services

Major Task	Subtasks and Additional Information
Learning more about WebLogic web services	<ul style="list-style-type: none"> • Features and standards supported by WebLogic web services • Overview of WebLogic web services • Choose between JAX-WS and RESTful web service • Overview of web services security
Using the samples (for WebLogic web service developers)	<ul style="list-style-type: none"> • Sample application and code examples • JAX-WS web service example • Examples of developing JAX-WS web service clients

Table 11-1 (Cont.) Roadmap for Web Services

Major Task	Subtasks and Additional Information
Developing web services using JAX-WS	<ul style="list-style-type: none"> Starting from Java Starting from WSDL Programming the JWS file Using JAX binding Invoking a web service Invoking a web service asynchronously Using web services reliable messaging Managing web service persistence Configuring message buffering for web services Managing web services in a cluster Using web services atomic transactions Publishing a web service endpoint Using callbacks Optimizing binary data transmissions using MTOM/XOP Using XML catalogs Handling exceptions using SOAP Creating and using SOAP message handlers Programming stateful JAX-WS web services using HTTP session
Developing RESTful web services	<ul style="list-style-type: none"> Standards to use for RESTful web service development on WebLogic Server Learning about RESTful web service development Defining the root resource class Defining the relative URI of the root resource class Customizing request and response message types More advanced RESTful web service tasks Developing RESTful web service clients Packaging and Deploying RESTful web service applications Securing RESTful web services and clients Testing RESTful web services Monitoring RESTful web services Using server-sent events
Deploying and administering WebLogic web services	<ul style="list-style-type: none"> Packaging and deploying RESTful web services Developing JAX-WS web services
Securing WebLogic web services	<ul style="list-style-type: none"> <i>Securing WebLogic Web Services for Oracle WebLogic Server</i>

Enterprise JavaBeans (EJBs)

Enterprise JavaBeans (EJBs) are Java EE components that implement EJB technology. EJBs run in the EJB container, a runtime environment within Oracle WebLogic Server. Although transparent to the application developer, the EJB container provides system-level services, such as transactions and security, to the EJBs deployed in it. These services enable you to quickly build and deploy EJBs, which form the core of transactional Java EE applications.

This chapter includes the following topics:

Understanding EJBs

EJB 3.2 technology is the server-side component architecture for the development and deployment of component-based business applications. EJB technology enables rapid and simplified development of distributed, transactional, secure, and portable applications based on Java EE technology.

The EJB 3.2 specification provides simplified programming and packaging model changes. The mandatory use of Java interfaces from previous versions has been removed, allowing plain old Java objects to be annotated and used as EJB components. The simplification is further enhanced through the ability to place EJB modules directly inside of web applications, removing the need to produce archives to store the web and EJB components and combine them together in an EAR file.

This topic contains the following sections:

EJB Documentation in WebLogic Server

See the following documents to know about using EJBs with WebLogic Server:

- For instructions on how to program, package, and deploy 3.2 EJBs on WebLogic Server, see *Developing Enterprise JavaBeans for Oracle WebLogic Server*.
- For instructions on how to organize and build WebLogic Server EJBs in a split directory environment, see *Developing Applications for Oracle WebLogic Server*.
- For more information on programming and packaging 2.x EJBs, see *Developing Enterprise JavaBeans, Version 2.1, for Oracle WebLogic Server*.

Additional EJB Information

To learn more about EJB concepts, such as the benefits of enterprise beans, the types of enterprise beans, and their life cycles, then visit the following web sites:

- Enterprise JavaBeans 3.2 Specification (JSR-345) at <https://jcp.org/en/jsr/detail?id=345>
- The "Enterprise Beans" section in the Java EE 8 Tutorial at <https://javaee.github.io/tutorial/partentbeans.html#BNBLR>

Session EJBs Implement Business Logic

Session beans implement business logic. A session bean instance serves one client at a time. There are three types of session beans: stateful, stateless, and singleton.

For detailed information about the types of session beans and when to use them, see [What Is a Session Bean](#) in the Java EE 8 Tutorial.

Stateful Session Beans

Stateful session beans maintain state information that reflects the interaction between the bean and a particular client across methods and transactions. A stateful session bean can manage interactions between a client and other enterprise beans, or manage a workflow.

Example: A company web site that allows employees to view and update personal profile information could use a stateful session bean to call a variety of other beans to provide the services required by a user, after the user clicks "View my Data" on a page:

- Accept the login data from a JSP, and invoke another EJB that validates the login data.
- Send confirmation of authorization to the JSP.
- Call a bean that accesses profile information for the authorized user.

Stateless Session Beans

A stateless session bean does not store session or client state information between invocations—the only state it might contain is not specific to a client, for instance, a cached database connection or a reference to another EJB. At most, a stateless session bean may store state for the duration of a method invocation. When a method completes, state information is not retained.

Any instance of a stateless session bean can serve any client—any instance is equivalent. Stateless session beans can provide better performance than stateful session beans, because each stateless session bean instance can support multiple clients, albeit one at a time. The client of a stateless session bean can be a web service endpoint.

Example: An Internet application that allows visitors to click a "Contact Us" link and send an email could use a stateless session bean to generate the email, based on the "to" and "from" information gathered from the user by a JSP.

Singleton Session Beans

Singleton session beans provide a formal programming construct that guarantees a session bean will be instantiated once per application in a particular Java Virtual Machine (JVM), and that it will exist for the life cycle of the application. With singletons, you can easily share state between multiple instances of an enterprise bean component or between multiple enterprise bean components in the application.

Singleton session beans offer similar functionality to stateless session beans but differ from them in that there is only one singleton session bean per application, as opposed to a pool of stateless session beans, any of which may respond to a client request. Like stateless session beans, singleton session beans can implement web service

endpoints. Singleton session beans maintain their state between client invocations but are not required to maintain their state across server crashes or shutdowns.

Example: An Internet application that provides a central counter service. A stateless singleton bean can be called from a Java client, with the count being consistently incremented by 1 as the client is invoked multiple times.

Message-Driven Beans Implement Loosely Coupled Business Logic

A message-driven bean (MDB) implements loosely coupled or asynchronous business logic in which the response to a request need not be immediate. A message-driven bean receives messages from a JMS Queue or Topic, and performs business logic based on the message contents. It is an asynchronous interface between EJBs and JMS.

Throughout its life cycle, an MDB instance can process messages from multiple clients, although not simultaneously. It does not retain state for a specific client. All instances of a message-driven bean are equivalent—the EJB container can assign a message to any MDB instance. The container can pool these instances to allow streams of messages to be processed concurrently.

The EJB container interacts directly with a message-driven bean—creating bean instances and passing JMS messages to those instances as necessary. The container creates bean instances at deployment time, adding and removing instances during operation based on message traffic.

See *Developing Message-Driven Beans for Oracle WebLogic Server*.

Example: In an online shopping application, where the process of taking an order from a customer results in a process that issues a purchase order to a supplier, the supplier ordering process could be implemented by a message-driven bean. While taking the customer order always results in placing a supplier order, the steps are loosely coupled because it is not necessary to generate the supplier order before confirming the customer order. It is acceptable or beneficial for customer orders to "stack up" before the associated supplier orders are issued.

EJB Anatomy and Environment

To develop an EJB, you must provide files for the EJB class, the business interfaces, and the helper classes. The environment for the EJB includes an EJB container and, optionally, one or more deployment descriptor files.

The following sections briefly describe classes required for each bean type, the EJB runtime environment, and the deployment descriptor files that govern a bean's runtime behavior:

EJB Components

The composition of a bean varies by bean type. [Table 12-1](#) defines the classes that make up each type of EJB, and defines the purpose of the class type.

 **Note:**

The EJB 2.1 and earlier API required that Local and Remote clients access the stateful or stateless session bean by means of the session bean's local or remote home and the local or remote component interfaces. These interfaces remain available for use with EJB 3.x; however, the EJB 2.1 Remote and Local client view is not supported for singleton session beans.

For more information see *Create EJB Classes and Interfaces in Developing Enterprise JavaBeans, Version 2.1, for Oracle WebLogic Server*.

Table 12-1 Components of EJB 3.1

EJB Component	Description	Stateless Session	Stateful Session	Singleton Session	MDB
Remote business interface	The remote business interface exposes business logic to remote clients—clients running in a separate application from the EJB. It defines the business methods a remote client can call.	Yes	Yes	Yes	No
Local business interface	The local business interface exposes business logic to local clients—those running in the same application as the EJB. It defines the business methods a local client can call.	Yes	Yes	Yes	No
Local No-interface	The no-interface view is a variation of the Local view that exposes the public methods of the bean class without the use of a separate business interface.	Yes	Yes	Yes	Yes
Bean class	The bean class implements business logic.	Yes	Yes	Yes	Yes

The EJB Container

An EJB container is a runtime container for beans that are deployed to an application server. The container is automatically created when the application server starts up, and serves as an interface between a bean and runtime services such as:

- Life cycle management
- Code generation
- Security
- Transaction management
- Locking and concurrency control

Embeddable EJB Container

Unlike traditional Java EE server-based execution, embeddable usage allows client code and its corresponding enterprise beans to run within the same virtual machine and class loader. This provides better support for testing, offline processing (for example, batch jobs), and the use of the EJB programming model in desktop applications.

Most of the services present in the enterprise bean container in a Java EE server are available in the embedded enterprise bean container, including injection, container-managed transactions, and security. Enterprise bean components execute similarly in both embedded and Java EE environments, and therefore the same enterprise bean can be easily reused in both standalone and networked applications.

For more information about the Embedded Enterprise Bean Container, see Using an Embedded EJB Container in Oracle WebLogic Server in *Developing Enterprise JavaBeans for Oracle WebLogic Server*.

EJB Metadata Annotations

The WebLogic Server EJB 3.2 programming model uses the metadata annotations feature in which you create an annotated EJB 3.2 bean file, and then use the WebLogic compile tool `weblogic.appc` (or its Ant equivalent `wlappc`) to compile the bean file into a Java class file and generate the associated EJB artifacts, such as the required EJB interfaces and optional deployment descriptors.

See Programming the Annotated EJB Class in *Developing Enterprise JavaBeans for Oracle WebLogic Server*.

Optional EJB Deployment Descriptors

As of EJB 3.0, you are no longer required to create the EJB deployment descriptor files (such as `ejb-jar.xml`). However, you can still use XML deployment descriptors if you want. In the case of conflicts, the deployment descriptor value overrides the annotation value.

If you are continuing to use deployment descriptors in your EJB implementation, refer to EJB Deployment Descriptors in *Developing Enterprise JavaBeans, Version 2.1, for Oracle WebLogic Server*.

WebLogic Server EJB 2.x has three deployment descriptors:

- `ejb-jar.xml`—The standard Java EE deployment descriptor. All beans must be specified in an `ejb-jar.xml`. An `ejb-jar.xml` can specify multiple beans that will be deployed together.
- `weblogic-ejb-jar.xml`—WebLogic Server-specific deployment descriptor that contains elements related to WebLogic Server features such as clustering, caching, and transactions. This file is required if your beans take advantage of WebLogic Server-specific features. Similar to `ejb-jar.xml`, `weblogic-ejb-jar.xml` can specify multiple beans that will be deployed together.
- `weblogic-cmp-jar.xml`—WebLogic Server-specific deployment descriptor that contains elements related to container-managed persistence for entity beans. Entity beans that use container-managed persistence must be specified in a `weblogic-cmp-jar.xml` file.

For descriptions of the WebLogic Server EJB 2.x deployment descriptors, refer to Deployment Descriptor Schema and Document Type Definitions Reference in *Developing Enterprise JavaBeans, Version 2.1, for Oracle WebLogic Server*.

EJBs Clients and Communications

An EJB can be accessed by server-side or client-side objects such as servlets, Java client applications, other EJBs, web services, and non-Java clients. Any client of an EJB, whether in the same or a different application, accesses it in a similar fashion. WebLogic Server automatically creates implementations of an EJB's home and business interfaces that can function remotely, unless the bean has only a local interface.

This topic includes the following sections:

Accessing EJBs

Clients access enterprise beans either through a no-interface view or through a business interface. A no-interface view of an enterprise bean exposes the public methods of the enterprise bean implementation class to clients. Clients using the no-interface view of an enterprise bean may invoke any public methods in the enterprise bean implementation class or any superclasses of the implementation class. A business interface is a standard Java programming language interface that contains the business methods of the enterprise bean.

The client of an enterprise bean obtains a reference to an instance of an enterprise bean through either dependency injection, using Java programming language annotations, or JNDI lookup, using the Java Naming and Directory Interface syntax to find the enterprise bean instance.

Dependency injection is the simplest way of obtaining an enterprise bean reference. Clients that run within a Java EE server-managed environment, JavaServer Faces web applications, JAX-RS web services, other enterprise beans, or Java EE application clients, support dependency injection using the `javax.ejb.EJB` annotation.

Applications that run outside a Java EE server-managed environment, such as Java SE applications, must perform an explicit lookup. JNDI supports a global syntax for identifying Java EE components to simplify this explicit lookup. For more information see, *Programming Access to EJB Clients in Developing Enterprise JavaBeans for Oracle WebLogic Server*.

Because of network overhead, it is more efficient to access beans from a client on the same machine than from a remote client, and even more efficient if the client is in the same application.

For information on programming client access to an EJB, see [Accessing Enterprise Beans](#) in the Java EE 8 Tutorial.

EJB Communications

WebLogic Server EJBs use:

- T3—To communicate with remote objects. T3 is a WebLogic-proprietary remote network protocol that implements the Remote Method Invocation (RMI) protocol.
- RMI—To communicate with remote objects. RMI enables an application to obtain a reference to an object located elsewhere in the network, and to invoke methods on

that object as though it were co-located with the client on the same JVM locally in the client's virtual machine.

An EJB with a remote interface is an RMI object. An EJB's remote interface extends `java.rmi.remote`. For more information on WebLogic RMI, see *Developing RMI Applications for Oracle WebLogic Server*.

- HTTP—An EJB can obtain an HTTP connection to a web server external to the WebLogic Server environment by using the `java.net.URL` resource connection factory. See *Configuring EJBs to Send Requests to an URL in Developing Enterprise JavaBeans, Version 2.1, for Oracle WebLogic Server*.

You can specify the attributes of the network connection an EJB uses by binding the EJB to a WebLogic Server custom network channel. See *Configuring Network Resources in Administering Server Environments for Oracle WebLogic Server*.

Securing EJBs

By default, any user defined in the security realm can invoke the public methods of an EJB. Therefore, if you want to restrict access to EJBs, you can create security policies for them, or use security-related annotations, to specify the roles that are allowed to invoke all, or a subset, of its methods. For information about restricting access to EJBs using annotations within the EJBs themselves, see *Securing Access to the EJB in Developing Enterprise JavaBeans for Oracle WebLogic Server*.

In addition, you create security roles and map users to roles using the WebLogic Server Administration Console to update your security realm. For details, see [Manage security roles in Oracle WebLogic Server Administration Console Online Help](#).

For more information about security and EJBs:

- Security Fundamentals in *Understanding Security for Oracle WebLogic Server* has introductory information about authentication, authorization and other security topics.
- Securing Enterprise JavaBeans (EJBs) in *Developing Applications with the WebLogic Security Service* provides instructions for configuring authentication and authorization for EJBs.
- *Securing Resources Using Roles and Policies for Oracle WebLogic Server* contains instructions for on configuring authentication and authorization for your EJBs using the WebLogic Server Administration Console.

Roadmap for EJBs in WebLogic Server

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to develop, deploy, and secure EJBs in the WebLogic Server environment.

Table 12-2 Roadmap for EJBs in WebLogic Server

Major Task	Subtasks and Additional Information
Understanding EJBs	<ul style="list-style-type: none"> • New Features and Changes in EJB • What Is New and Changed in EJB 3.2

Table 12-2 (Cont.) Roadmap for EJBs in WebLogic Server

Major Task	Subtasks and Additional Information
Simple EJB examples	<ul style="list-style-type: none"> • Example of a simple stateless EJB • Example of a simple stateful EJB • Example of an interceptor class • Packaged EJB 3.2 Examples in WebLogic Server • Example of invoking an entity from a session bean
Iterative EJB developing	<ul style="list-style-type: none"> • Overview of the EJB development process • Creating a source directory • Programming access to EJB clients • Programming and configuring transactions • Programming the EJB interface • Programming the EJB timer service • Programming the annotated EJB class • Programming optional interceptors • Compiling Java source code • Optionally creating and editing deployment descriptors • Packaging EJBs • Deploying EJBs
Programming the annotated EJB class	<ul style="list-style-type: none"> • Overview of metadata annotations and EJB bean files • Programming the bean file: requirements and changes from 2.x • Programming the bean file: typical steps • Complete list of metadata annotations by function
Deployment guidelines for EJBs	<ul style="list-style-type: none"> • Before you deploy an EJB • Understanding and performing deployment tasks • Deployment guidelines for EJBs
Using an embedded EJB container in Oracle WebLogic Server	<ul style="list-style-type: none"> • Overview of the embeddable EJB container • EJB 3.1 Lite functionality supported in the embedded EJB container
Configuring the Persistence Provider in Oracle WebLogic Server	<ul style="list-style-type: none"> • Overview of Oracle TopLink • Specifying a persistence provider • Using Oracle TopLink in Oracle WebLogic Server
Reference	<ul style="list-style-type: none"> • EJB metadata annotations reference
Information about EJB 2.x	<ul style="list-style-type: none"> • <i>Developing Enterprise JavaBeans, Version 2.1, for Oracle WebLogic Server</i>

Monitoring, Diagnosing, and Troubleshooting

Oracle WebLogic Server provides several built-in management, diagnostic, and automation tools to increase management and operational efficiency of the server, the underlying JVM, deployed applications, and configured resources. These tools include the WebLogic Diagnostic Framework (WLDF), WebLogic logging services, integration with Oracle HotSpot, SNMP support, and more.

This chapter includes the following topics:

WebLogic Diagnostics Framework

The WebLogic Diagnostics Framework (WLDF) is a monitoring and diagnostic framework that defines and implements a set of services that run within WebLogic Server processes and participate in the standard server life cycle. Using WLDF, you can create, collect, analyze, archive, and access diagnostic data generated by a running server and the applications deployed within its containers.

The data you collect using WLDF provides insight into the runtime performance of servers and applications and enables you to isolate and diagnose faults when they occur.

WLDF includes several components for collecting and analyzing data:

- Integration with Oracle HotSpot—If WebLogic Server is configured with Oracle HotSpot VM, WebLogic Server events can optionally be propagated to the Java Flight Recorder, a performance monitoring and profiling tool. WebLogic Server provides specific integration points with Java Flight Recorder:
 - WebLogic Server events are propagated to Java Flight Recorder for inclusion in a common data set for runtime or post-incident analysis.
 - The flight recording data is also included in WLDF diagnostic image captures, enabling you to capture flight recording snapshots based on WLDF watch rules. This full set of functionality enables you to capture and analyze runtime system information for both the JVM and the Fusion Middleware components running on it, in a single view.
- Diagnostic Image Capture—Creates a diagnostic snapshot from the server that can be used for post-failure analysis. The diagnostic image capture includes Java Flight Recorder data, if it is available, that can be viewed in Java Mission Control.
- Built-in Diagnostic Modules—Provides a simple and easy-to-use mechanism for performing basic health and performance monitoring of a WebLogic Server instance. The built-in diagnostic modules collect data from key WebLogic Server runtime MBeans that monitor the main components of a server instance, such as the WebLogic Server runtime, JDBC, JMS, and Java EE containers hosting servlets and EJBs.
- Archive—Captures and persists data events, log records, and metrics from server instances and applications.
- Instrumentation—Adds diagnostic code to WebLogic Server instances and the applications running on them to execute diagnostic actions at specified locations in the code. The Instrumentation component provides the means for associating a diagnostic

context with requests so they can be tracked as they flow through the system. The WebLogic Server Administration Console includes a Request Performance page, which shows real-time and historical views of method performance information that has been captured through the WLDF instrumentation capabilities, serving as a tool that can help identify performance problems in applications.

- **Harvester**—Captures metrics from runtime MBeans, including WebLogic Server MBeans and custom MBeans, which can be archived and later accessed for viewing historical data.
- **Policies and Actions**—Provides the means for monitoring server and application states and sending notifications based on criteria set in the watches.
- **Monitoring Dashboard**—The Monitoring Dashboard provides views and tools for graphically presenting diagnostic data about servers and applications running on them. The underlying functionality for generating, retrieving, and persisting diagnostic data is provided by the WebLogic Diagnostics Framework. The Monitoring Dashboard provides additional tools for presenting that data in charts and graphs.

The diagnostic data displayed by the Monitoring Dashboard consists of runtime MBean attributes with numeric or Boolean values that are useful to measure, either as their current values or as their changes over time. These values, referred to in the Monitoring Dashboard as metrics, originate from one or more runtime MBean instances from one or more servers in the domain.

WLDF provides a set of standardized application programming interfaces (APIs) that enable dynamic access and control of diagnostic data, as well as improved monitoring that provides visibility into the server. Independent Software Vendors (ISVs) can use these APIs to develop custom monitoring and diagnostic tools for integration with WLDF.

WLDF enables dynamic access to server data through standard interfaces, and the volume of data accessed at any given time can be modified without shutting down and restarting the server.

Logging Services

WebLogic logging services provide facilities for writing, viewing, filtering, and listening for log messages. These log messages are generated by WebLogic Server instances, subsystems, and Java EE applications that run on WebLogic Server or in client JVMs.

WebLogic Server subsystems use logging services to provide information about events such as the deployment of new applications or the failure of one or more subsystems. A server instance uses them to communicate its status and respond to specific events. For example, you can use WebLogic logging services to report error conditions or listen for log messages from a specific subsystem.

By default, WebLogic logging services use an implementation based on the Java Logging APIs. In addition, WebLogic Server also provides the Server Logging Bridge, which provides a lightweight mechanism for applications that currently use Java Logging to have their log messages redirected to WebLogic logging services. Applications can use the Server Logging Bridge with their existing configuration; no code changes or programmatic use of the WebLogic Logging APIs is required.

SNMP Support

WebLogic Server includes support for the Simple Network Management Protocol (SNMP), which you can use to enable enterprise-wide management systems for managing heterogeneous software and hardware environments from a single management console.

With SNMP, a manager sends a request for information about managed resources to an agent. The agent gathers the requested data and returns a response. You can also configure agents to issue unsolicited reports (notifications) to managers when they detect predefined thresholds or conditions on a managed resource.

To request data about a specific managed resource, a manager must be able to uniquely identify the resource. In SNMP, each type of managed resource is described in a Management Information Base (MIB) as a managed object with a unique object identifier (OID). Individual organizations define their specific managed objects in MIB modules. Both manager and agent must have access to the same MIB module to communicate about specific managed resources.

Custom JMX Applications

To integrate custom management systems with the WebLogic Server management system, WebLogic Server provides standards-based interfaces that are fully compliant with the Java Management Extensions (JMX) specification.

Software vendors can use these interfaces to monitor WebLogic Server MBeans, to change the configuration of a WebLogic Server domain, and to monitor the distribution (activation) of those changes to all server instances in the domain. While JMX clients can perform all WebLogic Server management functions without using Oracle's proprietary classes, Oracle recommends that remote JMX clients use WebLogic Server protocols (such as T3) to connect to WebLogic Server instances.

Java EE Management APIs

The Java EE Management specification describes a standard data model for monitoring and managing the runtime state of any Java EE web application server and its resources. It includes standard mappings of the model through a Java EE Management EJB Component (MEJB).

The Java EE Management APIs enable a software developer to create a single Java program that can discover and browse resources, such as JDBC connection pools and deployed applications, on any Java EE web application server. The APIs are part of the Java EE Management Specification, which requires all Java EE web application servers to describe their resources in a standard data model.

Roadmap for Monitoring, Diagnosing, and Troubleshooting in WebLogic Server

The WebLogic Server documentation set includes several introductory, procedural, and reference topics, including examples, that help you understand how to monitor, diagnose, and troubleshoot performance in WebLogic Server.

Table 13-1 Roadmap for Monitoring, Diagnosing, and Troubleshooting in WebLogic Server

Major Task	Subtasks and Additional Information
Learning more about WLDF components	<ul style="list-style-type: none"> • Data creation, collection, and instrumentation • Archive • Policy and Action • Data accessor • Monitoring dashboard and request performance pages • Diagnostic image capture • Understanding WLDF configuration
Learning more about WebLogic logging services	<ul style="list-style-type: none"> • What you can do with WebLogic logging services • Using message catalogs with WebLogic Server • Logging components and environment • Terminology • Overview of the logging process • Best practices for integrating Java logging with WebLogic logging services • Server log files and domain log files • Server and subsystem logs • Log message format • Message attributes • Message severity • Viewing WebLogic logging services • Configuring WebLogic logging services • Filtering WebLogic Server log messages • Subscribing to messages • Using the Server Logging Bridge
Using the Monitoring Dashboard	<ul style="list-style-type: none"> • About the monitoring dashboard interface • Understanding how metrics are collected and presented • The parts of a chart
Using SNMP with WebLogic Server	<ul style="list-style-type: none"> • WebLogic Server SNMP agents • Security for SNMP • MIB module for WebLogic Server • Monitoring custom MBeans • WebLogic Server notifications • SNMP proxies • WebLogic SNMP command-line utility
Creating JMX applications to manage WebLogic Server	<ul style="list-style-type: none"> • Developing custom management utilities with JMX • Developing manageable applications with JMX • Programming WebLogic deployment
Learning more about the Java EE Management APIs	<ul style="list-style-type: none"> • JMO hierarchy • JMO object names • Optional features of JMOs • Accessing JMOs • Accessing the MEJB on WebLogic Server • WebLogic Server extensions

Table 13-1 (Cont.) Roadmap for Monitoring, Diagnosing, and Troubleshooting in WebLogic Server

Major Task	Subtasks and Additional Information
Servers do not start	If you restart the server and encounter the following error, then follow the instructions in Doc ID 2691299.1 to solve the issue. <BEA-090402> <Authentication denied: Boot identity not valid; The user name and/or password from the boot identity file (boot.properties) is not valid.

14

Sample Applications and Code Examples

WebLogic Server provides a rich set of code examples and sample applications that show several approaches to learning about and working with WebLogic Server. These examples and sample applications are available through performing a custom installation and selecting to install the Server Examples.

This chapter includes the following topics:

Installing and Running the Examples

You can create a complete WebLogic domain that is configured with the full set of deployed code examples by selecting them in the installation program.

This section contains the following topics:

- [Installing the WebLogic Server Code Examples](#)
- [Starting the WebLogic Server Samples Domain](#)
- [Running the WebLogic Server Code Examples](#)

Installing the WebLogic Server Code Examples

When performing an installation of WebLogic Server, select **Complete with Examples** to obtain the WebLogic Server and Coherence examples. For more information about installing WebLogic Server, see *Installing the Oracle WebLogic Server and Coherence Software*. To set up the WebLogic Server samples domain, launch the Quick Start Configuration Wizard when prompted after installation.

If you do not automatically launch the Quick Start Configuration Wizard from the installation program, but instead choose to configure the code examples and sample domains later, you can run the Quick Start Configuration Wizard manually. See *Running the Quick Start Configuration Wizard* in *Creating WebLogic Domains Using the Configuration Wizard*.

Starting the WebLogic Server Samples Domain

Start the examples server using one of the following procedures. In these procedures, *DOMAIN_HOME* represents the location where the samples domain is configured on your machine; for example, `C:\ORACLE_HOME\user_projects\domains`.

On Windows: Use a command shell and navigate to the `DOMAIN_HOME\wl_server` directory. Enter the following command:

```
startWebLogic.cmd
```

On UNIX Bourne Shell: Navigate to the `DOMAIN_HOME/wl_server` directory. Enter the following command:

```
sh ./startWebLogic.sh
```

 **Note:**

By default, the examples server uses port 7001 to listen for incoming connections. The MedRec server also uses the same listen port by default, which means that you cannot run both domains at the same time without changing one of the listen ports. If you want to run both domains at the same time, use the Oracle WebLogic Server Administration Console to change the listen port of the examples server to something other than 7001, and then restart it. You can then run the MedRec server using its default listen port at the same that you run the examples server.

Running the WebLogic Server Code Examples

Review the instructions provided with the code examples for information about building, deploying and running the code examples. When you start the WebLogic Server samples domain, a browser is automatically launched that displays a web page from which you can browse the samples and obtain instructions for building, deploying, and running them.

Conventions

The instructions for building, deploying, and running the WebLogic Server code examples include a number of typographical conventions for indicating the path names for entities such as the WebLogic Server installation directory, the samples domain home directory, Apache Derby, and so on, in a platform-neutral way.

The following conventions are used:

- The instructions generally are for Windows command shells. If you are using a UNIX or Linux-based shell, substitute / for \ in path names.
- *ORACLE_HOME* represents the directory you specified as the Oracle Home when you installed WebLogic Server; for example, C:\Oracle\Middleware\Oracle_Home.
- *WL_HOME* represents the top-level installation directory for Oracle WebLogic Server. The default path is *ORACLE_HOME*/wlserver. (However, you are not required to install WebLogic Server in the Oracle Home directory.)
- *EXAMPLES_HOME* represents the directory in which the WebLogic Server code examples are configured. The default path is *ORACLE_HOME*\wlserver\samples\server.
- *DOMAIN_HOME* represents the directory in which the WebLogic Server sample domains are configured. The default path is *ORACLE_HOME*\user_projects\domains.

Source files for the code examples are separated from the domain configuration files, just as they should be in a real-world scenario. They are installed in the *EXAMPLES_HOME* directory.

The *DOMAIN_HOME*\wl_server directory contains the WebLogic Server examples domain; it contains your applications and the XML configuration files that define how your applications and Oracle WebLogic Server will behave, as well as startup and environment scripts.

The `EXAMPLES_HOME\examples\build` directory contains client and server classes required by the examples and Derby database.

The `WL_HOME\common\derby` directory contains Derby, a demonstration database that the examples are configured to use. It also contains scripts that start and stop the database. For more information about Derby, see <http://db.apache.org/derby>.

Java EE 8 Examples

Oracle WebLogic Server fully supports the Java Platform, Enterprise Edition (Java EE) 8 specification. The Java EE 8 examples that are included demonstrate how to implement Java EE 8 APIs and Oracle WebLogic Server-specific features.

The Java EE 8 examples are grouped into the following categories:

- JSON Binding 1.0—Use the Java API for JSON Binding (JSON-B) with JAX-RS.
- Java EE Security 1.0—Configure a `DatabaseIdentityStore` to point to a backend database and then use it as an `IdentityStore`.
- JSON Processing 1.1—Use JSON Patch, JSON Merge Patch, and JSON Pointer to update a JSON document.
- Servlet 4.0—Use the Servlet Mapping API, HTTP/2 Server Push, and the HTTP Trailer headers API.
- JAX-RS 2.1—Use the new Server-Sent Events (SSE) and the new REST Reactive Client API.
- JSF 2.3—Use the new features of JSF 2.3 such as direct support for WebSockets, class-level bean validation, CDI-compatible `@ManagedProperty` annotation feature, and Java date and time.
- JPA 2.2—Use injection in `@AttributeConverter` annotation, new Java EE 8 Date and Time, and also use support for retrieving the results of `Query` and `TypedQuery` as streams.
- CDI 2.0—Use asynchronous events, observer ordering, and also use interception factory.

Java EE 7 Examples

Oracle WebLogic Server fully supports the Java Platform, Enterprise Edition (Java EE) 7 specification. The Java EE 7 examples that are included demonstrate how to implement Java EE 7 APIs and Oracle WebLogic Server-specific features.

The Java EE 7 examples are grouped into the following categories:

- Batch Application Processing 1.0—Submit batch jobs and obtain information about submitted jobs using the `JobOperator` interface, and use the batch parallelization model to run partitioned job steps.
- Bean Validation 1.1—Use the bean validation group constraint and method level validation APIs.
- Context and Dependency Injection (CDI) 1.1—Use CDI events and the `@TransactionScoped` and `@Transactional` annotations.
- Concurrency Utilities 1.0—Create dynamic proxy objects using the `ContextService` interface, submit tasks using the `ManagedExecutorService` interface, submit delayed or periodic tasks using the `ManagedScheduledExecutorService` interface,

and obtain a managed thread from the Java EE container using the `ManagedThreadFactory` interface.

- Enterprise JavaBeans 3.2—Use the new session bean life cycle callback interceptor methods API and also use a message-driven bean to implement a listener interface with no methods.
- Expression Language 3.0—Use new EL features, including support for a standalone environment, static field or method references, new operators, Lambda expressions, and collection constructions and operations.
- Java API for RESTful Web Services (JAX-RS) 2.0—Use asynchronous processing, filters and interceptors, and server-sent events (SSE) Jersey support.
- Java EE Connector Architecture 1.7—Develop a resource adapter and deploy connector resources with annotations defined in the Java EE Connector Architecture 1.7 specification.
- Java Message Service API 2.0—Use the JMS API in EJBs and servlets.
- Java Persistence 2.1—Use JPA criteria update and criteria delete APIs, and the stored procedures API.
- JSF 2.2—Use Java Server Faces (JSF) resource library contracts, file upload, faces flows, and HTML5 features.
- JSON Processing 1.0—Use the Java API for JSON processing with JAX-RS.
- Servlet 3.1—Use the HTTP protocol upgrade API, use non-blocking I/O for asynchronous reads and writes, change a session ID, and handle uncovered HTTP methods.
- WebSocket—Process JSON-format data, using CDI and EJBs in WebSocket endpoints, enable a server to echo text sent by a client, and enable fallback to HTTP long polling as an alternative for WebSocket messaging.

Additional API Examples

Oracle WebLogic Server also includes a set of examples that demonstrate how to implement additional Java EE APIs and Oracle WebLogic Server-specific features. These examples are grouped in the following categories:

- Database Connectivity—Use Data Sources, Multi Data Sources, and Rowsets.
- EJB—Create stateless, stateful, entity, and message-driven EJBs, and more.
- Internationalization—Internationalize an application using simple message catalogs.
- Messaging—Use JMS topics, queues, and message-driven beans.
- Resource Adapter—Use an entity EJB to interact with a Java EE Connector Architecture resource adapter.
- Security—Use the Java Authentication and Authorization Service, SAML, and outbound and two-way SSL.
- Transactions—Use JTA to perform distributed transactions using the two phase commit protocol across two XA resources.
- Web Application—Create simple servlets and JSPs, use the HTTP Publish-Subscribe server, and more.
- Web Services—Create a variety of web services using JWS annotations.

- XML—Use the STAX API and XMLBeans
- Cluster—Cluster an EJB and use HTTP session state replication.
- Coherence—Use the Coherence container to host Coherence applications
- WebLogic Scripting Tool—Use the WebLogic Scripting Tool (WLST) to configure and manage a running Administration Server instance.
- Split Development—Use the WebLogic split development directory structure to build, package, and deploy Enterprise Applications.
- Service Component Architecture—Use WebLogic SCA, a lightweight Spring 2.5 (or higher) container, in a shopping cart application that demonstrates many of its key features.
- Spring—Use Spring-simplified configuration in a Spring-based web application.

Avitek Medical Records

Avitek Medical Records (also known as MedRec) is a comprehensive educational sample application that demonstrates WebLogic Server and Java EE features, as well as best practices. Avitek Medical Records is optionally installed with the WebLogic Server installation.

You can start MedRec from the `ORACLE_HOME/user_projects/domains/medrec` directory, where `ORACLE_HOME` is the directory you specified as the Oracle Home when you installed Oracle WebLogic Server.

The sample application, MedRec (Spring) demonstrates Spring Framework application development practices.

Derby Open-Source Database

Derby is an open source relational database management system based on Java, JDBC, and SQL standards. Derby is bundled with WebLogic Server for use by the sample applications and code examples as a demonstration database.

For more information about Derby, see <http://db.apache.org/derby>.

WebLogic Server Compatibility

Oracle attempts to support binary and source-level compatibility between Oracle WebLogic Server 14c (14.1.1.0.0) and versions 10.3.6, 12.1.3 and 12.2.1.x in the areas of persistent data, generated classes, and API compatibility. In some cases, it is impossible to avoid incompatibilities. Where incompatibilities arise, they are fully documented in *Upgrading Oracle WebLogic Server*.

This chapter includes the following topics:

Java EE 8 Compatibility

WebLogic Server 14c (14.1.1.0.0) is Java EE 8 compatible. This compatibility allows a Java EE 8 compliant application to be developed on one operating system platform, and deployed for production on another, without requiring Java EE 8 application code changes.

Oracle ensures this compatibility of Java EE 8 application portability within a WebLogic Server release level.

Generated Classes Compatibility

Upgrading to WebLogic Server 14c (14.1.1.0.0) does not require you to recompile applications in order to create new generated classes.

Compatibility Within a Domain

Within the scope of a WebLogic domain, Oracle WebLogic Server supports a wide range of compatibility with respect to the specific versions of WebLogic Server instances that can run in that domain, as well as the mix of hardware, operating system, and JVM platforms on which those server instances can run.

However, depending upon the specific configurations present in the domain, such as WebLogic clusters, Oracle has specific recommendations for how you can achieve optimal performance. The following topics provide key information regarding compatibility within WebLogic domains:

About WebLogic Server Version Numbers

Within a WebLogic domain, the Administration Server, Managed Server instances, and the domain itself each have a WebLogic Server version number. The version number contains five decimal places; for example, WebLogic Server 14.1.1.0.0. The meaning of each decimal place is described below:

- The first two decimal places together describe the Major Version number, for example "14.1" in **14.1**.1.0.0. The WebLogic Server 14.1 Major Version release is also branded as the WebLogic Server 14c Major Version release.
- The first three decimal places together describe the Minor Version number, for example "14.1.1" in **14.1.1**.0.0. WebLogic Server 14.1.1 (or 14.1.1.0.0) is the first Minor Version

release of the WebLogic Server 14.1 Major Version release. WebLogic Server 14.1.2 (or 14.1.2.0.0) would be the second Minor Version release of the WebLogic Server 14.1 Major Version release.

- Patch Set releases for WebLogic Server 14.1.1.0.0 will increment the fourth decimal place. For example, 14.1.1.1.0 would be the first patch set release.
- Patch Set Update releases are named uniquely by incrementing the fifth decimal place with the date of the Patch Set Update release in *YYMMDD* format; for example, 14.1.1.0.200731. This convention is used for Patch Set Update naming purposes; for example, naming downloads available on My Oracle Support. However, the application of a Patch Set Update does not change the version number of an existing WebLogic Server installation as referenced in the Oracle inventory directory (`oraInventory`) used by WebLogic Server 14.1.1 installers.

You can obtain the version number and Patch Set level of a WebLogic Server instance or domain several different ways. For example:

- For an Administration Server or Managed Server instance, you can view the version message sent to `stdout` when the server is started. For example:

```
<Version: WebLogic Server 14.1.1.0.0 Sat May 11 12:34:37 PDT 2019 1960751 >
```
- For a domain, you can view the value of the `<domain-version>` element in the domain configuration file, `config.xml`. For example:

```
<domain-version>14.1.1.0.0</domain-version>
```

WebLogic Version Compatibility

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 14.1.1, the Administration Server, Managed Servers, and the WebLogic domain must all be at version 14.1.1. Note the following guidelines for maintaining consistency in Patch Set Update and Interim or One-off Patch levels within a domain.

Note:

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. See *Upgrading Oracle WebLogic Server*.

- In general, the best practice is for all server instances within a domain to be at the same Patch Set Update (PSU) and Interim or One-off Patch level during steady-state operation. However, there may be cases where server instances are required to run at different PSUs and Interim or One-off Patch levels within a domain. The primary examples include:
 - When applying PSUs, Interim or One-off Patches in rolling fashion across server instances in the domain. In such cases, the maintenance should be applied to the Administration Server first, so that the Administration Server is at the same PSU and Interim or One-off Patch level (or higher) than its Managed Servers. See *About Rolling Upgrade* in *Upgrading Oracle WebLogic Server*.

- When there are specific requirements to run Managed Servers within a domain at different PSU and Interim or One-off Patch levels in steady-state operation. In such cases, the Administration Server should be at the highest PSU level, so that the Administration Server is at the same PSU level or higher than all of the Managed Servers. If Managed Servers within a domain are running with different Interim or One-off Patches, it will not be possible to apply a consistent set of Interim or One-off Patches to the Administration Server. Because this maintenance complexity may be difficult to manage, the general best practice is to use the same PSU and Interim or One-off Patch level across all servers in the domain.
- Server instances within a cluster or domain can run on any hardware and operating systems as long as the hardware and operating systems are listed on the Oracle Fusion Middleware Supported System Configurations page on Oracle Technology Network. However, note that running clustered Managed Server instances on different hardware and operating systems may impact load balancing and performance. In general, the best practice is to run all Managed Servers within a cluster on the same hardware and operating system.
- If the WebLogic domain is part of an Oracle Enterprise Manager Cloud Control installation, additional requirements exist regarding the combinations of hardware, operating system, and JVMs, that may be configured in the domain. See [Oracle Enterprise Manager Cloud Control Administrator's Guide](#).

For more information about WebLogic domains and additional details about domain compatibility, see Domain Restrictions in *Understanding Domain Configuration for Oracle WebLogic Server*.

Hardware, Operating System, and JVM Platform Compatibility

WebLogic Server instances within a domain can run on any hardware, operating system, and JVM platform as long as the hardware, operating systems, and JVMs are supported for the current version of WebLogic Server. For details, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.

Note:

Although this platform compatibility support extends to Managed Server instances within a cluster, Oracle strongly recommends that clusters be homogeneous with respect to the underlying hardware, operating system, and JVM. Managed Server instances running in the same cluster are assumed to be equivalent, so running clustered server instances on mixed platforms may have a negative impact on load balancing and performance. If you must operate a cluster on a mixed platform, Oracle strongly recommends that you understand the load balancing and performance implications.

Node Manager Compatibility

As a best practice, Oracle recommends that the version of Node Manager used in a WebLogic domain should match the version of the Administration Server.

If you are running a version of WebLogic Server prior to 12.1.2, see *Upgrading Oracle WebLogic Server* for additional Node Manager compatibility information that may be

applicable to your environment. For more information about Node Manager compatibility, see *Administering Node Manager for Oracle WebLogic Server*.

Persistent Data Compatibility

If you are upgrading Oracle WebLogic Server from release 10.3.6 to release 14c (14.1.1.0.0), a number of updates are required to several configuration files. However, the upgrade tooling that has been available in Oracle WebLogic Server since release 10.3.6 makes those configuration file updates for you automatically.

API Compatibility

WebLogic Server 10.3.6, 12.1.3, and 12.2.1.x applications deployed on WebLogic Server 14c (14.1.1.0.0) will function without modification. Exceptions to this rule include cases where APIs were deprecated in prior WebLogic Server releases and removed, or where API behavior was changed in order to conform to a specification or to fix incorrect behavior. In certain circumstances, a correction may cause your application to behave differently.

For a list of previously deprecated APIs that are removed in Oracle WebLogic Server 14.1.1.0.0, see Removed Functionality and Components in *What's New in Oracle WebLogic Server 14.1.1.0.0*.

Protocol Compatibility

Interoperability between WebLogic Server 14c (14.1.1.0.0) and WebLogic Server 10.3.6, 12.1.3, and 12.2.1.x is supported in several scenarios with regard to WebLogic clients, transport protocols, and WebLogic proxy plug-ins. However, a key interoperability restriction exists regarding WebLogic domains that are configured to use Compatibility security.

The supported client-server interoperability scenarios are as follows:

- A WebLogic Server 10.3.6, 12.1.3, and 12.2.1.x client can invoke RMI-based applications hosted on a WebLogic Server 14c (14.1.1.0.0) server using IIOP, T3, T3S, HTTP, and HTTPS. JMS applications can be invoked using T3, T3S, HTTP, and HTTPS.
- A WebLogic Server 14c (14.1.1.0.0) client can invoke RMI-based applications hosted on a WebLogic Server 10.3.6, 12.1.3, and 12.2.1.x server using IIOP, T3, T3S, HTTP, and HTTPS. JMS applications can be invoked using T3, T3S, HTTP, and HTTPS.

The supported server-server IIOP interoperability scenarios are as follows:

- WLS 14.1.1/JDK 11 servers <-> WLS 14.1.1/JDK 11 servers
- WLS 14.1.1/JDK 8 servers <-> WLS 14.1.1/JDK 8 servers
- WLS 14.1.1/JDK 8 servers <-> WLS 12.2.1.4/JDK 8 servers
- WLS 14.1.1/JDK 8 servers <-> WLS 12.2.1.3/JDK 8 servers
- WLS 14.1.1/JDK 8 servers <-> WLS 12.1.3/JDK 8 servers
- WLS 14.1.1/JDK 8 servers <-> WLS 12.1.3/JDK 7 servers
- WLS 14.1.1/JDK 8 servers <-> WLS 10.3.6/JDK 7 servers

WebLogic Server 14c (14.1.1.0.0) supports Java client to server IIOp communications and interoperability with the following restrictions:

- The only supported WebLogic Java IIOp clients are:
 - A `weblogic.jar` install client from 14.1.1.0.0 or earlier WebLogic Server versions that are supported and under error correction.
 - A `wlfullclient.jar` from earlier WebLogic Server versions that are supported and under error correction (WebLogic Server 14c (14.1.1.0.0) does not include `wlfullclient.jar`).
 - A `wlclient.jar` from earlier WebLogic Server versions that are supported and under error correction (WebLogic Server 14c (14.1.1.0.0) does not include `wlclient.jar`).
- A Java SE client (no JAR file from WebLogic Server in the class path) is *not* supported.
- For WebLogic Server 14c (14.1.1.0.0) instances running on JDK11, IIOp interoperability with Java clients is only available with a WebLogic Server 14c (14.1.1.0.0) install client running on JDK 11.
- Interoperability support is *not* available between WebLogic Server 14c (14.1.1.0.0) instances running on JDK8, and WebLogic Java IIOp clients running on JDK 11.

 **Note:**

Oracle recommends using a T3 capable WebLogic Java client and the T3 protocol instead of IIOp when possible. It is rare that the IIOp protocol is required for Java to Java communications; T3 is more efficient than IIOp. For more information, see [Overview of Standalone Clients](#) in *Developing Standalone Clients for Oracle WebLogic Server*.

One important restriction is with regard to interoperability between WebLogic Server 14c (14.1.1.0.0) and an older release of WebLogic Server that uses Compatibility security. As of WebLogic Server version 12.2.1, support for Compatibility security is removed in both the server and client. To enable interoperability with a version of WebLogic Server that uses Compatibility security, you can choose one of the following options:

- Configure the network channel used for communicating with the older WebLogic Server domain to use the IIOp protocol instead of T3. See [Configuring Network Resources](#) in *Administering Server Environments for Oracle WebLogic Server*.
- Use a WebLogic Server version 12.1.3 or prior client. See [Overview of Standalone Clients](#) in *Developing Standalone Clients for Oracle WebLogic Server*.
- Upgrade the domain of the older WebLogic Server release so that it no longer uses Compatibility security.