Oracle® Fusion Middleware

Upgrade Guide for Java EE

11*g* Release 1 (11.1.1)

E10126-03

April 2010

ORACLE®

Oracle Fusion Middleware Upgrade Guide for Java EE, 11*g* Release 1 (11.1.1)

E10126-03

Primary Author: Peter LaQuerre

Contributing Author: Reza Shafii

Contributors: Janga Aliminati, Pyounguk Cho, Paul Dickson, Robert Donat, Shail Goel, William Norcott, Michael Rubino, Robert St. Jean, Gavin Steyn, Sitaraman Swaminathan, Suresh Srinivasan, Zhong Xu, Lixin Zheng

Contents

Preface ix

Audience ix
Documentation Accessibility ix
Related Documents x
Conventions x

1 Summary of the Java EE Upgrade Process

1.1 Flow Chart of the Java EE Upgrade Process 1-1
1.2 Table Describing the Steps in the Java EE Upgrade Process 1-2

Part I Preparing for a Java EE Upgrade

2 Supported Starting Point for Java EE Upgrade

3 Introduction to Oracle WebLogic Server for OC4J Users

3.1 Key Oracle WebLogic Server Concepts for OC4J Users 3-1
3.1.1 Comparing the OC4J and Oracle WebLogic Server Architectures 3-1
3.1.1.1 Standalone OC4J and Standalone Oracle WebLogic Server 3-2
3.1.1.2 OC4J and Oracle WebLogic Server Integrated With a Web Server 3-2
3.1.1.3 OC4J and Oracle WebLogic Server Clustering Features 3-3
3.1.2 Comparing OC4J and Oracle WebLogic Server Directory Structures 3-4
3.1.3 Understanding Oracle WebLogic Server Domains for OC4J Users 3-5
3.1.3.1 Basic Content and Organization of a Domain 3-5
3.1.3.2 Additional Facts About Oracle WebLogic Server Domains 3-6
3.1.3.2.1 A WebLogic Server Installation Can be Used to Configure Multiple Domains 3-6
3.1.3.2.2 A WebLogic Server Instance Is Always Associated With a Single Java Virtual Machine Process 3-6
3.1.3.2.3 A WebLogic Server Instance Processes All Application Requests on the Same Port by Default 3-6
3.1.3.2.4 A WebLogic Server Instance Is Always Configured With an HTTP Listener and Does Not Support AJP 3-7
3.2 Oracle WebLogic Server Installation and Configuration Tools for OC4J Users 3-7
3.3 Oracle WebLogic Server Administration Tools for OC4J Users 3-8
3.3.1 Comparison of OC4J and Oracle WebLogic Server Administration Tools 3-8

3.3.2 Typical Oracle WebLogic Server Administration Tasks for OC4J Users 3-9
3.3.2.1 Starting and Stopping Servers 3-9
3.3.2.2 Performing Diagnostics on a Domain 3-9
3.3.2.3 Viewing Log Files for a Domain 3-10
3.3.2.4 Configuring and Tuning Thread Pools 3-10
3.4 Standards Support for OC4J and Oracle WebLogic Server 3-10

Part II Upgrading Your Java EE Applications and Environment

4 Upgrading Your Java EE Applications

4.1 Task 1: Verify that Your Application Deploys and Works Successfully on OC4J 4-1
4.2 Task 2: Select Your Development Tools 4-1
4.2.1 General Guidelines for Selecting Your Development Tools 4-1
4.2.2 Using the SmartUpgrade Oracle JDeveloper Extension and Command-Line Tool .. 4-2
4.3 Task 3: Verify That Your Application Supports Java Development Kit (JDK) 6 4-2
4.4 Task 4: Upgrade the Application Deployment Descriptors 4-2
4.4.1 Comparison of OC4J and Oracle WebLogic Server Deployment Descriptors 4-2
4.4.2 Guidelines and Resources for Upgrading Deployment Descriptors for Oracle WebLogic Server 4-3
4.4.3 About Security Elements in Deployment Descriptor Files 4-4
4.4.4 Upgrading Deployment Plans 4-4
4.5 Task 5: Review Oracle WebLogic Server API Support 4-4
4.5.1 APIs Available With the Java Required Files (JRF) Domain Template 4-4
4.5.2 Other Oracle WebLogic Server API Requirements 4-5
4.6 Task 6: Upgrade the Application Web Services 4-6
4.6.1 General Guidelines for Upgrading to Oracle WebLogic Server JAX-RPC and JAX-WS Web Services 4-6
4.6.2 Generating Oracle WebLogic Server Web Services From an OC4J WSDL 4-7
4.6.3 Web Services Specifications Supported by OC4J and Oracle WebLogic Server 4-7

5 Upgrading Your Java EE Environment

5.1 Task 1: Install and Configure an Oracle WebLogic Server Development Domain 5-1
5.1.1 Differences Between a Development Environment and a Test or Production Environment 5-1
5.1.2 Installing and Configuring a Development Domain with Oracle JDeveloper 5-2
5.1.3 Installing and Configuring a Development Domain with Oracle SOA Suite, WebCenter, or Application Developer 5-2
5.1.3.1 Advantages of Installing an Oracle SOA Suite, WebCenter, or Application Developer Development Environment 5-3
5.1.3.2 Selecting an Oracle Fusion Middleware Software Suite 5-3
5.1.3.3 Steps Required to Install and Configure an Oracle SOA Suite, WebCenter, or Application Developer Domain 5-3
5.1.4 Using the Java Required Files (JRF) Domain Template 5-4
5.1.4.1 Creating a New Domain With the JRF Template 5-4
5.1.4.2 Extending an Existing Domain With the JRF Template 5-5
5.2 Task 2: Verify the New Oracle Fusion Middleware 11*g* Environment 5-5

5.3 Task 3: Configure Oracle WebLogic Server Resources to Support Your Applications 5-6
5.3.1 Configuring JDBC Data Sources on Oracle WebLogic Server 5-6
5.3.1.1 General Information About Defining Data Sources for OC4J and Oracle WebLogic Server 5-6
5.3.1.2 Upgrading Application-Level OC4J Data Sources 5-7
5.3.1.3 Upgrading Instance and Group-Level OC4J Data Sources 5-7
5.3.1.4 JDBC Connection Pools and Managed Data Sources in OC4J and Oracle WebLogic Server 5-7
5.3.2 Configuring OC4J JMS Resources on Oracle WebLogic Server 5-7
5.3.2.1 Overview of JMS Support in OC4J and Oracle WebLogic Server 5-7
5.3.2.2 Creating and Managing JMS Resources in OC4J and Oracle WebLogic Server . 5-8
5.3.3 Configuring OC4J Remote JMS Resources on Oracle WebLogic Server 5-8
5.3.4 Using Shared Libraries and Class Loading on Oracle WebLogic Server 5-9
5.3.5 Configuring Startup and Shutdown Classes 5-10
5.3.6 Configuring Security on Oracle WebLogic Server 5-10
5.3.7 Configuring Logging on Oracle WebLogic Server 5-11
5.4 Task 4: Redeploy the Application on Oracle WebLogic Server 5-12
5.5 Task 5: Verify the Redeployed Applications 5-12

6 Upgrading Application Clients

6.1 Impact of Upgrade on Java Server Pages and Servlet Clients 6-1
6.2 Impact of Upgrade on Java Naming and Directory Interface Clients 6-1
6.2.1 Modifying Clients to Use the Oracle WebLogic Server JNDI Provider 6-2
6.2.2 Understanding the Scope of the Oracle WebLogic Server JNDI Namespace 6-2
6.3 Impact of Upgrade on Enterprise Java Bean Clients 6-3
6.3.1 Impact on Remote Standalone EJB Clients 6-3
6.3.2 Impact on Clients That Use OC4J-Based EJB Interfaces 6-4
6.4 Impact of Upgrade on JMS Clients 6-4
6.4.1 Changes Required When the JMS Provider is Upgraded to WebLogic Server 6-4
6.4.2 Changes Required When the JMS Provider Remains in OC4J 6-5

7 Upgrading a Java EE and Web Server Environment

7.1 Task 1: Understand the Differences Between Using Oracle HTTP Server with OC4J and Oracle WebLogic Server 7-1
7.1.1 Configuring Web Sites and AJP Connections in Oracle WebLogic Server 7-1
7.1.2 Installing and Configuring Oracle HTTP Server for Oracle WebLogic Server 7-2
7.1.2.1 How Oracle HTTP Server Is Configured for OC4J 7-2
7.1.2.2 How Oracle HTTP Server is Configured for Oracle WebLogic Server 7-2
7.1.3 Using Web Servers Other than Oracle HTTP Server with Oracle WebLogic Server. 7-3
7.1.4 Understanding Oracle HTTP Server Interoperability Issues When Upgrading to Oracle Fusion Middleware 11*g* 7-3
7.2 Task 2: Install and Configure an Oracle Fusion Middleware Web Tier 7-3
7.2.1 Deciding Upon a Location for Your Web Tier Components 7-3
7.2.2 Associating the Web Tier Components with an Oracle WebLogic Server Domain .. 7-4
7.2.3 Locating the Web Tier Installation and Configuration Documentation 7-4
7.3 Task 3: Upgrade Your Oracle Application Server 10*g* Web Tier Components to

Oracle Fusion Middleware 11*g* 7-5
7.3.1 Task 3a: Start the Upgrade Assistant for an Web Tier Upgrade 7-5
7.3.2 Task 3b: Use the Upgrade Assistant to Upgrade the Web Tier Components 7-7
7.3.2.1 Upgrading the Web Tier Components 7-7
7.3.2.2 Important Notes When Using the Source Oracle Home Ports in the Destination Oracle Instance 7-8
7.4 Task 4: Configure the Web Tier To Route Requests to Your Oracle Fusion Middleware Environment 7-9
7.5 Task 5: Perform Any Required Post-Upgrade Tasks for the Web Tier Components 7-9
7.5.1 Verifying the Location of the Oracle HTTP Server and Oracle Web Cache Wallets After Upgrade 7-9
7.5.2 Verifying and Updating the Oracle HTTP Server and Oracle Web Cache Ports After Upgrade 7-10
7.6 Task 6: Verify the Web Tier Upgrade 7-10

A orion-web.xml and orion-ejb-jar.xml Upgrade Reference

orion-web.xml A-2
<classpath> A-2
<contextParamMappingFinding> A-2
<mimeMappings> A-3
<virtual-directory> A-3
<access-mask> A-3
<servlet-chaining> A-4
<request-tracker> A-4
<session-tracking> A-4
<session-tracker> A-5
<resource-ref-mapping> A-5
<lookup-context> A-5
<resource-env-ref-mapping> A-6
<env-entry-mapping> A-6
<ejb-ref-mapping> A-6
<service-ref-mapping> A-7
<expiration-setting> A-7
<jazn-web-app> A-7
<security-role-mapping> A-8
<web-app-class-loader> A-8
search-local-classes-first A-8
include-war-manifest-class-path A-9
autojoin-session A-9
default-buffer-size A-9
default-charset A-10
default-mime-type A-10
development A-11

directory-browsing ... A-11
enable-jsp-dispatcher-shortcut ... A-11
file-modification-check-interval ... A-12
jsp-cache-directory ... A-12
jsp-cache-tlds ... A-12
jsp-print-null ... A-13
jsp-taglib-locations ... A-13
jsp-timeout ... A-13
persistence-path ... A-14
schema-major-version ... A-14
schema-minor-version ... A-14
servlet-webdir ... A-15
simple-jsp-mapping ... A-15
source-directory ... A-15
temporary-directory ... A-16
orion-ejb-jar.xml ... A-17
<session-deployment> ... A-17
copy-by-value ... A-17
idletime ... A-18
min-instances ... A-18
max-instances ... A-18
max-instances-threshold ... A-19
max-tx-retries ... A-19
resource-check-interval ... A-19
passivate-count ... A-20
persistence-filename ... A-20
pool-cache-timeout ... A-20
timeout ... A-21
transaction-timeout ... A-21
<ejb-ref-mapping> ... A-21
<resource-ref-mapping> ... A-22
<resource-env-ref-mapping> ... A-22
<message-destination-ref-mapping> ... A-23
<session-type>Stateful</session-type> ... A-23
<message-driven-deployment> ... A-23
connection-factory-location ... A-24
dequeue-retry-count ... A-24
dequeue-retry-interval ... A-25
destination-location ... A-25
listener-threads ... A-25
max-delivery-count ... A-26

resource-adapter A-26
subscription-name A-26
wrapper-class A-27
<config-property> A-27

Index

Preface

This preface contains the following sections:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

This manual is intended for Oracle Fusion Middleware application developers who are responsible for developing Java EE applications, as well as administrators who deploy the applications and maintain Oracle Fusion Middleware installations. It is assumed that the readers of this manual have knowledge of the following:

- Oracle Fusion Middleware system administration and configuration
- The configuration and expected behavior of the systems being upgraded
- Java EE development and best practices

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Related Documents

For more information, see the following related documentation available in the Oracle Fusion Middleware 11*g* documentation library:

- Related Upgrade Documentation
  - *Oracle Fusion Middleware Upgrade Planning Guide*
  - *Oracle Fusion Middleware Upgrade Guide for Oracle SOA Suite, WebCenter, and ADF*
  - *Oracle Fusion Middleware Upgrade Guide for Oracle Portal, Forms, Reports, and Discoverer*
  - *Oracle Fusion Middleware Upgrade Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Installation Planning Guide*
- *Oracle Fusion Middleware Administrator's Guide*

Conventions

The following text conventions are used in this document:

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

1

Summary of the Java EE Upgrade Process

This chapter provides a high-level overview of how you can upgrade your Java EE environment and your deployed applications from Oracle Application Server 10*g* and Oracle Containers for Java EE (OC4J) to Oracle Fusion Middleware 11*g* and Oracle WebLogic Server.

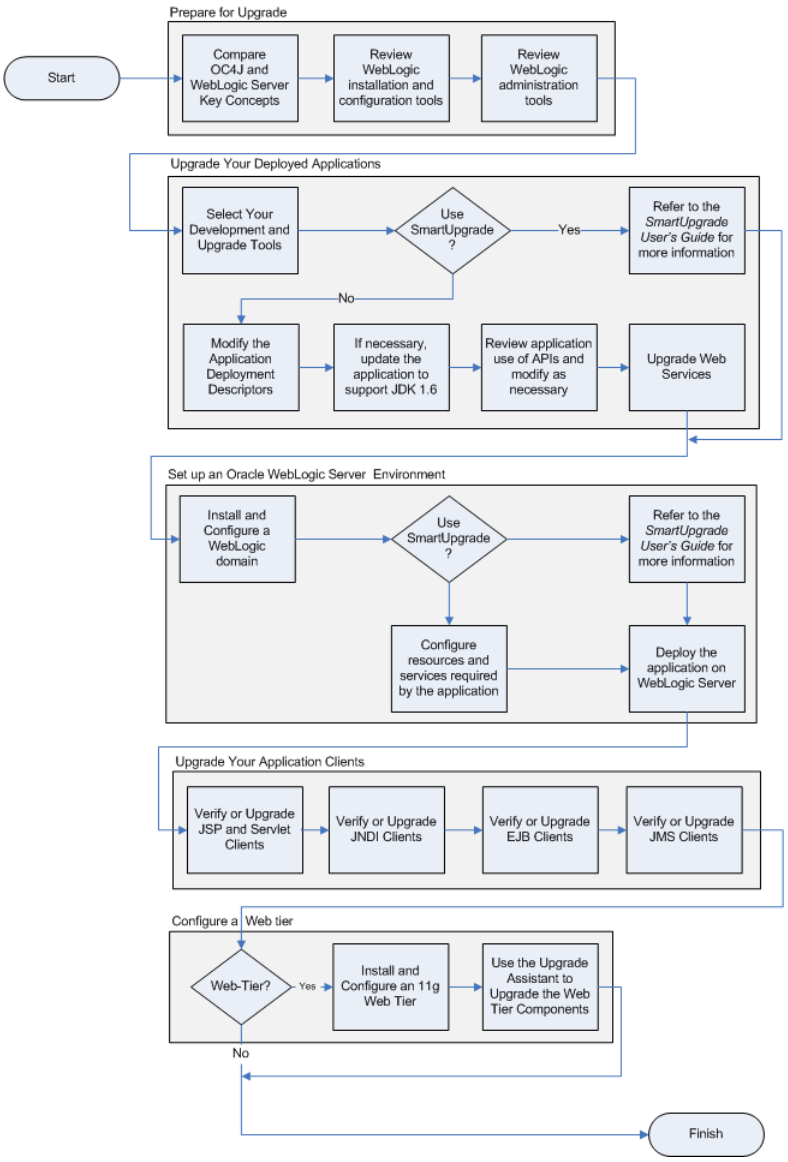
Refer to the following sections for more information:

- Flow Chart of the Java EE Upgrade Process
- Table Describing the Steps in the Java EE Upgrade Process

1.1 Flow Chart of the Java EE Upgrade Process

Figure 1–1 provides a flow chart of the Java EE upgrade process. Review this chart to get familiar with the tasks you will be required to perform.

Figure 1–1 Flow Chart of the Java EE Upgrade Process



1.2 Table Describing the Steps in the Java EE Upgrade Process

Table 1–1 describes each of the steps in the upgrade process flow chart, which is shown in Figure 1–1. The table also provides information on where to get more information on each step in the process.

Table 1–1 Table Describing the Steps in the Java EE Upgrade Process

Step	Description	More Information
Prepare for Upgrade	Before you begin the upgrade of your Java EE applications and environment from OC4J to Oracle WebLogic Server: ■ Review the key concepts of Oracle WebLogic Server and how they differ from OC4J. ■ Understand the different installation and configuration tasks required for Oracle WebLogic Server. ■ Review the administration tools and processes you will have to use in the Oracle WebLogic Server environment.	Part I, "Preparing for a Java EE Upgrade"
Upgrade Your Deployed Applications	Modify the applications as necessary so they will deploy and run successfully on Oracle WebLogic Server Note that the SmartUpgrade command-line tool and Oracle JDeveloper extension can help you identify the tasks required to upgrade your applications, and in some cases, generate specific artifacts to help you modify the application.	Chapter 4, "Upgrading Your Java EE Applications" *Oracle Fusion Middleware SmartUpgrade User's Guide*
Set up an Oracle WebLogic Server Environment	After you upgrade your application source code, you can then install and configure an Oracle WebLogic Server environment for the application. When you configure the Oracle WebLogic Server domain, you an also configure any resources required by the application. Note that the SmartUpgrade command-line tool and Oracle JDeveloper extension can help you identify how to configure the required resources for your application in Oracle WebLogic Server.	Chapter 5, "Upgrading Your Java EE Environment" *Oracle Fusion Middleware SmartUpgrade User's Guide*
Upgrade Your Application Clients	Upgrade any client applications that depend upon the applications you have upgraded to Oracle WebLogic Server.	Chapter 6, "Upgrading Application Clients"

Table 1–1 (Cont.) Table Describing the Steps in the Java EE Upgrade Process

Step	Description	More Information
Configure a Web Tier	Optionally, install and configure a Web tier to route requests to your test and production environments. If you were using Oracle HTTP Server or Oracle Web Cache in Oracle Application Server 10*g*, then you can upgrade your Oracle HTTP Server and Oracle Web Cache configuration using the Oracle Fusion Middleware Upgrade Assistant.	Chapter 7, "Upgrading a Java EE and Web Server Environment"

Part I

Preparing for a Java EE Upgrade

Part I contains the following chapters:

- Chapter 2, "Supported Starting Point for Java EE Upgrade"
- Chapter 3, "Introduction to Oracle WebLogic Server for OC4J Users"

2

Supported Starting Point for Java EE Upgrade

This guide provides instructions for upgrading your Java EE applications and environment from Oracle Application Server 10*g* Release 3 (10.1.3.1.0) to Oracle Fusion Middleware 11*g*.

Note the following about the Oracle Application Server 10*g* Release 3 (10.1.3) software:

- Oracle Application Server 10*g* Release 3 (10.1.3.1.0) included the components of the Oracle SOA Suite, but provided options for installing only OC4J, or only OC4J and Oracle HTTP Server. These "Standalone OC4J Instance" and "Integrated Web Server and OC4J Middle Tier" topologies are the focus of this guide.

  For information about upgrading your SOA environment and SOA applications, see the *Oracle Fusion Middleware Upgrade Guide for Oracle SOA Suite, WebCenter, and ADF*.

- Before you begin the upgrade process, you should ensure that you have applied the latest Oracle Application Server 10*g* Release 3 (10.1.3) patch set.

  The 10.1.3.4 patch set was the latest patch set available at the time this guide was published.

  For a list of the latest patch sets available for your installation, refer to My Oracle Support (formerly, Oracle*MetaLink*):

  `http://support.oracle.com`

  Note: If you are currently running Oracle Fusion Middleware 11*g*, refer to the *Oracle Fusion Middleware Patching Guide*, which provides information about applying the latest Oracle Fusion Middleware patches.

  This guide, as well as the other upgrade guides available in the Oracle Fusion Middleware 11*g* documentation library, provide instructions for upgrading from Oracle Application Server 10*g* to the latest Oracle Fusion Middleware 11*g* release.

3

Introduction to Oracle WebLogic Server for OC4J Users

In past releases, Oracle Application Server included Oracle Containers for Java EE (OC4J), which provided an industry-standard container for developing, deploying, and managing your Java EE applications.

For Oracle Fusion Middleware 11*g*, OC4J is replaced with Oracle WebLogic Server.

As a result, the core architecture of Oracle Fusion Middleware is built around the WebLogic Server domain, which consists of an Administration Server and one or more managed servers.

Refer to the following sections for more information:

- Key Oracle WebLogic Server Concepts for OC4J Users
- Oracle WebLogic Server Installation and Configuration Tools for OC4J Users
- Oracle WebLogic Server Administration Tools for OC4J Users
- Standards Support for OC4J and Oracle WebLogic Server

3.1 Key Oracle WebLogic Server Concepts for OC4J Users

The following sections provide some key concepts for OC4J users as they prepare to upgrade to Oracle WebLogic Server:

- Comparing the OC4J and Oracle WebLogic Server Architectures
- Comparing OC4J and Oracle WebLogic Server Directory Structures
- Understanding Oracle WebLogic Server Domains for OC4J Users

3.1.1 Comparing the OC4J and Oracle WebLogic Server Architectures

When comparing the topologies of OC4J and Oracle WebLogic Server, we can consider two basic topologies:

- Standalone OC4J and Standalone Oracle WebLogic Server
- OC4J and Oracle WebLogic Server Integrated With a Web Server
- OC4J and Oracle WebLogic Server Clustering Features

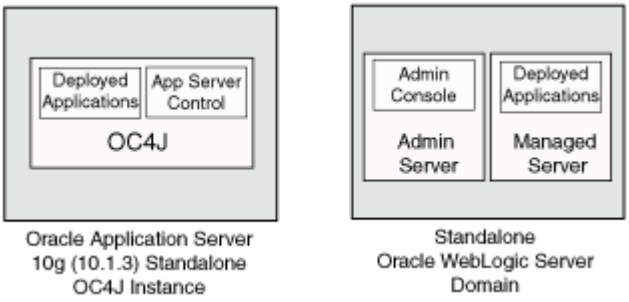
3.1.1.1 Standalone OC4J and Standalone Oracle WebLogic Server

Figure 3–1 compares a typical standalone OC4J instance with a standalone Oracle WebLogic Server domain. In both cases, HTTP requests to each are handled by built-in HTTP listeners.

In Oracle WebLogic Server, the managed servers are similar to the OC4J instances you configured in Oracle Application Server 10*g*. However, Oracle WebLogic Server always provides a dedicated **administration server**, which hosts the Oracle WebLogic Server Administration Console. The Administration Console provides a Web-based management tool for Oracle WebLogic Server, similar to the Application Server Control used to manage OC4J.

Note that you can configure a simple Oracle WebLogic Server development domain with no managed servers. In such an environment, you would deploy your applications to the administration server. However, more typically, the administration server is dedicated to hosting the Administration Console and you deploy your applications to one or more managed servers within the domain.

Figure 3–1 Comparison of Oracle WebLogic Server and OC4J Standalone Architecture

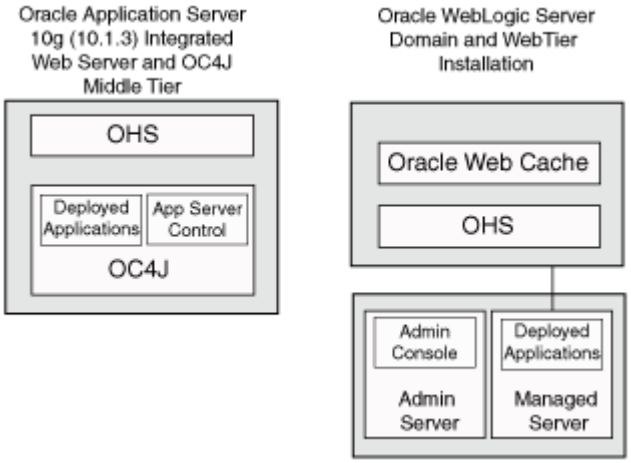


3.1.1.2 OC4J and Oracle WebLogic Server Integrated With a Web Server

Figure 3–2 shows a typical Oracle Application Server 10*g* Release 2 (10.1.2) Integrated Web Server and OC4J middle tier installation and how it compares to a similar topology in Oracle Fusion Middleware 11*g*.

In Oracle Fusion Middleware, you must install and configure a separate Web tier installation, using the installation and configuration tools available on the WebTier and Utilities CD-ROM.

Figure 3–2 Comparison of Oracle WebLogic Server and OC4J with a Front-End Web Server



3.1.1.3 OC4J and Oracle WebLogic Server Clustering Features

Oracle Application Server 10*g* Release 3 (10.1.3) introduced the following concepts related to clustering:

- The Oracle Application Server **cluster topology**, which enables multiple Oracle Application Server instances to be managed from a single, active Oracle Enterprise Manager Application Server Control.

  From the Cluster Topology page in Application Server Control, you can view the multiple Oracle Application Server instances in the cluster topology and perform management tasks on those instances. The different instances within the cluster topology communicate via Oracle Notification Service (ONS).

- **OC4J Groups**, which provides a mechanism for grouping OC4J instances within the cluster topology and performing group-wide tasks on all the OC4J instances at once.

  For example, after you created a group of OC4J instances, you can deploy an application to the group or modify data sources for the group. One key restriction is that each OC4J within the group must be identical in configuration to the other OC4J instances in the group.

- **OC4J Application Clustering**, which is the habilitate to communicate state information among applications deployed to different OC4J instances within the cluster topology.

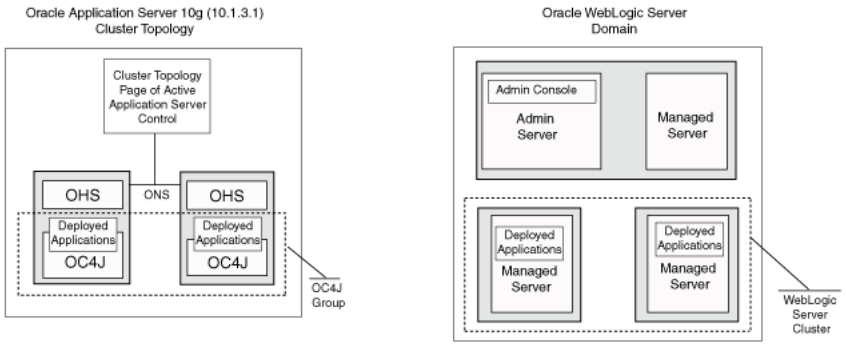
For more information about how these OC4J features compare to Oracle WebLogic Server, refer to the following:

- Table 3–1 compares the OC4J clustering features with those available in Oracle WebLogic Server.
- Figure 3–3 illustrates differences between OC4J and Oracle WebLogic Server clustering.

Table 3–1 Comparing OC4J Clustering Features with Oracle WebLogic Server

OC4J Feature	Oracle WebLogic Server Equivalent Feature	More Information
Oracle Application Server cluster topology	Oracle WebLogic Server domain	"Understanding WebLogic Server Domains" in the *Oracle Fusion Middleware Understanding Domain Configuration for Oracle WebLogic Server*
OC4J groups	Oracle WebLogic Server clusters	"Understanding WebLogic Server Clustering" and "Cluster Architectures" in *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server*
OC4J application clustering	Oracle WebLogic Server HTTP Session State Replication	"HTTP Session State Replication" in *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server*

Figure 3–3 Comparison of OC4J Groups and Oracle WebLogic Server Clusters



3.1.2 Comparing OC4J and Oracle WebLogic Server Directory Structures

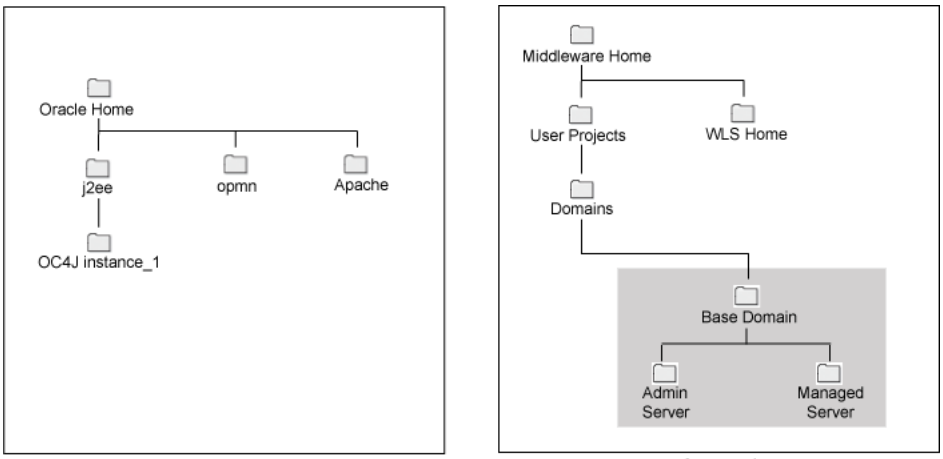
The directory structure of a typical Oracle WebLogic Server domain differs from the directory structure of an Oracle Application Server 10*g* instance in several ways.

When you install Oracle Application Server 10*g*, you create a single Oracle home that contains the `j2ee` directory. The OC4J-specific configuration files and log files reside inside the `j2ee` directory.

In Oracle WebLogic Server, the installation is enclosed within a Middleware home. Within the Middleware home, the installer creates the Oracle WebLogic Server home directory. When you configure a domain, the Oracle WebLogic Server Configuration Wizard creates a new domain inside the `user_projects` directory.

Figure 3–4 shows the difference between the Oracle WebLogic Server and OC4J directory structures.

Figure 3–4 Comparison of the Oracle WebLogic Server and OC4J Directory Structures



Directory Structure for an Oracle Application Server 10g (10.1.3) Web Server and Java EE Installation

Directory Structure for a Typical Oracle WebLogic Server Domain

3.1.3 Understanding Oracle WebLogic Server Domains for OC4J Users

The following sections are designed to help OC4J users learn about the features and capabilities of Oracle WebLogic Server domains:

- Basic Content and Organization of a Domain
- Additional Facts About Oracle WebLogic Server Domains

3.1.3.1 Basic Content and Organization of a Domain

Depending on whether you are using 10*g* Release 2 (10.1.2) or 10*g* Release 3 (10.1.3), you organize your Oracle Application Server 10*g* environment in one of two ways:

- In Oracle Application Server 10*g* Release 3 (10.1.3), you organize your applications servers and OC4J instances into a cluster topology.
- In Oracle Application Server 10*g* Release 2 (10.1.2), you can add multiple Oracle Application Server instances into a farm and your OC4J instances into Oracle Application Server Clusters.

Oracle Fusion Middleware 11*g* uses an entirely different mechanism to organize your environment. The Oracle WebLogic Server environment is grouped into logical groups called domains. These domains consist of the following:

- A single **administration server**, which is used to manage the domain.
- One or more **managed servers**, which are used to deploy the Oracle Fusion Middleware Java components, as well as your custom Java EE applications.

In previous versions of Oracle Application Server 10*g*, you installed, created, and configured OC4J instances, and one or more Java Virtual Machines (JVMs) per OC4J instance.

In Oracle Fusion Middleware 11*g*, you configure an Oracle WebLogic Server domain with one administration server and one or more managed servers to deploy your Java EE applications.

3.1.3.2 Additional Facts About Oracle WebLogic Server Domains

The following sections provide more detailed information about Oracle WebLogic Server domains that can be helpful as you transition from an OC4J environment to an Oracle WebLogic Server environment:

- A WebLogic Server Installation Can be Used to Configure Multiple Domains
- A WebLogic Server Instance Is Always Associated With a Single Java Virtual Machine Process
- A WebLogic Server Instance Processes All Application Requests on the Same Port by Default
- A WebLogic Server Instance Is Always Configured With an HTTP Listener and Does Not Support AJP

3.1.3.2.1 A WebLogic Server Installation Can be Used to Configure Multiple Domains As shown in Figure 3–4, the binary and configuration files associated with the OC4J instances you create are stored in a subdirectory structure inside the Oracle Application Server 10*g* Oracle home (the `j2ee` directory structure). Not only is this direct file system association required, but the relationship between the OC4J binaries and the configuration of instances is at a per instance level.

In contrast, Oracle WebLogic Server provides a clear separation between the installed software and its different configuration instances. A single Oracle WebLogic Server installation can be used to create multiple domains, each with a different set of servers.

By default, the Oracle WebLogic Server configuration wizard assumes that you will place the configuration files for each domain inside the `user_projects/domains` directory. However, the files associated with a domain can reside anywhere within the file system. For a WebLogic Server instance to run from a domain directory, the only requirement is that the Oracle WebLogic Server directory must be accessible.

Note also that in the WebLogic Server model, the relationship between the WebLogic Server binaries and the configuration of instances is at a per domain (set of instances) level.

3.1.3.2.2 A WebLogic Server Instance Is Always Associated With a Single Java Virtual Machine Process OC4J executes on the Java Virtual Machine (JVM) of the standard Java Development Kit (JDK). By default, each OC4J instance uses one JVM. However, you can configure an OC4J instance so it runs on multiple JVMs. You can configure the OC4J instance in this manner using the `numproc` property or by using the Application Server Control console.

When you configure an OC4J instance to run on multiple JVMs, the OC4J instance is essentially running on multiple processes. This can improve performance and provide a level of fault tolerance for your deployed applications. However, multiple JVMs also require additional hardware resources to run efficiently.

There is no equivalent setting or capability in Oracle WebLogic Server. Instead, each server always runs on a single Java Virtual Machine. However, you can obtain the same capability as the OC4J `numproc` setting by increasing the number of managed servers running on the same host and within the domain.

3.1.3.2.3 A WebLogic Server Instance Processes All Application Requests on the Same Port by Default An OC4J instance uses a different set of listen ports for each protocol for which it can accept requests. OC4J uses the concept of OC4J Web sites to configure specific HTTP, HTTPS, AJP, or AJPS ports (or port range) on each OC4J instance. Similarly,

dedicated ports (and port ranges) can be configured on an OC4J instance for RMI and JMS traffic.

The WebLogic Server request port and protocol management model is by default different from OC4J's in two important ways:

- First, an Oracle WebLogic Server instance is configured to have only two listen ports for incoming application requests: one port for accepting non-encrypted requests (which has to be set) and the other for accepting SSL encrypted requests (which is optional).
- Second, these ports are configured to accept requests for all supported protocols as opposed to being dedicated to a specific one as is the case with OC4J instances.

Although in general, this default listening port model is sufficient for a majority of use cases, Oracle WebLogic Server offers a feature called **network channels**, which allows you to configure a WebLogic Server instance with additional ports dedicated to specific protocols. This feature can be used for the special use cases where such a configuration is required.

3.1.3.2.4 A WebLogic Server Instance Is Always Configured With an HTTP Listener and Does Not Support AJP One of the supported OC4J configurations is a topology where a front-end Web server (in most cases, Oracle HTTP Server) is configured to receive incoming HTTP requests. The requests are then routed from Oracle HTTP Server to the appropriate OC4J instance via the AJP protocol. In this configuration, the OC4J instance does not receive HTTP requests directly and no HTTP listener is configured on the OC4J instance.

In contrast, an Oracle WebLogic Server instance must always accept HTTP requests and has no support for AJP. However, WebLogic Server domains can be (and frequently are) fronted by a Web tier for security and scalability purposes. A number of Web servers, including Oracle HTTP Server, are certified to act as a Web tier to servers within a WebLogic Server domain.

For more information, see Chapter 7, "Upgrading a Java EE and Web Server Environment".

3.2 Oracle WebLogic Server Installation and Configuration Tools for OC4J Users

Unlike Oracle Application Server 10*g*, Oracle WebLogic Server separates the tasks of installing and configuring your environment.

In Oracle Application Server 10*g*, you use Oracle Universal Installer to install and configure your Oracle Application Server environment, including the OC4J instances within that environment.

With Oracle WebLogic Server, you use two separate tools to install and configure your environment:

- **The Oracle WebLogic Server installer**, which you use to create the Middleware home and place the necessary Oracle WebLogic Server files on disk, in preparation for configuring your Oracle WebLogic Server domains.
- **The Oracle WebLogic Server Configuration Wizard**, which you use to create and configure your Oracle WebLogic Server domains.

This separation of installation and configuration tasks allows you create multiple domains from a single Oracle WebLogic Server instance, as described in Section 3.1.3.2.1.

3.3 Oracle WebLogic Server Administration Tools for OC4J Users

The following sections compare the administrations tools available for managing OC4J and Oracle WebLogic Server, as well as a summary of how you perform some typical management tasks in Oracle WebLogic Server:

- Comparison of OC4J and Oracle WebLogic Server Administration Tools
- Typical Oracle WebLogic Server Administration Tasks for OC4J Users

3.3.1 Comparison of OC4J and Oracle WebLogic Server Administration Tools

The tools you use to administer an Oracle WebLogic Server domain are different than those you use to administer an OC4J-based environment:

- Table 3–2 compares the equivalent administration tools for the two products.
- Table 3–3 describes additional administration tools not available for OC4J.

For a complete list of the Oracle WebLogic Server administration tools, see "Summary of System Administration Tools and APIs" in *Introduction to Oracle WebLogic Server*.

Table 3–2 Comparison of OC4J and Oracle WebLogic Server Administration Tools

10*g* Release 3 (10.1.3) Administration Tool	**Equivalent Oracle WebLogic Server Administration Tools**	**More Information**
Oracle Enterprise Manager Application Server Control	Oracle WebLogic Server Administration Console Note, that an additional Web-based management tool called Oracle Enterprise Manager Fusion Middleware Control is also available. You use Fusion Middleware Control to manage the farm and the OPMN-managed components of the farm.	"Overview of Oracle Fusion Middleware Administration Tools" in the *Oracle Fusion Middleware Administrator's Guide*
admin_client.jar	WLST (Oracle WebLogic Server command-line scripting tool)	"Getting Started Using the Oracle WebLogic Scripting Tool (WLST)" in the *Oracle Fusion Middleware Administrator's Guide*
opmnctl	opmnctl Note that OPMN is supported in Oracle Fusion Middleware 11*g*, but only for managing specific Oracle Fusion Middleware system components. For other configuration tasks in Oracle Fusion Middleware 11*g*, such as administering the managed servers in a domain, you use WLST.	"Getting Started Using Oracle Process Manager and Notification Server" in the *Oracle Fusion Middleware Oracle Process Manager and Notification Server Administrator's Guide*

Table 3–3 Additional Oracle WebLogic Server Administration Tools

Administration Tool	Description
weblogic.Deployer	Oracle WebLogic Server provides a packaged deployment tool, `weblogic.Deployer`, to provide deployment services for WebLogic Server for ears, wars, rars, jars and other deployment artifact. Any deployment operation that can be implemented using the WebLogic deployment API is implemented, either in part or in full, by `weblogic.Deployer`. This tool provides similar capabilities to OC4J admin_client.jar for Java EE artifact deployment. In earlier OC4J releases deployment to OC4J was provided by admin.jar and the `dcmctl` tooling.
Ant tasks	Oracle WebLogic Server provides a set of administration Ant tasks that allow for the execution of WebLogic Server administrative processes within Apache Ant scripts. These Ant tasks perform a similar type of function as the OC4J Ant tasks.

3.3.2 Typical Oracle WebLogic Server Administration Tasks for OC4J Users

The following sections describe some typical administration tasks and you perform them in Oracle WebLogic Server:

- Starting and Stopping Servers
- Performing Diagnostics on a Domain
- Configuring and Tuning Thread Pools

3.3.2.1 Starting and Stopping Servers

You can start and stop OC4J instances using Application Server Control or the OPMN command line (opmnctl). Similarly, you can start and top WebLogic server instances using the Oracle WebLogic Server Administration Console or the WebLogic Scripting Tool (WLST).

Additionally, WebLogic server instances can also be started through the start-up scripts available in the bin directory of the domain directory. These scripts include the `startWebLogic` script for starting the Administration server and the `startManagedWebLogic` script for starting the managed servers in a domain.

For more information, see *Oracle Fusion Middleware Managing Server Startup and Shutdown for Oracle WebLogic Server*.

3.3.2.2 Performing Diagnostics on a Domain

The diagnostics capabilities of Oracle WebLogic Server are provided through the WebLogic Diagnostics Framework (WLDF). WLDF provides features that meet or exceed the capabilities of the OracleAS Dynamic Monitoring Service (DMS), which was the monitoring server for Oracle Application Server 10*g*.

The WLDF features, which include dynamic code instrumentation, image capture, watches, and notifications, result in extended application diagnostics capabilities which can greatly reduce the total cost of ownership of maintaining Java EE applications.

Furthermore, WLDF capabilities can be exposed as custom dashboards within the Administration Console through the WLDF Console Extension features. Applications that currently use DMS can continue to do so in Oracle Fusion Middleware 11*g*.

Note that if you are using third-party application management tools, your third-party tools can typically be updated and continue operating against the upgraded applications running on Oracle WebLogic Server.

3.3.2.3 Viewing Log Files for a Domain

Oracle WebLogic Server Logging Services provides a comprehensive set of logging features similar to the Oracle Application Server 10*g* logging capabilities. The capabilities of the Oracle Diagnostics Logging (ODL) framework can also been integrated into Oracle WebLogic Server using the Java Required Files (JRF) template available in Oracle Fusion Middleware 11g.

For more information, see Section 5.1.4, "Using the Java Required Files (JRF) Domain Template".

As a result, if an application is using the ODL framework directly for logging, it does not require modification before you deploy it to Oracle WebLogic Server. The JRF ODL integration into WebLogic Server is as follows:

- ODL log messages are sent to a log file that is kept on the file system separate from the Oracle WebLogic Server log files. they are stored in the following location:

  `domain_directory/servers/server_name/logs/server_name-diagnostic.log`

- Critical messages (errors) are logged both in the ODL and Oracle WebLogic Server domain log file.
- ODL log query and configuration JMX MBeans are available from the Administration server of the domain.

3.3.2.4 Configuring and Tuning Thread Pools

OC4J Server instances use different thread pools for different purposes (system, HTTP, and JCA are the default startup thread pools). The parameters of each thread pool (such as maximum and minimum thread counts) can be individually tuned to achieve an optimal application request throughput for a particular environment.

A Oracle WebLogic Server instance has by default a single thread pool, the thread count of which is automatically tuned to achieve maximum overall throughput. All requests are enqueued upon arrival in a common queue and prioritized according to administratively configured goals such as an application's desired response time or its fair-share usage of all available threads relative to other applications.

This Oracle WebLogic Server feature--referred to as WebLogic Work Managers--effectively allows Oracle WebLogic Server instances to self-tune their thread counts for optimal request processing.

For more information, see "Using Work Managers to Optimize Scheduled Work" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*.

3.4 Standards Support for OC4J and Oracle WebLogic Server

Table 3–4 compares the Java standards supported by OC4J and Oracle WebLogic Server.

Table 3–4 Comparison of Java Standards Supported by OC4J and Oracle WebLogic Server

Standard	Version Supported by OC4J	Version Supported by Oracle WebLogic Server
Java SE	6.0	6.0
Java EE	1.4/5.0	5.0
JSP	1.1 to 2.0	1.1 to 2.1
JSF	1.1	1.1 and 1.2

Table 3–4 (Cont.) Comparison of Java Standards Supported by OC4J and Oracle WebLogic Server

Standard	Version Supported by OC4J	Version Supported by Oracle WebLogic Server
Servlet	2.2 to 2.5	2.2 to 2.5
EJB	2.1 and 3.0	2.1 and 3.0
JAX-WS	Not Supported	2.1
JAX-RPC	1.1	1.1
JMS	1.0.2b and 1.1	1.0.2b and 1.1
JNDI	1.2	1.2
JCA	1.5	1.5
JTA	1.1	1.1
JMX	1.2	1.2
Java EE Deployment	1.0	1.2
Java EE Management	1.0	1.1
JDBC	3.0	3.0

Part II

Upgrading Your Java EE Applications and Environment

Part II contains the following chapters:

- Chapter 4, "Upgrading Your Java EE Applications"
- Chapter 5, "Upgrading Your Java EE Environment"
- Chapter 6, "Upgrading Application Clients"
- Chapter 7, "Upgrading a Java EE and Web Server Environment"

4

Upgrading Your Java EE Applications

This chapter summarizes the general tasks you will likely need to perform when upgrading your OC4J applications and redeploying them on Oracle WebLogic Server.

This chapter contains the following sections:

- Task 1: Verify that Your Application Deploys and Works Successfully on OC4J
- Task 2: Select Your Development Tools
- Task 3: Verify That Your Application Supports Java Development Kit (JDK) 6
- Task 4: Upgrade the Application Deployment Descriptors
- Task 5: Review Oracle WebLogic Server API Support
- Task 6: Upgrade the Application Web Services

4.1 Task 1: Verify that Your Application Deploys and Works Successfully on OC4J

Before you upgrade any Oracle Application Server 10*g* applications to Oracle Fusion Middleware 11*g*, you should first make sure that the application is currently deployed and running successfully on Oracle Application Server 10*g*.

Also, make note of any application-specific configuration changes you performed to the OC4J instance where you deployed the application. For example, if the application requires any specific data sources, JMS servers, or other resources, you will need to make similar configuration changes to the Oracle WebLogic Server domain, as described later in this chapter.

4.2 Task 2: Select Your Development Tools

Refer to the following sections for information that can help you select the best tools for developing your Java EE applications for the Oracle WebLogic Server and Oracle Fusion Middleware platform:

- General Guidelines for Selecting Your Development Tools
- Using the SmartUpgrade Oracle JDeveloper Extension and Command-Line Tool

4.2.1 General Guidelines for Selecting Your Development Tools

For Java EE applications that do not take advantage of Oracle technologies, such as the Oracle Application Development Framework (ADF), Oracle Metadata Services (MDS), Oracle SOA Suite, or Oracle WebCenter, you can make the code changes using any development tools you are accustomed to using.

However, if you want to use any Oracle technologies, Oracle recommends the use of Oracle JDeveloper, an integrated development environment (IDE) that makes it much easier and efficient to develop, test, and deploy ADF, SOA, and WebCenter applications for Oracle Fusion Middleware.

4.2.2 Using the SmartUpgrade Oracle JDeveloper Extension and Command-Line Tool

Besides offering the ability to easily develop applications that take advantage of Oracle techologies, such as Oracle ADF, Oracle SOA, and Oracle WebCenter, Oracle JDeveloper also provides an extension that can help you upgrade your OC4J applications to Oracle WebLogic Server.

The SmartUpgrade extension allows you to use Oracle JDeveloper to analyze an existing enterprise archive (EAR) file and generate a SmartUpgrade report that steps you through a series of "findings". Each finding provides advice for how to modfiy the applicatoin so you can successfully deploy it on Oracle WebLogic Server.

For more information, refer to the *Oracle Fusion Middleware SmartUpgrade User's Guide*.

4.3 Task 3: Verify That Your Application Supports Java Development Kit (JDK) 6

Oracle Fusion Middleware 11*g* supports JDK SE 6. Before you redeploy your application on Oracle WebLogic Server, verify that the Java source code in your application is compatible with JDK 6.

For more information, refer to the resources available on the following Sun Microsystems Web site:

`http://java.sun.com/javase/6/`

4.4 Task 4: Upgrade the Application Deployment Descriptors

Both OC4J and Oracle WebLogic Server support not only the Java EE standard deployment descriptors, but also corresponding proprietary descriptors. However, when you redeploy an OC4J application on Oracle WebLogic Server, you must modify the application descriptors, such as `application.xml` and `web.xml`, to comply with the requirements of Oracle WebLogic Server.

Use the following sections to learn more about upgrading your application deployment descriptors:

- Comparison of OC4J and Oracle WebLogic Server Deployment Descriptors
- Guidelines and Resources for Upgrading Deployment Descriptors for Oracle WebLogic Server

4.4.1 Comparison of OC4J and Oracle WebLogic Server Deployment Descriptors

If you are familiar with the standard deployment descriptors and the OC4J-specific descriptors, you can use Table 4–1 to locate the equivalent deployment descriptor files in the Oracle WebLogic Server environment.

Table 4–1 J2EE, OC4J, and WebLogic Server Deployment Descriptors

J2EE Standard Descriptor	OC4J Proprietary Descriptor	WebLogic Proprietary Descriptor
`application.xml`	`orion-application.xml`	`weblogic-application.xml`
`web.xml`	`orion-web.xml`	`weblogic.xml`
`ejb-jar.xml`	`orion-ejb-jar.xml`	`weblogic-ejb-jar.xml`
`application-client.xml`	`orion-application-client.xml`	`weblogic-appclient.xml`
`ra.xml`	`oc4j-ra.xml`	`weblogic-ra.xml`
`webservices.xml`	`oracle-webservices.xml`	`weblogic-webservices.xml`

4.4.2 Guidelines and Resources for Upgrading Deployment Descriptors for Oracle WebLogic Server

To prepare your applications for WebLogic Server deployment, you must remove OC4J-specific deployment descriptors and replace them with their equivalent WebLogic Server specific settings.

For a successful deployment, examine each of the deployment descriptors and perform one the following actions for each deployment descriptor feature used by the application:

- If the OC4J deployment descriptor feature has a direct mapping within the equivalent WebLogic Server specific deployment descriptor, then use the equivalent WebLogic Server descriptor with the appropriate elements and values.
- If the OC4J deployment descriptor feature does not have a direct mapping, then review the appropriate Oracle WebLogic Server documentation.

  Features that are not directly mapped to Oracle WebLogic Server deployment descriptors can often be achieved by configuring the WebLogic domain accordingly.

Table 4–2 provides a list of documentation resources that will help you upgrade your deployment descriptors to Oracle WebLogic Server.

Table 4–2 Documentation Resources for Upgrading Deployment Descriptors to Oracle WebLogic Server

When upgrading to this WebLogic-Specific Deployment Descriptor...	Refer to these documentation resources...
weblogic-application.xml	"Enterprise Application Deployment Descriptor Elements" in *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*
weblogic.xml	Appendix A, "orion-web.xml and orion-ejb-jar.xml Upgrade Reference" provides upgrade advice for each of the deployment descriptor elements in the `orion-web.xml` and `orion-ejb.jar.xml` files.
weblogic-ejb-jar.xml	Appendix A, "orion-web.xml and orion-ejb-jar.xml Upgrade Reference" provides upgrade advice for each of the deployment descriptor elements in the `orion-web.xml` and `orion-ejb.jar.xml` files.
weblogic-appclient.xml	"Client Application Deployment Descriptor Elements" in *Oracle Fusion Middleware Programming Stand-alone Clients for Oracle WebLogic Server*
weblogic-ra.xml	"Configuring the weblogic-ra.xml File" in *Oracle Fusion Middleware Programming Resource Adapters for Oracle WebLogic Server*

Table 4–2 (Cont.) Documentation Resources for Upgrading Deployment Descriptors to Oracle WebLogic

When upgrading to this WebLogic-Specific Deployment Descriptor...	Refer to these documentation resources...
weblogic-webservices.xml	"WebLogic Web Service Deployment Descriptor Element Reference" in *Oracle Fusion Middleware WebLogic Web Services Reference for Oracle WebLogic Server*

4.4.3 About Security Elements in Deployment Descriptor Files

If you use the suggested procedures in Section 5.3.6, "Configuring Security on Oracle WebLogic Server", then the security configurations, including authentication methods, security constraints, and EJB method permissions, that are contained in standard Java EE application deployment descriptors, such as `web.xml` and `ejb-jar.xml`, can remain untouched for upgrade and will continue to function when the application is deployed to Oracle WebLogic Server.

For security configurations specified in OC4J specific descriptors (for example, security role mappings), see the WebLogic Server Security documentation to map each configuration to an element within the equivalent Oracle WebLogic Server deployment descriptor described in Section 4.4.1, "Comparison of OC4J and Oracle WebLogic Server Deployment Descriptors".

4.4.4 Upgrading Deployment Plans

Deployment plans are a standard Java EE server capability supported by both Oracle WebLogic Server and OC4J. However, deployment plans are not portable between application servers. When you upgrade from OC4J to Oracle WebLogic Server, you must regenerate and save your application deployment plans saved as part of the deployment process on Oracle WebLogic Server. Alternatively, you can construct new deployment plans using the `weblogic.PlanGenerator` command line tool.

For more information, see "Overview of weblogic.PlanGenerator" in *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*.

4.5 Task 5: Review Oracle WebLogic Server API Support

Before you can redeploy your Oracle Application Server 10*g* application on Oracle WebLogic Server, you must review your applications to identify the application programming interfaces referenced by the source code.

For more information, refer to the following sections:

- APIs Available With the Java Required Files (JRF) Domain Template
- Other Oracle WebLogic Server API Requirements

4.5.1 APIs Available With the Java Required Files (JRF) Domain Template

Oracle Fusion Middleware11*g* provides an Oracle WebLogic Server domain template, referred to as the Oracle Java Required Files (JRF) template. You can use this template to create (or extend) an Oracle WebLogic Server domain. The resulting domain contains an updated version of some of the key capabilities and features of Oracle Application Server 10*g*.

For more information, see Section 5.1.4, "Using the Java Required Files (JRF) Domain Template".

Specifically, the JRF template enables support for the following Oracle Application Server features in Oracle Fusion Middleware 11*g*:

- Dynamic Monitoring System (DMS)
- Diagnostics and Logging Framework (ODL)
- Oracle HTTP Client
- Oracle Java Object Cache
- Oracle XML
- Oracle Security Developer Tools
- Oracle Platform Security Services (OPSS)
- Oracle Globalization Development Kit

Applications that use these APIs can take advantage of a JRF-extended domain and will not require any modification beyond those necessary as a result of potential updates to the APIs.

For more information, see Chapter 5, "Upgrading Your Java EE Environment".

4.5.2 Other Oracle WebLogic Server API Requirements

Table 4–3 provide a summary of additional Oracle Application Server 10*g* APIs and a summary of how they are affected by an upgrade to Oracle Fusion Middleware 11*g*.

Table 4–3 Other API Changes for Oracle Fusion Middleware 11g

API	Description and Actions Required	More Information
Oracle JAZN (Java Authorization)	The JRF domain template provided with Oracle Fusion Middleware 11*g* provides an updated and equivalent set of features provided by the Oracle Platform Security Services (OPSS) API. Applications that currently use the Oracle JAZN API for security management must be updated to use OPSS so that they can be deployed to a JRF-extended Oracle WebLogic Server domain as part of the upgrade.	"Introduction to Oracle Platform Security Services" in the *Oracle Fusion Middleware Security Guide*
Oracle TopLink	You can continue to use Oracle Toplink by ensuring that the target Oracle WebLogic Server domain is configured to use Oracle TopLink as the JPA persistence provider. Oracle WebLogic Server is certified by Oracle to fully support Oracle TopLink.	"Integrating TopLink with Oracle WebLogic Server" in the *Oracle Fusion Middleware Developer's Guide for Oracle TopLink*
Oracle JSP Tag Libraries (ojsputil.jar, jstl.jar)	You can continue to use these tag libraries by ensuring that the Oracle Application Server 10*g* TLD and JAR files associated with the Oracle JSP tag libraries remain (or if not already there, are placed) in the following directories of the WAR file: ■ `WEB-INF/tld` ■ `WEB-INF/lib`	Not applicable.

Table 4–3 (Cont.) Other API Changes for Oracle Fusion Middleware 11g

API	Description and Actions Required	More Information
Oracle Web Cache Invalidation	The Web Cache Invalidation API can continue to be used by ensuring that the appropriate OracleAS 10g jar file is available to the application when deployed to WebLogic Server. An updated, but fully backward compatible, version of this API is also available in as part of the JRF domain template.	Section 4.5.1, "APIs Available With the Java Required Files (JRF) Domain Template"
OracleAS Web Services Oracle Web Services Proxy Oracle Web Services SOAP Oracle Web Services UDDI Client	Any application using OracleAS Web Services will require modification to use the equivalent set of Oracle Fusion Middleware or WebLogic Server APIs and features. More specifically: ▪ Applications using the OracleAS Web Services, Oracle Web Services Proxy, or Oracle Web Services SOAP API must be modified to use the standard based (JAX-RPC or JAX-WS) Web Services APIs of WebLogic Server. ▪ Applications using the Oracle Web Services UDDI Client API must be modified to use the UDDI v1, v2, or v3 compliant Oracle Service Registry UDDI client API.	*Oracle Fusion Middleware Introducing WebLogic Web Services for Oracle WebLogic Server*
OC4J Job Scheduler	Oracle provides an upgrade and migration path for the OC4J Job Scheduler to the Oracle Fusion Middleware Enterprise Scheduler (ESS).	*Oracle Ultra Search vs Oracle Secure Enterprise Search, Frequently Asked Questions*, which is available in PDF format on the Oracle Technology Network (OTN).
OC4J Support for JSP	Any application using this API will require modification to use the equivalent set of Java Standard Tag Library (JSTL) tags.	*Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*
OC4J JMX MBeans	Any application using the OC4J JMX MBeans directly for application and environment management purposes will require modification to use the equivalent set of WebLogic Server JMX MBeans.	*Oracle Fusion Middleware Developing Manageable Applications With JMX for Oracle WebLogic Server*

4.6 Task 6: Upgrade the Application Web Services

To upgrade your Web services from OC4J to Oracle WebLogic Server, refer to the following sections:

- General Guidelines for Upgrading to Oracle WebLogic Server JAX-RPC and JAX-WS Web Services
- Generating Oracle WebLogic Server Web Services From an OC4J WSDL
- Web Services Specifications Supported by OC4J and Oracle WebLogic Server

4.6.1 General Guidelines for Upgrading to Oracle WebLogic Server JAX-RPC and JAX-WS Web Services

In general, to upgrade your Web services from OC4J to Oracle WebLogic Server, you must upgrade your application to use the equivalent Java Web services API on WebLogic Server:

- For most OC4J Web services applications, this means upgrading from OC4J JAX-RPC Web services to Oracle WebLogic Server JAX-RPC Web services. In general, the JAX-RPC upgrade process consists of re-generating the Java artifacts on Oracle WebLogic Server to the identical underlying Java business logic using Oracle WebLogic Server Web services tooling.
- If you are using an older release of OC4J, where Web services standards did not exist within Java EE, Oracle recommends that you upgrade to the Java EE 5.0 standard JAX-WS on Oracle WebLogic Server.

For more information, see:

- "How Do I Choose Between JAX-WS and JAX-RPC?" in *Oracle Fusion Middleware Introducing WebLogic Web Services for Oracle WebLogic Server*.

4.6.2 Generating Oracle WebLogic Server Web Services From an OC4J WSDL

For absolute fidelity to a specific OC4J Web services public API (its WDSL), you also can regenerate the Web services on WebLogic Server from the OC4J WSDL and deploy the resulting Web service on WebLogic Server. This process is referred to as the "top-down" approach to developing Web services.

After producing the equivalent and deployable Web service artifacts on Oracle WebLogic Server, you can then apply the equivalent quality of service (QOS) capabilities, such as WS-Security and WS-ReliableMessaging as a secondary administrative operation.

4.6.3 Web Services Specifications Supported by OC4J and Oracle WebLogic Server

Table 4–4 compares the Web services standard specifications supported by OC4J and Oracle WebLogic Server. Note that Oracle WebLogic Server supports all of the Web services standards and specifications supported by OC4J, except for WS-Reliability.

Table 4–4 Web Services Specifications Supported by OC4J and Oracle WebLogic Server

Web Services Specification	OC4J Support	Oracle WebLogic Server Support
SOAP 1.1 and 1.2	Yes	Yes
WSDL 1.1	Yes	Yes
WS-I 1.0 and 1.1	Yes	Yes
XML Signature	Yes	Yes
XML Encryption	Yes	Yes
SAML	Yes	Yes
WS-Addressing	Yes	Yes
WS-Security	Yes	Yes
WS-Reliability	Yes	No
WS-SecurePolicy	No	Yes
WS-Policy	No	Yes
WS-PolicyAttachment	No	Yes
WS-Trust	No	Yes
WS-Conversation	No	Yes

Table 4–4 (Cont.) Web Services Specifications Supported by OC4J and Oracle WebLogic Server

Web Services Specification	OC4J Support	Oracle WebLogic Server Support
WS-SecureConversation	No	Yes
WS-ReliableMessaging	No	Yes

5

Upgrading Your Java EE Environment

This chapter describes how to upgrade a basic Java EE environments. However, you can use these instructions to develop an understanding of the upgrade process and apply this knowledge in your planning of other upgrade scenarios.

Upgrading a basic Java EE environment involves the following key tasks:

- Task 1: Install and Configure an Oracle WebLogic Server Development Domain
- Task 2: Verify the New Oracle Fusion Middleware 11g Environment
- Task 3: Configure Oracle WebLogic Server Resources to Support Your Applications
- Task 4: Redeploy the Application on Oracle WebLogic Server
- Task 5: Verify the Redeployed Applications

5.1 Task 1: Install and Configure an Oracle WebLogic Server Development Domain

To test and verify your upgraded Java EE applications, you must install Oracle WebLogic Server. The following sections describe information about installing and configuring Oracle WebLogic Server:

- Differences Between a Development Environment and a Test or Production Environment
- Installing and Configuring a Development Domain with Oracle JDeveloper
- Installing and Configuring a Development Domain with Oracle SOA Suite, WebCenter, or Application Developer
- Using the Java Required Files (JRF) Domain Template

5.1.1 Differences Between a Development Environment and a Test or Production Environment

When you are developing and upgrading your Java EE applications, you will likely want to install a basic Oracle WebLogic Server environment that you can use for testing your applications quickly and efficiently. This will help you frequently deploy and test your applications as you make required code changes.

This environment differs from your test or production environment in the following ways:

- A development environment is typically a single-node environment. There is no need for a Web tier or clustering capabilities to provide load-balancing or high availability. However, the development must include the various resources (such

as JDBC data sources, JMS providers, and database instances) required to support the application during development.

Use the instructions in this chapter to configure a development environment.

- The Oracle WebLogic Server domain you create in your development environment can be configured in development mode, which makes it quicker and easier to configure the resources of the domain and to deploy and redeploy applications to test code changes. You select development mode or production mode when you create your domain using the Oracle WebLogic Server Configuration Wizard.

  For more information, see "Examples: Using the Configuration Wizard" in *Oracle WebLogic Server Creating WebLogic Domains Using the Configuration Wizard*.

  Note that the default tuning parameters of Oracle WebLogic Server are different, depending upon whether you are configuring a development or production domain. For more information, see "Development vs. Production Mode Default Tuning Values" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

5.1.2 Installing and Configuring a Development Domain with Oracle JDeveloper

Oracle WebLogic Server is available as part of Oracle JDeveloper Studio. As a result, if you are using Oracle JDeveloper Studio as your integrated development environment (IDE), you can quickly and easily install and configure an Oracle WebLogic Server development domain as part of the Oracle JDeveloper installation.

The Oracle WebLogic Server domain you create with Oracle JDeveloper can useful as a local development environment where you can test your applications quickly and easily. You can also apply the Java Required Files (JRF) template to the development domain, which will enable you to develop and test applications that take advantage of Oracle technologies, such as the Oracle Application Development Framework.

Note, however, that an Oracle WebLogic Server domain you create with the Oracle JDeveloper installer does have some limitations. For example:

- It is not designed to be associated with the Oracle Fusion Middleware Web tier components.
- It cannot be configured to include Oracle Enterprise Manager Fusion Middleware Control.

For complete instructions for installing and configuring Oracle WebLogic Server with Oracle JDeveloper Studio, see "Installing the Oracle JDeveloper Studio Edition" in *Oracle Fusion Middleware Installation Guide for Oracle JDeveloper*.

5.1.3 Installing and Configuring a Development Domain with Oracle SOA Suite, WebCenter, or Application Developer

As an alternative to installing and configuring an Oracle WebLogic Server as part of a Oracle JDeveloper installation, you can install Oracle WebLogic Server from the Oracle WebLogic Server CD-ROM, which is part of the Oracle Fusion Middleware 11*g* Media pack, or by downloading Oracle WebLogic Server from the Oracle Technology Network (OTN).

Refer to the following sections for more information:

- Advantages of Installing an Oracle SOA Suite, WebCenter, or Application Developer Development Environment
- Selecting an Oracle Fusion Middleware Software Suite

- Steps Required to Install and Configure an Oracle SOA Suite, WebCenter, or Application Developer Domain

5.1.3.1 Advantages of Installing an Oracle SOA Suite, WebCenter, or Application Developer Development Environment

Using this alternative, you can:

- Install your test environment on a separate host and then configure your local copy of Oracle JDeveloper to connect to and deploy to the remote environment.
- Ensure that the Oracle Application Development Framework runtime software is available in your test domain.
- Take advantage of Oracle Enterprise Manager Fusion Middleware Control, which is not available as part of a domain configured with Oracle JDeveloper.
- Use other integrated development environments besides Oracle JDeveloper to build and test your applications.

5.1.3.2 Selecting an Oracle Fusion Middleware Software Suite

You can choose from several different Oracle Fusion Middleware software suites, which offer a variety of runtime environments, depending on the types of applications you plan to deploy.

In particular, you can choose from the following Oracle Fusion Middleware software suite:

- Application Developer, which you can use to install and configure Oracle WebLogic Server domains that take advantage of the Oracle Application Development Framework (Oracle ADF) and Oracle Enterprise Manager Fusion Middleware Control.
- Oracle SOA Suite, which provides runtime technologies required to support Oracle SOA Suite applications you develop with Oracle JDeveloper, as well as Oracle ADF and Fusion Middleware Control.
- Oracle WebCenter, which provides runtime technologies for WebCenter applications you develop with Oracle JDeveloper, as well as Oracle ADF and Fusion Middleware Control.

5.1.3.3 Steps Required to Install and Configure an Oracle SOA Suite, WebCenter, or Application Developer Domain

The following is a summary of the steps for installing and configuring the domain.

Note that the procedures described in this section assume you have downloaded the latest version of Oracle WebLogic Server. For more information, refer to "Obtaining the Latest Oracle WebLogic Server and Oracle Fusion Middleware 11*g* Software" in the *Oracle Fusion Middleware Upgrade Planning Guide*.

1. Use the Oracle WebLogic Server installer to install the Oracle WebLogic Server software on disk and to create the Middleware home.
2. Install the Oracle SOA Suite, WebCenter, or Application Developer Oracle home inside the Middleware home.
3. Apply any required patches to the Oracle WebLogic Server or Oracle Fusion Middleware home.
4. Use the Oracle Fusion Middleware Configuration Wizard to configure the domain.

While using the wizard, verify that the Oracle ADF and Enterprise Manager templates are selected.

5. Start the domain.

For complete information on installing and configuring an Oracle SOA Suite, WebCenter, or Application Development environement, see the one of the following installation guides:

- *Oracle Fusion Middleware Installation Guide for Oracle SOA Suite*
- *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*
- *Oracle Fusion Middleware Installation Guide for Application Developer*

5.1.4 Using the Java Required Files (JRF) Domain Template

When you configure Oracle WebLogic Server, you configure each domain using domain templates. One of the domain templates available with Oracle Fusion Middleware 11*g* is the Java Required Files (JRF) template.

The JRF template provides important Oracle libraries and other capabilities that support new versions of APIs that many OC4J applications depend upon.

For information on the types of APIs in the JRF template that are important to upgraded OC4J applications, see Section 4.5.1, "APIs Available With the Java Required Files (JRF) Domain Template".

To create or extend a domain using the JRF template, refer to the following:

- Creating a New Domain With the JRF Template
- Extending an Existing Domain With the JRF Template

5.1.4.1 Creating a New Domain With the JRF Template

There are two ways to create a new domain using the JRF template:

- Install and configure a development domain using the Oracle JDeveloper 11*g* installer.

  The resulting domain is automatically created using the JRF template.

  Note: You cannot configure Oracle Enterprise Manager Fusion Middleware Control in an Oracle WebLogic Server domain created with Oracle JDeveloper.

- Install and configure an Application Developer, Oracle SOA Suite, or Oracle WebCenter domain.

  When you configure any Oracle Fusion Middleware software suite, you have the option of selecting the JRF template while running the configuration tool.

  You also have the option of selecting the Oracle Enterprise Manager template, which allows you to manage the domain with Oracle Enterprise Manager Fusion Middleware Control.

  For more information, refer to the appropriate Oracle Fusion Middleware installation guide.

5.1.4.2 Extending an Existing Domain With the JRF Template

To extend an existing domain with the JRF template, use one of the following options:

- Use the Oracle JDeveloper 11*g* installer.

  In the Oracle JDeveloper installer, select a custom installation and select the ADF Runtime component. This step allows you to install the ADF runtime jar files and domain templates to the server environment.

  Note: if you use Oracle JDeveloper to install and configure a Oracle WebLogic Server domain and to apply the JRF template, Oracle Enterprise Manager Fusion Middleware Control cannot be configured in the domain.

  For more information, see "Creating and Extending WebLogic Domains" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

- Run the Oracle WebLogic Server configuration wizard from an Application Developer, Oracle SOA Suite, or Oracle WebCenter Oracle home.

  Select the option to extend the domain, and then select the JRF template when prompted with the list of available templates.

  You can also choose to apply the Oracle Enterprise Manager template, which provides you with the ability to use Fusion Middleware Control to manage the domain.

  For more information, refer to the appropriate Oracle Fusion Middleware installation guide.

- Use Fusion Middleware Control or the `ApplyJRF` WebLogic Scripting Tool (WLST) command to apply the JRF template to an existing WebLogic server instance.

  For more information, see "Applying Java Required Files to a Managed Server or Cluster," in the *Oracle Fusion Middleware Administrator's Guide*.

5.2 Task 2: Verify the New Oracle Fusion Middleware 11*g* Environment

To verify that the new Oracle Fusion Middleware environment is installed and configured and ready to use, do the following:

1. Log in to the Oracle WebLogic Administration Console, using the following URL and the weblogic administration credentials you provided during the configuration:

   `http://node_name.domain.com:7001/console`

2. In the left pane of the Console, select **Environment** and then select **Servers**.
3. Review the servers that were created as part of your domain and verify that the servers are up and running.

5.3 Task 3: Configure Oracle WebLogic Server Resources to Support Your Applications

After you have modified your applications, you must then ensure that any services required by the application are configured on the Oracle WebLogic Server domain.

The following sections provide information on the typical Oracle WebLogic Server administration tasks that are required before you deploy an application you are upgrading from Oracle Application Server 10*g*:

- Configuring JDBC Data Sources on Oracle WebLogic Server
- Configuring OC4J JMS Resources on Oracle WebLogic Server
- Configuring OC4J Remote JMS Resources on Oracle WebLogic Server
- Using Shared Libraries and Class Loading on Oracle WebLogic Server
- Configuring Security on Oracle WebLogic Server
- Configuring Logging on Oracle WebLogic Server

5.3.1 Configuring JDBC Data Sources on Oracle WebLogic Server

The following sections provide information about upgrading OC4J JDBC data sources to Oracle WebLogic Server:

- General Information About Defining Data Sources for OC4J and Oracle WebLogic Server
- Upgrading Application-Level OC4J Data Sources
- Upgrading Instance and Group-Level OC4J Data Sources
- JDBC Connection Pools and Managed Data Sources in OC4J and Oracle WebLogic Server

5.3.1.1 General Information About Defining Data Sources for OC4J and Oracle WebLogic Server

In general, you can create the equivalent OC4J data source configuration on WebLogic Server, based on the database connection information, pooling requirements and JDBC driver.

When you create a data source for an application deployed on OC4J, you can define the data source in one of two ways:

- For a specific OC4J instance or OC4J group where the application will be deployed.
- Package the data source definition as part of the application archive in a file named `data-sources.xml`.

Both these methods are supported by Oracle WebLogic Server, but you must perform some configuration tasks, either on the Oracle WebLogic Server domain or within the application, depending on the method you use.

Just as in OC4J, a WebLogic Server JDBC data source is an object bound to a JNDI context which provides database connectivity through a pool of JDBC connections. Applications look up a data source in the JNDI context in order to use a database connection from this pool.

5.3.1.2 Upgrading Application-Level OC4J Data Sources

Oracle WebLogic Server data sources are typically defined at domain level and applied across the cluster or to specific managed servers within a domain. However, if you have defined your OC4J data sources in the OC4J-supported `data-sources.xml` file of your application archive, then you can implement a similar configuration in Oracle WebLogic Server.

For Oracle WebLogic Server, data sources can be packaged as a JDBC module within the application. This provides the equivalent capability of application-level data sources in OC4J.

For more information, see "Configuring JDBC Application Modules for Deployment" in *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*.

5.3.1.3 Upgrading Instance and Group-Level OC4J Data Sources

If you have defined data sources for your OC4J instance or group, you must define an equivalent set of data sources in your new Oracle WebLogic Server domain.

For each JDBC data source configured within the OC4J environment, create a new Oracle WebLogic Server JDBC data source with the same JNDI name in the target domain.

For more information, see "Create a JDBC Data Source" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

Note that Oracle WebLogic Server JDBC data sources can be configured to take advantage of Oracle Real Application Clusters (RAC). For more information, see the *Oracle Fusion Middleware High Availability Guide*.

5.3.1.4 JDBC Connection Pools and Managed Data Sources in OC4J and Oracle WebLogic Server

There are two important differences between OC4J and Oracle WebLogic Server JDBC data source connection pooling:

- Oracle WebLogic Server JDBC data sources have an implicit connection pool associated with them and, therefore, you don not have to create an explicit connection pool in the domain as you do in the OC4J environment.
- Oracle WebLogic Server JDBC data sources always behave like managed Oracle data sources; there is no equivalent to OC4J native data sources.

5.3.2 Configuring OC4J JMS Resources on Oracle WebLogic Server

The following sections provide information on upgrading your OC4J JMS resources to Oracle WebLogic Server:

- Overview of JMS Support in OC4J and Oracle WebLogic Server
- Creating and Managing JMS Resources in OC4J and Oracle WebLogic Server

5.3.2.1 Overview of JMS Support in OC4J and Oracle WebLogic Server

OC4J provided JMS support via a set of services called Oracle Enterprise Messaging Service (OEMS).

OEMS provides a messaging platform for building and integrating distributed applications. It provides the framework for Oracle messaging and message integration solutions, and is based on industry standards, such as the Java Message Service (JMS) and J2EE Connector Architecture (J2CA).

OEMS supports three types of JMS provider persistence models:

- In-memory
- File-based
- Oracle DB Advanced Queuing (AQ)

Oracle WebLogic Server provides direct equivalents for the in-memory and file-based JMS providers.

For more information, see "Overview of JMS and WebLogic Server" in *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

5.3.2.2 Creating and Managing JMS Resources in OC4J and Oracle WebLogic Server

In OC4J, you configure JMS connection factories and destinations for a JMS server on an individual OC4J instance. The connection factories and destinations are then mapped to resource providers or JMS connectors.

In WebLogic Server, you create JMS resources within an WebLogic JMS module. JMS modules are targeted to a WebLogic JMS Server within a domain. WebLogic JMS servers provide a central point which allows for the configuration of message persistence, durable subscribers, message paging, and quotas for their targeted JMS destinations.

To upgrade the JMS configuration in your OC4J environment to Oracle WebLogic Server:

1. Create a set of WebLogic JMS servers with configurations that reflect the OC4J environment's JMS resource providers, connectors, connection factories and destination configurations.
2. Create a WebLogic Server JMS module for each set of JMS connection factories and destinations with common configurations.
3. Populate the module with JMS connection factories and destinations that have the same JNDI name as their equivalent version in OC4J.
4. Finally, target the JMS modules to the appropriate WebLogic JMS server within the domain.

For more information, see *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

5.3.3 Configuring OC4J Remote JMS Resources on Oracle WebLogic Server

In OC4J, you configure remote destinations and connection factories for third-party JMS providers such as WebSphereMQ, Tibco, and SonicMQ as part of a JMS connector configuration.

In Oracle WebLogic Server, you access remote destinations through the WebLogic Server Foreign Server resources, which enable users to integrate external JMS providers with WebLogic Server. The Foreign Server resources provide a mapping between a domain's JNDI tree and external remote JNDI names of JMS destinations and connection factories.

To upgrade an OC4J external JMS provider configuration to an Oracle WebLogic Server domain, create a JMS module that contains a foreign server. Then create a set of foreign connection factories and foreign destinations that can serve as a proxy to the remote destinations that need to be accessed from the domain.

For more information, see "Configuring Foreign Server Resources to Access Third-Party JMS Providers" in the *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

5.3.4 Using Shared Libraries and Class Loading on Oracle WebLogic Server

When upgrading to Oracle WebLogic Server, you can construct an Oracle WebLogic Server target environment that uses a class-loading configuration similar to the one used in the OC4J source environment.

However, due to the differences between the OC4J and Oracle WebLogic Server class loading models, it is important to develop a good understanding of Oracle WebLogic Server application class loading prior to setting up the target Oracle WebLogic Server configuration.

Table 5–1 summarizes the options available in Oracle WebLogic Server for application developers who used the class loading configurations available in the OC4J environment. The table provides a high level mapping of the main OC4J approaches to making a class available to an application to the most comparable way of achieving the same outcome within a WebLogic Server environment.

For more information, see "Creating Shared Java EE Libraries and Optional Packages" in *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*.

Table 5–1 Comparison of OC4J and Oracle WebLogic Server Class-Loading Models

OC4J Approach	Comparable Oracle WebLogic Server Approach
Class is made available to the application's class loader through Oracle-specific `<library>` element of the deployment descriptor.	Add the class or JAR file to the application's `APP-INF/classes` or `APP-INF/lib` directories, respectively. For Web applications, add the class or JAR files to the application's `WEB-INF/classes` or `WEB-INF` directories respectively.
Class is exposed to specific applications as an OC4J shared library.	Deploy the JAR file to the Oracle WebLogic Server instance or cluster as a WebLogic Server shared library. Note that there are some important differences in the concept of shared libraries between OC4J and WebLogic: ▪ First, WebLogic Server shared libraries must be referenced from the applications using them; there is no way of forcing all deployed applications to use a shared library. (See the information below for information on how to achieve this in an Oracle WebLogic Server domain.) ▪ Second, this referencing essentially exports the content of the shared library to the application's class loader's classpath, as opposed to making it available within a dedicated class loader--as a child of the system class loader--as is the case in OC4J. ▪ Finally, WebLogic Server shared libraries can be an EAR, WAR, or JAR file and the scope of their inclusion within the application is controlled by the scope of the deployment descriptor (`weblogic-application.xml` or `weblogic.xml`) that references the archive.
Class is exposed to all applications on all OC4J instances by either referencing the class in the default application's `application.xml` within the Oracle home or by dropping of the JAR file into the *`ORACLE_HOME`*`/applib` directory.	Place the JAR file in the domain directory's `/lib` sub-directory. This will ensure that the JAR file's class is available (within a separate system level classloader) to all applications running on WebLogic Server instances in the domain.

Table 5–1 (Cont.) Comparison of OC4J and Oracle WebLogic Server Class-Loading Models

OC4J Approach	Comparable Oracle WebLogic Server Approach
Class is added to the classpath of the OC4J instance and made available to the entire server instance through the system class loader.	Configure the Oracle WebLogic Server instance so either the `POST_CLASSPATH` or `PRE_CLASSPATH` environment variables are set prior to server start-up.

5.3.5 Configuring Startup and Shutdown Classes

Any startup or shutdown class configured within the source OC4J environment should be converted to a set of Oracle WebLogic Server startup or shutdown classes. Each class must then be configured within the target Oracle WebLogic Server domain and targeted to the Oracle WebLogic Server instances corresponding to the associated OC4J instances in the source environment.

Unlike OC4J startup and shutdown classes, an Oracle WebLogic Server startup or shutdown class does not require any specific interface or provide pre-deployment or post-deployment methods. Instead, you implement the custom logic within the standard `main()` method of the class.

WebLogic Server allows for pre-deployment and post-deployment execution of this logic by providing configuration parameters, which must be set accordingly when configuring a domain with the startup or shutdown class.

To convert an OC4J startup or shutdown class, it might therefore be necessary to create two WebLogic Server startup and shutdown classes:

- One that contains the code from the original class pre(Un)deploy method
- One that contains the code from the post(Un)deploy method.

Although pre-configured parameters can be passed to the `main()` method of a WebLogic Server startup or shutdown class, Oracle WebLogic Server startup classes have no access to arguments in the way that JNDI context and configuration hash table parameters are passed to an OC4J startup class.

If the custom logic within the startup class makes use of these parameters, then this logic should be modified to obtain the JNDI context from scratch and access the server configuration through the Oracle WebLogic Server JMX interfaces.

For more information, see the following:

- "Programming Application Life Cycle Events" in *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*
- "Configure startup classes" and "Configure shutdown classes" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*

5.3.6 Configuring Security on Oracle WebLogic Server

To support the security requirements of your application, you must map the security features of OC4J to the equivalent security features in Oracle WebLogic Server.

Table 5–2 describes how specific OC4J security configurations can be mapped to a WebLogic Server environment.

Table 5–2 Comparison of OC4J and Oracle WebLogic Server Security Features

For this OC4J Security Feature...	Perform the following task in Oracle WebLogic Server...	More Information
Users and groups are stored in the `system-jazn-data.xml` file.	Move the user and group information contained in the `system-jazn-data.xml` file should be moved to the embedded LDAP server.	"Managing the Embedded LDAP Server" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*
OC4J is configured to use an external LDAP provider.	Configure the Oracle WebLogic Server domain with the same LDAP server as you were using for OC4J.	"Configuring LDAP Authentication Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*
Users are authenticated against a database.	Configure an RDBMS authentication provider, which can be one of three types: ■ SQL Authenticator ■ Read-only SQL Authenticator ■ Custom RDBMS Authenticator	"Configuring RDBMS Authentication Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*
OC4J environment is configured with Java single sign-on or subject propagation between multiple OC4J server instances.	WebLogic Server single sign-on and subject propagation are automatic across the server and clusters within a domain and therefore no special configuration is required.	Not applicable.
OC4J environment is configured with custom JAAS login modules.	Create an Oracle WebLogic Server authentication provider within the target domain, either out-of-the-box or a custom provider which wraps the JAAS login module functionality.	"Configuring Login Modules" in the *Oracle Fusion Middleware Security Guide* "Configuring Authentication Providers" in the *Oracle Fusion Middleware Securing Oracle WebLogic Server*
OC4J environment is configured with Oracle Access Manager.	Configure the Oracle WebLogic Server domain to use Oracle Access Manager.	"Integrating the Security Provider for WebLogic SSPI" in the *Oracle Access Manager Integration Guide* in the Oracle Identity Management 10*g* (10.1.4) Identity Management instancedocumentation library on the Oracle Technology Network (OTN).
OC4J server instances are configured with SSL encryption.	Configure the Oracle WebLogic Server domain to use SSL.	"Configuring SSL" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*
OC4J environment uses Oracle Wallet to store security keys.	Store your security keys in a JKS key store in the WebLogic Server domain.	"Configuring Identity and Trust" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*

5.3.7 Configuring Logging on Oracle WebLogic Server

WebLogic Logging Services provides a comprehensive set of logging features that provide capabilities similar to OC4J. As with OC4J, the Oracle Diagnostics Logging (ODL) framework can be integrated into Oracle WebLogic Server through the Oracle Java Required Files (JRF) domain template.

For more information, see Section 5.1.4, "Using the Java Required Files (JRF) Domain Template".

As a result, if an application is using the ODL framework for logging, it requires no modification when deployed to Oracle WebLogic Server. The JRF ODL integration into Oracle WebLogic Server is as follows:

- ODL log messages are sent to a separate log file that is kept in a well-known location on the file system:

  domain_directory/servers/*server_name*/logs/*server_name*-diagnostic.log

- Critical messages (errors) are double-logged both in the ODL and WebLogic domain log file.
- ODL log queries and configuration JMX MBeans are available in the domain's WebLogic administration server.

For more information, see *Oracle Fusion Middleware Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server*.

5.4 Task 4: Redeploy the Application on Oracle WebLogic Server

After you have compiled your application successfully, you can then deploy the application on the Oracle WebLogic Server environment you installed and configured earlier.

You can redeploy your Java EE applications using any of the following typical tools:

- Apache Ant
- WLST, the Oracle WebLogic Server scripting tool
- The Oracle WebLogic Administration Console

For more information, see *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*.

5.5 Task 5: Verify the Redeployed Applications

After you have deployed your Java EE applications on Oracle WebLogic Server, you can verify the applications by doing the following:

- Log in to the Oracle WebLogic Administration Console and review the deployments on the domain. You can also perform various monitoring tasks and post-deployment tasks from the console.
- Navigate in your browser to the application URL and verify that the features of the application are working as they did when you verified them on OC4J earlier in this procedure.

If find any problems with the application, review the domain log files to diagnose the problem. For more information, see "Configuring Log Files and Filtering Log Messages" in the Oracle WebLogic Server documentation library.

6 Upgrading Application Clients

When you upgrade your Java EE applications to Oracle WebLogic Server and Oracle Fusion Middleware 11*g*, the external interfaces exposed by your applications can be affected. In turn, client applications that depend on those interfaces can be affected.

The following sections describe the ramifications of upgrade on application clients, as well as guidelines for addressing any the resulting client issues:

- Impact of Upgrade on Java Server Pages and Servlet Clients
- Impact of Upgrade on Java Naming and Directory Interface Clients
- Impact of Upgrade on Enterprise Java Bean Clients
- Impact of Upgrade on JMS Clients

6.1 Impact of Upgrade on Java Server Pages and Servlet Clients

When an application is upgraded to WebLogic Server, JSP and servlet clients can be affected because of differences in the HTTP session state replication model between Oracle WebLogic Server and OC4J.

Unlike OC4J clusters, which can support any number of in-memory replicated copies of the HTTP session state, Oracle WebLogic Server in-memory HTTP session state replication supports only a primary-secondary, two-copy model.

In most cases, this difference should have no impact on JSP and Servlet clients; however, for rare cases where an application might explicitly rely on more than two copies of the HTTP session state to be available for its clients, consider using Oracle Coherence.

For more information, refer to the information about Oracle Coherence on the Oracle Technology Network (OTN):

6.2 Impact of Upgrade on Java Naming and Directory Interface Clients

The following sections describe considerations for clients of upgraded applications that use the OC4J Java Naming and Directory Interface (JNDI) provider:

- Modifying Clients to Use the Oracle WebLogic Server JNDI Provider
- Understanding the Scope of the Oracle WebLogic Server JNDI Namespace

6.2.1 Modifying Clients to Use the Oracle WebLogic Server JNDI Provider

If any clients of your upgraded applications use the OC4J Java Naming and Directory Interface (JNDI) provider to lookup application interfaces or resources, then you must modify those clients so they use the Oracle WebLogic Server JNDI provider instead.

You can change the application's JNDI initial context creation code as follows:

1. Identify all instances of the OC4J JNDI URLs in the client code.

   Typically the OC4J URL is structured in the following format:

   prefix://*host*:*RMI_or_OPMN_request_port*:*oc4j_instance*/*application-name*

   An example URL for an Oracle Application Server 10*g* installation with an OC4J instance named `oc4j1` and a deployed application called `myapplication` would be as follows:

   ```
   opmn:ormi://127.0.0.1:6003:oc4j1/myapplication
   ```

   Note that the prefix can be `opmn:ormi` for a full Oracle Application Server installation that is using the Oracle Process Management and Notification infrastructure, or it can be just `ormi:` if you are using a standalone OC4J installation.

2. Change the URL of the provider so it points to the target WebLogic Server domain's administration server using the `t3` protocol.

   For example:

   ```
   t3://127.0.0.1:7001
   ```

3. Make sure the security credentials are valid within the target Oracle WebLogic Server domain.

4. Change the initial context factory to the Oracle WebLogic Server `WLInitialContextFactory` class.

   This class should also be made available to the client application's class loader through a Oracle WebLogic Server client jar file. You create the client jar file (`wlfullclient.jar`) using the WebLogic JarBuilder tool.

   For more information, see the following sections in *Oracle Fusion Middleware Programming Stand-alone Clients for Oracle WebLogic Server*:

   - "Understanding the WebLogic Full Client"
   - "Using the WebLogic JarBuilder Tool"

Note that if the client is itself running within an OC4J server instance, the `environment-naming-url-factory-enabled` attribute in the server's `server.xml` may have to be set to true to allow the use of multiple JNDI providers within the same OC4J instance.

6.2.2 Understanding the Scope of the Oracle WebLogic Server JNDI Namespace

Another important difference between the OC4J and Oracle WebLogic Server JNDI providers that might impact client applications is the scoping of JNDI namespaces.

OC4J JNDI objects can have an explicit application scope. Therefore, when performing a lookup, OC4J JNDI clients can use a URL which identifies a specific OC4J server instance and includes the name of the target application.

The Oracle WebLogic Server JNDI objects on the other hand always have a global namespace. Therefore, a Oracle WebLogic Server JNDI client performing a lookup cannot specify a URL identifying a target application explicitly.

As a result, as part of the upgrade to WebLogic Server, you must ensure that the JNDI name of all JNDI resources deployed to the same WebLogic Server domain are unique, regardless of the application to which they belong. If necessary, JNDI clients must be modified to use their target object's unique JNDI name.

6.3 Impact of Upgrade on Enterprise Java Bean Clients

There are two cases where the upgrade of an application to Oracle WebLogic Server could have impact an impact on EJB client applications. Refer to the following sections for more information:

- Impact on Remote Standalone EJB Clients
- Impact on Clients That Use OC4J-Based EJB Interfaces

6.3.1 Impact on Remote Standalone EJB Clients

EJB clients in this category are either stand-alone or deployed to an OC4J server.

For this category of EJB clients, you must modify the client to use the Oracle WebLogic Server JNDI provider, as described in Section 6.2.1, "Modifying Clients to Use the Oracle WebLogic Server JNDI Provider".

The use of the WebLogic Server JNDI provider will lead to the client application obtaining WebLogic Server EJB client stubs that can potentially impact the client application as follows:

- **RMI Protocol:** Client applications must use one of the Oracle WebLogic Server RMI transport protocols, rather than the OC4J RMI transport protocol, ORMI.

  The default WebLogic RMI transport protocol is the Oracle WebLogic Server `T3` protocol, but you can also use IIOP.

- **Load Balancing:** In OC4J, client EJB requests can be configured to be load-balanced through the `InitialContext` JNDI object (random or sticky) across the OC4J cluster for each invocation of `Context.lookup()`.

  In Oracle WebLogic Server, EJB client request load-balancing is handled automatically by remote EJB client stubs. The load-balancing behavior of these stubs is configured through `weblogic-ejb-jar.xml` deployment descriptor configurations and can be set to occur at `InitialContext` creation or EJB method invocation time.

Note that the considerations mentioned here apply to either stand-alone clients or clients currently running within an OC4J server instance, and also those being upgraded to run within a WebLogic Server instance.

For EJB clients that are not deployed as stand-alone applications and that will continue running within an OC4J server instance and making remote invocations to an upgraded WebLogic Server EJB application, the following two additional implications of an upgrade should also be considered:

- First, the application's security context will not be automatically propagated. If this security propagation is necessary, the client will require modification in order to explicitly use the existing security context's credentials at the creation of the WebLogic Server JNDI initial context.

- Second, JTA transaction propagation and XA recovery within the context of the remote EJB invocations will not be possible and if needed the client application itself will require upgrade.

6.3.2 Impact on Clients That Use OC4J-Based EJB Interfaces

Applications that you upgrade might act as remote clients to EJB components that will continue running within an OC4J server. You must modify clients in this category to use the OC4J `RMIInitialContextFactory` JNDI initial context factory located in the `oc4jclient.jar` file.

The `oc4jclient.jar` file must be available to the Oracle WebLogic Server class loaders for the application. For more information, see Section 5.3.4, "Using Shared Libraries and Class Loading on Oracle WebLogic Server".

Note that the propagation of security context requires the configuration of the target OC4J server instance as a Oracle WebLogic Server SSL client. Furthermore, JTA transaction propagation and XA recovery within the context of the remote EJB invocations will not be possible. If these features are required, then you must upgrade the target EJB application.

6.4 Impact of Upgrade on JMS Clients

It should first be noted that the information contained in this section is with regards to JMS clients that are not Message Driven Beans (MDB). MDBs are usually tightly coupled to the JMS provider's resources and as such should always be upgraded together with these resources. Specific WebLogic Server MDB capabilities and behavior should be considered during the upgrade of MDB applications.

The following sections describe two scenarios where the upgrade of an application to Oracle WebLogic Server might have an impact on JMS clients:

- Changes Required When the JMS Provider is Upgraded to WebLogic Server
- Changes Required When the JMS Provider Remains in OC4J

6.4.1 Changes Required When the JMS Provider is Upgraded to WebLogic Server

In this scenario, the JMS provider--and its related resources such as destinations and connection factories--is upgraded to Oracle WebLogic Server. You must modify both the upgraded client applications and the existing OC4J client applications still running within an OC4J server.

Specifically, you must modify the client applications to use the WebLogic Server JNDI provider as described in Section 6.2.1, "Modifying Clients to Use the Oracle WebLogic Server JNDI Provider".

When you modify the clients to use the Oracle WebLogic Server JNDI provider, the client applications obtain JMS resources from the Oracle WebLogic Server JMS provider and the following considerations should be taken into account because they could impact client applications:

- **Message Ordering:** Like the OC4J JMS provider, Oracle WebLogic Server provides the capability to guarantee strictly ordered message processing. Additionally, the WebLogic Server JMS "Unit of Order" feature allows for additional message order processing capabilities. For more information, see "Using Message Unit-of-Order" in *Oracle Fusion Middleware Programming JMS for Oracle WebLogic Server*.

- **Connection Pooling:** Unlike the OC4J JMS provider, the Oracle WebLogic Server JMS provider provides no pooling capability for stand-alone clients. If this feature is required, stand-alone clients should be modified to implement this capability through explicit re-use of JMS resources.
- **Network Connections:** Clients using the WebLogic Server JMS provider use a single network connection per client virtual machine, regardless of the number of JMS connection objects used. This behavior is slightly different from the OC4J JMS provider, which associates each JMS connection object with a separate network connection.

JMS clients that are not stand-alone applications and that continue running within an OC4J server instance while using JMS resources within a WebLogic Server environment should use the OC4J Oracle Enterprise Messaging Server JMS Connector feature.

6.4.2 Changes Required When the JMS Provider Remains in OC4J

In this scenario, the JMS provider remains within an OC4J infrastructure and client applications being upgraded to Oracle WebLogic Server require adjustments. These types of client applications should treat the OC4J JMS resources as remote JMS providers and use the Oracle WebLogic Server "foreign server" feature in order to provide access to the OC4J JMS resources to the WebLogic Server deployed JMS clients.

For more information, see "Configuring Foreign Server Resources to Access Third-Party JMS Providers" in *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

7

Upgrading a Java EE and Web Server Environment

This chapter provides information for users who installed and integrate Oracle HTTP Server with their Oracle Application Server 10*g* Release 3 (10.1.3.1.0) OC4J instances.

Specifically, if you are using Oracle HTTP Server or Oracle Web Cache as a front-end to your deployed applications, use the following sections to upgrade those components to the Oracle Fusion Middleware 11*g* Web Tier Suite:

- Task 1: Understand the Differences Between Using Oracle HTTP Server with OC4J and Oracle WebLogic Server
- Task 2: Install and Configure an Oracle Fusion Middleware Web Tier
- Task 3: Upgrade Your Oracle Application Server 10g Web Tier Components to Oracle Fusion Middleware 11g
- Task 4: Configure the Web Tier To Route Requests to Your Oracle Fusion Middleware Environment
- Task 5: Perform Any Required Post-Upgrade Tasks for the Web Tier Components
- Task 6: Verify the Web Tier Upgrade

7.1 Task 1: Understand the Differences Between Using Oracle HTTP Server with OC4J and Oracle WebLogic Server

Review the following information about the differences between using Web servers with OC4J and using Web servers with Oracle WebLogic Server:

- Configuring Web Sites and AJP Connections in Oracle WebLogic Server
- Installing and Configuring Oracle HTTP Server for Oracle WebLogic Server
- Using Web Servers Other than Oracle HTTP Server with Oracle WebLogic Server
- Understanding Oracle HTTP Server Interoperability Issues When Upgrading to Oracle Fusion Middleware 11g

7.1.1 Configuring Web Sites and AJP Connections in Oracle WebLogic Server

OC4J provided users with the ability to define multiple "Web sites" for each OC4J instance. In other words, you could define a unique listener, with its own port and protocol. Each Web application deployed to the OC4J instance could be bound to a specific OC4J web site, which directed any requests to that specific port and protocol to the desired application.

By default, every OC4J instance provided a default Web site that was preconfigured by the `default-web-site.xml` configuration file. You could then modify the default Web site configuration or define additional Web sites for specific applications.

Oracle WebLogic Server does not support the concept of a Web site. Instead, each Oracle WebLogic Server managed server is assigned a unique listening port. This listening port can be modified, but it always listens on this port for all supported protocols, such as HTTP, HTTPS, RMI, and so on. You can also configure a second, secure (SSL) port for each managed server.

However, unlike OC4J, Oracle WebLogic Server does not support the AJP or AJPS protocol, which is used in OC4J environments for communications between a front-end Web server and the OC4J instance. For more information, see Section 7.1.2, "Installing and Configuring Oracle HTTP Server for Oracle WebLogic Server".

To accommodate multiple listeners in Oracle WebLogic Server, you can do one of the following:

- Create multiple Oracle WebLogic Server managed servers and configure them to listen on unique ports. You can then deploy applications to each server and each application will listen on the listening port assigned to its host managed server.
- Configure multiple **network channels** per managed server. An Oracle WebLogic Server network channel is a configurable resource that defines the attributes of a network connection to a managed server. For each network channel, you can configure a set of attributes that are similar to those provided by OC4J Web sites.

  For more information, see "Configuring Network Resources" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*.

7.1.2 Installing and Configuring Oracle HTTP Server for Oracle WebLogic Server

This section includes the following information about using Oracle WebLogic Server with a Web server:

- How Oracle HTTP Server Is Configured for OC4J
- How Oracle HTTP Server is Configured for Oracle WebLogic Server

7.1.2.1 How Oracle HTTP Server Is Configured for OC4J

In previous versions of Oracle Application Server, it is common (and often recommended) to configure your environment with a front-end Web server. The Web server receives user requests and routes specific requests, based on the context root of the URL, to the applications deployed on the OC4J server.

Most OC4J users configure Oracle HTTP Server as the front-end to their Java EE server environment. Oracle HTTP Server is a component of the Oracle Fusion Middleware product set. The Advanced installation options in Oracle Application Server 10*g* Release 3 (10.1.3) automatically configure the Oracle HTTP Server to serve as a front-end to the OC4J server.

The connection between Oracle HTTP Server and OC4J is managed by the `mod_oc4j` module that is included with the Oracle HTTP Server software, and is transferred over the AJP protocol.

7.1.2.2 How Oracle HTTP Server is Configured for Oracle WebLogic Server

The same topology can be configured with Oracle WebLogic Server. However, in Oracle Fusion Middleware 11*g*, you install Oracle HTTP Server separately from Oracle WebLogic Server, as part of the Oracle Fusion Middleware Web Tier installation. The

Web Tier installation can also include Oracle Web Cache, which adds improved performance and caching capabilities to the Web tier.

After you install Oracle HTTP Server as part of a Web Tier installation, you can then configure the new `mod_wl_ohs` module, which allows requests to be proxied from Oracle HTTP Server to Oracle WebLogic Server. The `mod_wl_ohs` module provides similar capabilities for Oracle WebLogic Server as `mod_oc4J` did for OC4J.

For more information, see "mod_wl_ohs" in the *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

Note that Oracle HTTP Server is installed and configured automatically for certain Oracle Fusion Middleware 11*g* components that require a Web server. For example, Oracle HTTP Server is automatically installed and configured with the following Oracle Fusion Middleware components:

- Oracle Identity Federation in the Oracle Identity Management software suite
- Oracle Portal and Oracle Business Intelligence Discoverer in the Oracle Portal, Forms, Reports and Discoverer suite

However, for a Java EE environment, such as those described in this guide, you must install and configure Oracle HTTP Server separately from Oracle WebLogic Server.

7.1.3 Using Web Servers Other than Oracle HTTP Server with Oracle WebLogic Server

Oracle WebLogic Server supports other Web servers, as well as Oracle HTTP Server.

For more information, see *Oracle Fusion Middleware Using Web Server Plug-Ins with Oracle WebLogic Server*.

7.1.4 Understanding Oracle HTTP Server Interoperability Issues When Upgrading to Oracle Fusion Middleware 11*g*

Before you proceed with an Oracle HTTP Server upgrade, be sure to review the section, "About Oracle HTTP Server Interoperability During Upgrade" in the *Oracle Fusion Middleware Upgrade Planning Guide*.

7.2 Task 2: Install and Configure an Oracle Fusion Middleware Web Tier

The following sections describe the options available for installing, configuring, and upgrading a Web tier environment as a front end to your upgraded Oracle WebLogic Server environment:

- Deciding Upon a Location for Your Web Tier Components
- Associating the Web Tier Components with an Oracle WebLogic Server Domain
- Locating the Web Tier Installation and Configuration Documentation

7.2.1 Deciding Upon a Location for Your Web Tier Components

Before you install and configure Oracle HTTP Server as part of a Web Tier installation, consider where you want to install the Oracle HTTP Server. If you plan to use the Upgrade Assistant to upgrade your configuration settings from a previous version of Oracle HTTP Server, then you must install the Oracle HTTP Server on the same host as the Oracle Application Server 10*g* Oracle HTTP Server Oracle home.

If upgrading the configuration is not a requirement for your environment, then you can install and configure Oracle HTTP Server on a separate host and configure it later to send HTTP requests to your Oracle WebLogic Server domain.

For more information, see Task 4: Configure the Web Tier To Route Requests to Your Oracle Fusion Middleware Environment.

7.2.2 Associating the Web Tier Components with an Oracle WebLogic Server Domain

When you install Oracle HTTP Server as part of a Web Tier installation, you can choose whether or not to associate the Web Tier components with an Oracle WebLogic Server domain. Consider the following possible topologies:

- Configure the Web tier components as part of an Oracle WebLogic Server domain.

  With this option, you can add Oracle HTTP Server and (optionally) Oracle Web Cache to an existing domain.

  For example, you can add the Web tier components to the Oracle SOA Suite, WebCenter, or Application Developer domain you are using to deploy your Java EE applications.

  This can be an advantage because you can then easily configure the Web tier components as a front end to your deployed applications, and you can then use Oracle Enterprise Manager Fusion Middleware Control to manage the domain, as well as the applications you deploy on the domain.

  Note: Oracle does not support associating your Web tier components with a development domain created with Oracle JDeveloper. For more information, see Section 5.1, "Task 1: Install and Configure an Oracle WebLogic Server Development Domain".

- Configure the Web tier components without a domain.

  With this option, the Oracle HTTP Server and (optionally) the Oracle Web Cache instances are installed "standalone" in a separate Oracle home and are not associated with a domain.

  Consider this topology if you are installing the Web tier components on a separate host, you are not planning to upgrade your previous Oracle Application Server 10*g* configuration settings, and you are not planning to manage your Web tier components with Oracle Enterprise Manager Fusion Middleware Control.

7.2.3 Locating the Web Tier Installation and Configuration Documentation

When you are ready to install and configure Oracle HTTP Server and (optionally) Oracle Web Cache, refer to the *Oracle Fusion Middleware Installation Guide for Web Tier* for complete instructions.

If you are not planning to associate the Web tier components with an Oracle WebLogic Server domain, you can optionally use the *Oracle Fusion Middleware Quick Installation Guide for Web Tier*.

7.3 Task 3: Upgrade Your Oracle Application Server 10*g* Web Tier Components to Oracle Fusion Middleware 11*g*

If you used Oracle HTTP Server previously, you can use the Oracle Fusion Middleware Upgrade Assistant to upgrade specific Oracle HTTP Server configuration settings from Oracle Application Server 10*g* Release 3 (10.1.3) to your new Oracle Fusion Middleware 11*g* Oracle HTTP Server instance.

Alternatively, if you are using a Web server other than Oracle HTTP Server, or if you have installed Oracle HTTP Server on a separate host from the host where your Oracle Application Server 10*g* Release 3 (10.1.3) environment resides, then you must manually reconfigure your new Oracle Fusion Middleware environment.

Refer to the following sections for information on using the Oracle Fusion Middleware Upgrade Assistant to upgrade your Oracle HTTP Server configuration to 11*g*:

- Task 3a: Start the Upgrade Assistant for an Web Tier Upgrade
- Task 3b: Use the Upgrade Assistant to Upgrade the Web Tier Components

7.3.1 Task 3a: Start the Upgrade Assistant for an Web Tier Upgrade

To start the Upgrade Assistant using the graphical user interface:

> **Note:** You can also use the Upgrade Assistant command-line interface to upgrade your Oracle Application Server 10*g* Oracle homes. For more information, see "Using the Upgrade Assistant Command-Line Interface" in the *Oracle Fusion Middleware Upgrade Planning Guide*.

1. Change directory to the *ORACLE_HOME*`/bin` directory of the Oracle Fusion Middleware installation.
2. Enter the following command to start the Upgrade Assistant.

   On UNIX system:

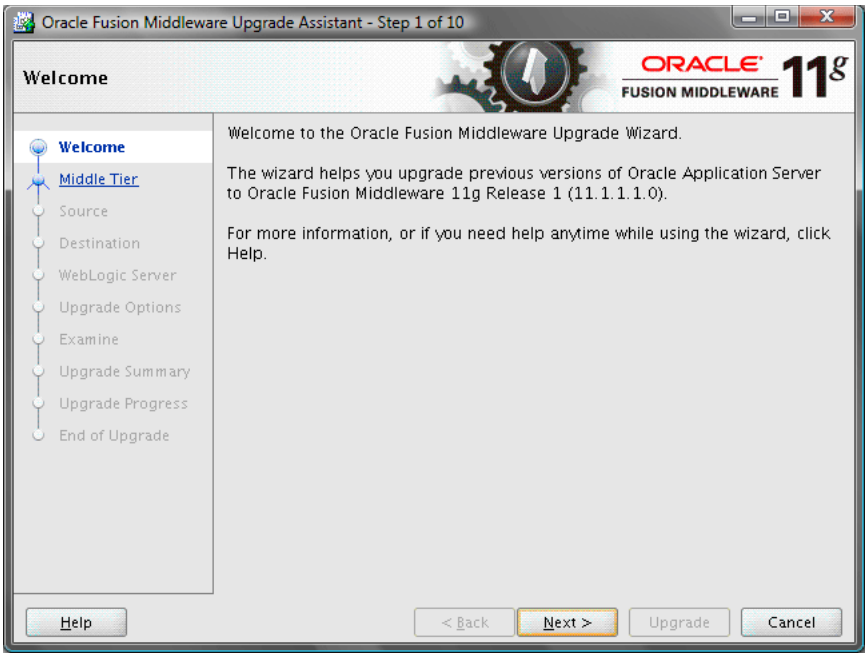
   ```
   ./ua
   ```

   On Windows systems:

   ```
   ua.bat
   ```

   The Upgrade Assistant displays the Welcome screen as shown in Figure 7–1

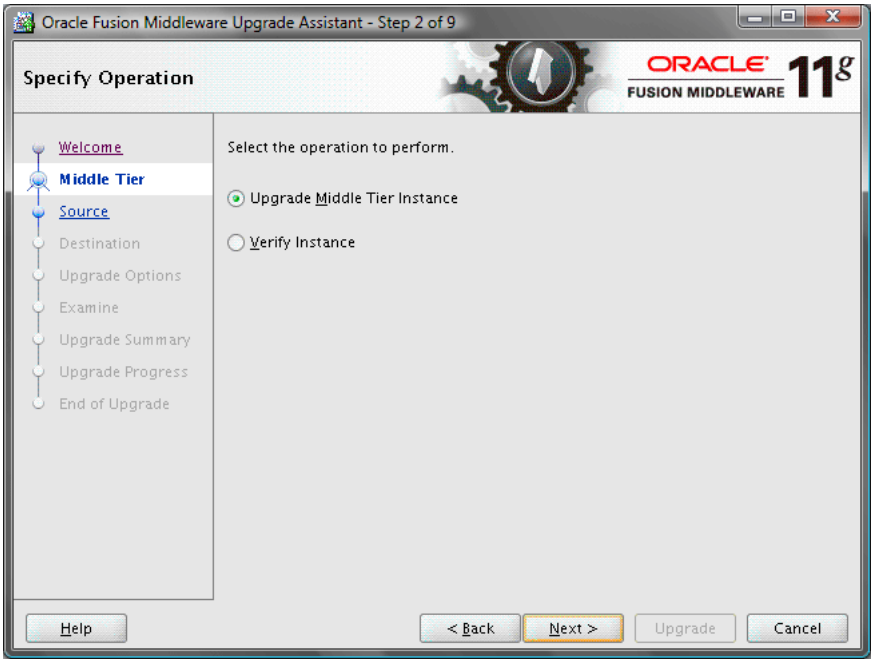
Figure 7–1 Upgrade Assistant Welcome Screen



3. Click **Next** to display the Specify Operation screen (Figure 7–2).

   The options available in the Upgrade Assistant are specific to the Oracle home from which it started. For example, when you start Upgrade Assistant from an Web Tier Oracle home, the options shown on the Specify Operation screen are the valid options for the components in a typical Web Tier Oracle home.

Figure 7–2 Specify Operation Screen When Upgrading a Web Tier Installation



7.3.2 Task 3b: Use the Upgrade Assistant to Upgrade the Web Tier Components

The following sections provide information about upgrading your Oracle HTTP Server to Oracle Fusion Middleware 11*g*:

- Upgrading the Web Tier Components
- Important Notes When Using the Source Oracle Home Ports in the Destination Oracle Instance

7.3.2.1 Upgrading the Web Tier Components

To upgrade your Web tier components to Oracle Fusion Middleware 11*g*:

1. Start the Upgrade Assistant as described in Task 3a: Start the Upgrade Assistant for an Web Tier Upgrade.
2. Select **Middle Tier Instance** on the Specify Operation screen (Figure 7–2).
3. Refer to Table 7–1 for a description of the Upgrade Assistant screens that require input from you during a middle-tier instance upgrade and the options on each screen.

   The Upgrade Assistant performs the following tasks and provides the progress on each task:

   - Examines the components and schemas to be upgraded and verifies that they can be upgraded successfully.
   - Provides a summary of the components to be upgraded so you can verify that the Upgrade Assistant is upgrading the components and schemas you expect.
   - Provides a progress screen so you can see the status of the upgrade as it proceeds.
   - Alerts you of any errors or problems that occur during the upgrade.

     See Also: Section B.1, "Troubleshooting Upgrade Assistant Problems and Issues" in the *Oracle Fusion Middleware Upgrade Planning Guide* for specific instructions for troubleshooting problems that occur while running the Upgrade Assistant

   - Displays the End of Upgrade screen, which confirms that the upgrade was complete.

Table 7–1 Upgrade Assistant Screens That Require Input During a Middle-Tier Instance Upgrade

Upgrade Assistant Screen	Description
Specify Source Home	Select the 10*g* Release 2 (10.1.2) or 10*g* (10.1.4) Identity Management instance source Oracle home. If the Oracle home you want to upgrade does not appear in the drop-down lists, see Section B.1.2.1, "Source Oracle Home Not Listed by OracleAS Upgrade Assistant" in the *Oracle Fusion Middleware Upgrade Planning Guide*.
Specify Destination Instance	Enter the complete path to the 11*g* Oracle instance, or click **Browse** to locate the instance directory.
Specify WebLogic Server	Enter the host and Administration Server port for the Oracle WebLogic Server you configured in Task 2: Install and Configure an Oracle Fusion Middleware Web Tier.

Table 7–1 (Cont.) Upgrade Assistant Screens That Require Input During a Middle-Tier Instance Upgrade

Upgrade Assistant Screen	Description
Specify Upgrade Options	Select the upgrade options you want to apply to the Oracle Portal, Forms, Reports, and Discoverer upgrade: ■ **Use source Oracle home ports in destination:** If you want to migrate the port assignments used by your Oracle Application Server 10g Oracle home to your new Oracle Fusion Middleware Oracle instance. For more information, see Section 7.3.2.2, "Important Notes When Using the Source Oracle Home Ports in the Destination Oracle Instance". If you do not select this option, and you are upgrading Oracle Web Cache, see Section 7.5.2, "Verifying and Updating the Oracle HTTP Server and Oracle Web Cache Ports After Upgrade" ■ **Stop source components before upgrade:** By default, this check box is selected and all the components in the Source Oracle home will be stopped before the upgrade process begins. Stopping the source components is necessary to avoid any port conflicts when you select the Use source Oracle home ports in destination option. ■ **Start destination components after successful upgrade:** if you want the Upgrade Assistant to automatically start the components in the destination Oracle home after the upgrade is complete. If you do not select this option, then you will have to manually start the destination instance after the upgrade. Click **Help** to display more information about the upgrade options on this screen.

7.3.2.2 Important Notes When Using the Source Oracle Home Ports in the Destination Oracle Instance

When you select the source Oracle home ports in destination option in the Oracle Fusion Middleware Upgrade Assistant, note the following:

- If you select this option, then you will not be able to run both the 10*g* and 11*g* middle tiers at the same time; otherwise, port conflicts will occur.
- If you are upgrading to multiple instances of a particular Oracle Fusion Middleware 11*g* component, note that you can select this option only once for each component that you upgrade on a host; otherwise port conflicts will result.

  For example, suppose you upgrade and Oracle HTTP Server in one Oracle instance on MYHOST1.

  If you use the option again while upgrading another Oracle HTTP Server instance in another Oracle instance on MYHOST1, then the same listening ports are assigned to the second Oracle HTTP Server instance. Two instances of Oracle HTTP Server on the same host cannot use the same listening ports.
- If you install and configure both Oracle Web Cache and Oracle HTTP Server 11*g* as part of a Web Tier and Utilities installation, and you use this option to upgrade a 10g Oracle home where only Oracle HTTP Server is installed, then you must modify the Oracle Web Cache instance after the upgrade.

  Specifically, since you are now using the 10*g* ports, you must modify the Oracle Web Cache instance so it sends requests to the port that was used in 10*g*, rather

than the Oracle HTTP Server listening port assigned during the WebTier and Utilities installation and configuration.

For more information, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

7.4 Task 4: Configure the Web Tier To Route Requests to Your Oracle Fusion Middleware Environment

To configure Oracle HTTP Server to route requests to Oracle WebLogic Server, use the instructions in the *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

In particular refer to these sections in the *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*:

- "Understanding Oracle HTTP Server Modules"
- "Configuring the mod_wl_ohs Module"

7.5 Task 5: Perform Any Required Post-Upgrade Tasks for the Web Tier Components

The following sections describe some configuration tasks that you might have to perform after upgrading to your Web tier components to 11*g*:

- Verifying the Location of the Oracle HTTP Server and Oracle Web Cache Wallets After Upgrade
- Verifying and Updating the Oracle HTTP Server and Oracle Web Cache Ports After Upgrade

---

Note: If you configured Oracle HTTP Server and Oracle Web Cache with Oracle Application Server Single Sign-On, then additional post-upgrade tasks are necessary.

For more information, see "Web Tier Component Post-Upgrade Tasks When Using Oracle Single Sign-On" in the *Oracle Fusion Middleware Upgrade Guide for Oracle Portal, Forms, Reports, and Discoverer*.

7.5.1 Verifying the Location of the Oracle HTTP Server and Oracle Web Cache Wallets After Upgrade

If you configured a secure socket layer (SSL) wallet for Oracle Web Cache 10*g*, then consider the following information about the Oracle Web Cache upgrade process:

- If you stored the wallet in a directory inside the Oracle Web Cache 10*g* Oracle home, then during upgrade, the wallet files in the directory are moved to the following new directory in the Oracle Fusion Middleware 11*g* Instance home:

  `ORACLE_INSTANCE/config/WebCache/component_name/keystores/wallet_dir_name10g`

  For example:

  `ORACLE_INSTANCE/config/WebCache/webcache1/keystores/wc_wallets10g`

- If you stored the wallet in a directory outside of the Oracle Web Cache 10*g* Oracle home, then you can continue to use the original pre-upgrade location after you upgrade to Oracle Web Cache 11*g*.

If you configured an SSL wallet for Oracle HTTP Server 10*g*, then the Upgrade Assistant upgrades the Oracle HTTP Server 10*g* wallets to the new 11*g* format. The Oracle HTTP Server 11*g* wallets are saved in the following location in the 11*g* Oracle instance directory:

ORACLE_INSTANCE/config/OHS/*component_name*/keystores/default/

If you have defined specific wallets for each user, then the wallets are saved in subdirectory of the keystores directory. The name of the directory is based on the name of the directory where you stored the 10*g* wallets.

For example, if you stored your 10*g* wallets in a directory called `/home/jones/security/mywallets/`, then after the upgrade, you can find the upgraded wallets in the following directory:

ORACLE_INSTANCE/config/OHS/*component_name*/keystores/security_mywallets/

7.5.2 Verifying and Updating the Oracle HTTP Server and Oracle Web Cache Ports After Upgrade

If you did not select **Use source Oracle home ports in destination** on the Specify Upgrade Options screen of the Upgrade Assistant, then after the upgrade of the Web tier components, you should verify the ports used by the upgraded Oracle HTTP Server and Oracle Web Cache 11*g* instances.

Specifically, you should verify the listening ports, origin servers, site definitions, and site-to-server mapping settings, and make changes if appropriate.

If the Oracle HTTP Server and Oracle Web Cache components reside in the same instance and you upgrade them together, no modifications should be necessary.

However, in the following circumstances, you must re-configure the ports and connections between your Oracle HTTP Server and Oracle Web Cache instances:

- If you have upgraded the Oracle HTTP Server and Oracle Web Cache instances separately--for example, if you upgrade Oracle HTTP Server on one host and later upgrade an associated Oracle Web Cache instance on another host.
- If you are using an Oracle Web Cache cluster, or you typically configure multiple Oracle Web Cache instances routing to multiple Oracle HTTP Server instances.

For more information about configuring the connections between Oracle HTTP Server and Oracle Web Cache, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

7.6 Task 6: Verify the Web Tier Upgrade

To verify that your Web Tier upgrade was successful:

1. Run the Upgrade Assistant again and select **Verify Instance** on the Specify Operation page.

   Follow the instructions on the screen for information on how to verify that specific Oracle Fusion Middleware components are up and running.

2. If you associated the Web Tier with an Oracle WebLogic Server domain and the Oracle Enterprise Manager template was applied to the domain, then use the

Fusion Middleware Control to verify that the Web Tier components are up and running.

For more information, see "Getting Started Using Oracle Enterprise Manager Fusion Middleware Control" in the *Oracle Fusion Middleware Administrator's Guide*.

A

orion-web.xml and orion-ejb-jar.xml Upgrade Reference

This appendix contains reference information you can use when preparing your Oracle Application Server 10*g* applications for redeployment on Oracle WebLogic Server.

Refer to the following sections for more information about upgrading elements of the following OC4J deployment descriptors:

- orion-web.xml
- orion-ejb-jar.xml

orion-web.xml

The OC4J-specific application-level Web descriptor, `orion-web.xml`, is distributed in the `/WEB-INF` directory of your WAR files. It is used to add OC4J-specific settings, or override any settings in `web.xml`.

In Oracle WebLogic Server, the equivalent vendor specific deployment descriptor is called `weblogic.xml`, and it resides within the `/WEB-INF` directory of the web module.

When redeploying your 10*g* applications on Oracle WebLogic Server, you must convert any specific OC4J settings you have set in your Web module to the WebLogic Server equivalents in the `weblogic.xml` file. For more information, refer to the information provided for each element in the `orion-web.xml` file in this appendix.

<classpath>

OC4J Definition

Allows OC4J web applications to refer to code sources (for example, JAR or ZIP files), which are located inside and outside the Web application scope. Any valid code sources contained in these locations are added to the Web application's class-loader at runtime.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

Two options:

- Copy the referenced JAR files to the Web application's `WEB-INF/lib` directory.

  OR

- Package the classes in a JAR file and deploy the JAR file as an Oracle WebLogic Server shared library, which is referenced by the application through a `<library-ref>` element in the application's `weblogic-application.xml` deployment descriptor.

More Information

"Creating Shared Java EE Libraries and Optional Packages" in the *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*

<contextParamMappingFinding>

OC4J Definition

Overrides the value specified through a corresponding `<context-param>` element in `web.xml` for a servlet context parameter.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

There is no direct equivalent in Oracle WebLogic Server. However, It is possible to override context parameters via the following from any servlet or filter:

```
getServletContext().getInitParameter("ContextParam")
```

More Information
"ServletConfig" Java servlet interface in the *Java 2 Platform, Enterprise Edition, v 1.3 API Specification* on the java.sun.com Web site

<mimeMappings>

OC4J Definition
Defines the path to a file containing MIME mappings.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
There is no direct equivalent in Oracle WebLogic Server. However, it is possible to override the value specified in the `<mimeMappings>` element through a corresponding `<mime-mapping>` element in `web.xml`.

More Information
"web.xml Deployment Descriptor Elements" in *Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.

<virtual-directory>

OC4J Definition
Adds a virtual directory mapping for static content, working in a way that is conceptually similar to symbolic links on a UNIX system, for example.

The virtual directory enables you to make the contents of the real document root directory available to the application without physically residing in the Web application WAR file. This is useful, for example, when linking to an enterprise-wide error page from multiple WAR files.

Equivalent Entry in weblogic.xml
<virtual-directory-mapping>

Upgrade Advice
Create an equivalent `<virtual-directory-mapping>` entry in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

<access-mask>

OC4J Definition
Specifies optional access masks for the application. You can use host names or host domains to filter clients, use IP addresses and subnets to filter clients, or you can use both.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
Create the same filter at either your network firewall or application load-balancer, if available. Alternatively, consider using an Oracle WebLogic Server network connection filter to provide a filter for the Oracle WebLogic Server domain.

However, it is important to note that Oracle WebLogic Server network connection filters can act only on an entire domain and cannot be set on a per application basis.

More Information

"Using Network Connection Filters" in the *Oracle Fusion Middleware Programming Security for Oracle WebLogic Server*

<servlet-chaining>

OC4J Definition

Specifies a servlet to call when the response of the current servlet is set to a specified MIME type.

The specified servlet is called after the current servlet. Use this for filtering or transforming certain kinds of output.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

Create a standard filter to achieve the same functionality. OC4J servlet chaining is an older and proprietary mechanism with functionality similar to that of standard servlet filtering, which was introduced in version 2.3 of the Servlet specification.

More Information

"The Essentials of Filters" on the `java.sun.com` Web site.

<request-tracker>

OC4J Definition

Specifies a servlet to use as a request tracker, which is invoked for each separate request sent from a browser to the server, at the same time as the corresponding response is committed (immediately before the response is actually sent).

Request trackers are useful for logging information.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

Create a standard Servlet Request Listener that contains the functionality of the request tracker servlet.

OC4J servlet request tracker is an older and proprietary mechanism with functionality similar to that of a standard request listener, which was introduced in version 2.4 of the Servlet specification.

More Information

"Servlet Life Cycle" in the J2EE 1.4 Tutorial on the `java.sun.com` Web site.

<session-tracking>

OC4J Definition

Specifies the session-tracking settings for this application.

Session tracking is accomplished through cookies, assuming a cookie-enabled browser.

Equivalent Entry in weblogic.xml
<session-descriptor>

Upgrade Advice
Use the `<session-descriptor>` element in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

<session-tracker>

OC4J Definition
This subelement of `<session-tracking>` specifies a servlet to use as a session tracker.

A session tracker is invoked as soon as a session is created; specifically, at the same time as the invocation of the `sessionCreated()` method of the HTTP session listener (an instance of a class implementing the `javax.servlet.http.HttpSessionListener` interface).

Session trackers are useful for logging information, for example.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
Create a standard HttpSessionListener that contains the functionality of the session tracker servlet.

More Information
"Servlet Life Cycle" in the J2EE 1.4 Tutorial on the `java.sun.com` Web site.

<resource-ref-mapping>

OC4J Definition
Declares a JNDI location for an external resource, such as a data source, JMS queue, or mail session. This is in conjunction with a corresponding `<resource-ref>` element in the `web.xml` file, which declares the resource.

Equivalent Entry in weblogic.xml
<resource-description>

Upgrade Advice
Use the `<resource-description>` element in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

<lookup-context>

OC4J Definition
This element, through its location attribute, specifies an optional JNDI context that will be used instead of the default context in looking up the resource mapped in the parent `<resource-ref-mapping>` element.

This is useful when you are connecting to third-party modules, such as a third-party JMS server, for example.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
In Oracle WebLogic Server, the default module context is used to look up the JNDI name. However, if the third-party JNDI module is a JMS server and the intent is to use the associated connection factories and destinations, WebLogic JMS enables you to reference third-party JMS providers within a local WebLogic Server JNDI tree.

More Information
"Configuring Foreign Server Resources to Access Third-Party JMS Providers" in *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*

<resource-env-ref-mapping>

OC4J Definition
Declares a JNDI location for an environment resource. This is in conjunction with a corresponding `<resource-env-ref>` element in the `web.xml` file, which declares the resource.

Equivalent Entry in weblogic.xml
<resource-env-description>

Upgrade Advice
Use the `<resource-env-description>` element in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

<env-entry-mapping>

OC4J Definition
Overrides the value specified through a corresponding `<env-entry>` element in web.xml, for an environment entry.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
There is no direct equivalent in Oracle WebLogic Server. However, it is possible to override the value specified in a corresponding `<env-entry>` element in `web.xml`.

More Information
"web.xml Deployment Descriptor Elements" in *Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.

<ejb-ref-mapping>

OC4J Definition
Declares a JNDI location for an EJB. This is in conjunction with a corresponding `<ejb-ref>` or `<ejb-local-ref>` element to declare the EJB in the `web.xml` file.

Equivalent Entry in weblogic.xml
<ejb-reference-description>

Upgrade Advice
Use the `<ejb-reference-description>` element in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

<service-ref-mapping>

OC4J Definition
Use this element in conjunction with a `<service-ref>` element that appears in the `web.xml` file to declare a Web service.

Equivalent Entry in weblogic.xml
<service-ref-description>

Upgrade Advice
Use the `<service-ref-description>` element in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

<expiration-setting>

OC4J Definition
Sets the expiration for a given set of resources; that is, how long before the resources will expire in the browser. (The browser reloads an expired resource upon the next request for it.)

This is useful for caching policies, such as not reloading images as frequently as documents.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
This is performance feature OC4J Servlet container in that this can reduce the requests to the server by asking the browser to cache certain requests.

There is no equivalent feature in Oracle WebLogic Server.

More Information
None.

<jazn-web-app>

OC4J Definition
Configures the OracleAS JAAS Provider and Single Sign-On (SSO) properties for servlet execution. You must set these features appropriately to invoke a servlet under the privileges of a particular security subject.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice

In Oracle WebLogic Server, security subject propagation is automatically supported within the same domain.

However, if the identity is propagated between multiple domains, cross-domain security or global trust must be enabled.

More Information

Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server

"Enable Cross Domain Security between domains" in the Oracle WebLogic Server Administration Console online help

<security-role-mapping>

OC4J Definition

This element maps a security role to specified users and groups, or to all users. It maps to a security role of the same name specified through a `<security-role>` element in the `web.xml` file. Use either the implies All attribute or an appropriate combination of subelements—`<group>`, `<user>`, or both.

Equivalent Entry in weblogic.xml

<security-role-assignment>

Upgrade Advice

Use the `<security-role-assignment>` element in `weblogic.xml`. The security role name should be created in the destination Oracle WebLogic Server domain.

More Information

"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

<web-app-class-loader>

This section describes the attributes supported by the OC4J `<web-app-class-loader>` element in the orion-web.xml deployment descriptor.

search-local-classes-first

OC4J Definition

Set this to "true" to search and load WAR file classes before system classes. By default, system classes are searched and loaded first.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

Configure a filtering classloader by adding a `<prefer-application-package>` element in the application's `weblogic-application.xml`, which lists the packages be loaded explicitly from the application and ensure that the "required" version of the JAR file is available to the application, either as a application shared library or by bundling the JAR file in the `APP-INF/lib` or `WEB-INF/lib` directories of the application.

Unlike OC4J, if a WebLogic Server filtered class cannot be resolved through the application class-loader, Oracle WebLogic Server will not attempt to load it from parent class-loaders. It is therefore important to ensure that the package names listed

within the `<prefer-application-packages>` element of `weblogic-application.xml` are correct.

More Information

"Using a Filtering Classloader" in *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*

"Creating Shared Java EE Libraries and Optional Packages" in *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*

include-war-manifest-class-path

OC4J Definition

Set this attribute to "false" to not ignore the classpath specified in the WAR file manifest `Class-Path` attribute when searching and loading classes from the WAR file (regardless of the `search-local-classes-first` setting). Otherwise, the classpath from the WAR file manifest is included.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

WebLogic Server does not read the classpath specified in the WAR file manifest. However, it is possible to bundle the classes as part of application `WEB-INF/lib` directory.

More Information

"Creating Shared Java EE Libraries and Optional Packages" in *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*

autojoin-session

OC4J Definition

Specifies whether users should be assigned a session as soon as they log in to the application.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

In WebLogic Server, users are automatically assigned a session as soon as they log in to the application. Unlike OC4J, this behavior cannot be disabled with a deployment descriptor element.

More Information

None.

default-buffer-size

OC4J Definition

Specifies the default size of the output buffer for servlet responses, in bytes.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice
There is no direct equivalent of this configuration element in Oracle WebLogic Server. However, it is possible to call `setBufferSize()` using the response object from any servlet. A JSP page can also include a buffer size directive, `<%@ page buffer%>`.

More Information
None.

default-charset

OC4J Definition
In OC4J 10*g* Release 3 (10.1.3.1.0), for JSP pages and for the servlet container, this attribute specifies the ISO character set to use by default.

In general, for JSP 2.0 users, Oracle instead recommends standard `<page-encoding>` functionality (under the `web.xml <jsp-config>` element, according to the JSP 2.0 specification), to specify character sets according to URL patterns.

However, `default-charset` may be useful if you have large numbers of JSP pages, particularly across multiple applications, to avoid the necessity of making numerous changes in your EAR files.

Also, you can use `default-charset` to set a base default, then use `<page-encoding>` functionality to override the default for particular URL patterns.

Equivalent Entry in weblogic.xml
jsp-descriptor/encoding

Upgrade Advice
Use the `jsp-descriptor/encoding` configuration in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

default-mime-type

OC4J Definition
Specifies a default content type for servlet responses, for situations where the `setContentType()` method is not called from the servlet implementation.

If `default-mime-type` is not specified, then there is no default content type.

Equivalent Entry in weblogic.xml
container-descriptor/default-mime-type

Upgrade Advice
Use the `container-descriptor/default-mime-type` configuration in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

development

OC4J Definition
Use this OC4J-specific flag during development; it prompts the OC4J server to check a particular directory for updates to servlet source files.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
Oracle WebLogic Server does not auto-compile java sources. However:

- To enable dynamic class loading for JSP files, configure `jsp-descriptor/page-check-seconds` in `weblogic.xml`.
- To enable dynamic class loading for servlets, configure `container-descriptor/servlet-reload-check-secs` in `weblogic.xml`.

In development mode, both these flags are set with the default value of one (1) second. In a production environment, these flags are disabled and need to be set explicitly.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

directory-browsing

OC4J Definition
This attribute specifies whether to allow directory browsing for a URL that ends in "/".

Equivalent Entry in weblogic.xml
container-descriptor/index-directory-enabled

Upgrade Advice
Use the `container-descriptor/index-directory-enabled` configuration in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

enable-jsp-dispatcher-shortcut

OC4J Definition
This OC4J-specific flag for performance tuning in conjunction with a "true" setting for the `simple-jsp-mappingattribute`.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
No equivalent in Oracle WebLogic Server.

More Information
None.

file-modification-check-interval

OC4J Definition
Determines when to check a static file, such as an HTML file, to see whether its timestamp has changed and it should therefore be reloaded from the file system.

Equivalent Entry in weblogic.xml
container-descriptor/index-directory-enabled

Upgrade Advice
Use the `container-descriptor/resource-reload-check-sec` element in `weblogic.xml`.

Also consider these related `weblogic.xml` settings:

- To use similar setting for JSP, use the `jsp-descriptor/page-check-seconds`.
- To use similar setting for servlets, use `container-descriptor/servlet-reload-check-secs`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

jsp-cache-directory

OC4J Definition
Specifies the JSP cache directory, which is used as a base directory for output files from the JSP translator.

Equivalent Entry in weblogic.xml
jsp-descriptor/workingDir

Upgrade Advice
Use the `workingDir` attribute of the `<jsp-descriptor>` element in `weblogic.xml`.

However, note that unlike the `jsp-cache-directory` configuration in OC4J, this directory is not relevant for TLDs.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

jsp-cache-tlds

OC4J Definition
Indicates whether persistent TLD caching is enabled for JSP pages.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
There is no need to upgrade this configuration, because Oracle WebLogic Server supports TLD caching, and TLD caching is ON by default.

Unlike OC4J, TLD caching cannot be disabled with a deployment descriptor element.

More Information
None.

jsp-print-null

OC4J Definition
Set this flag to "false" to print an empty string instead of the default "null" string for null output from a JSP page.

Equivalent Entry in weblogic.xml
jsp-descriptor/print-nulls

Upgrade Advice
Use the `print-nulls` attribute of the `<jsp-descriptor>` element in `weblogic.xml`.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

jsp-taglib-locations

OC4J Definition
Use this attribute to provide a semicolon-delimited list of one or more directories to use as "well-known" locations if persistent TLD caching is enabled for JSP pages (through the `jsp-cache-tlds` attribute).

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
There is no need to upgrade this configuration, because Oracle WebLogic Server supports TLD caching, and TLD caching is ON by default.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server

jsp-timeout

OC4J Definition
Specifies a period of time after which any JSP page will be removed from memory if it has not been requested.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
This feature affects applications in which efficient memory utilization is a key factor. It frees up resources in situations where some JSP pages are called infrequently.

There is no need to upgrade this feature.

More Information
None.

persistence-path

OC4J Definition
Indicates where to store `servletHttpSessionobjects` for persistence across server restarts or application redeployments.

Session objects must be serializable (directly or indirectly implementing the `java.io.Serializable` interface) or remoteable (directly or indirectly implementing the `java.rmi.Remote` interface) for this feature to work.

Equivalent Entry in weblogic.xml
- `persistent-store-type` attribute of the `session-descriptor` element
- `save-sesssions-enables` attribute of the `container-descriptor` element

Upgrade Advice
Set the `persistent-store-type` attribute of the `session-descriptor` element to "memory," and set the `save-sessions-enabled` attribute of the `container-descriptor` element to "true" in `weblogic.xml`.

This will ensure that sessions are serialized to disk across application redeployments or server restarts.

More Information
"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server

schema-major-version

OC4J Definition
The major version number of the `orion-web.xml` XSD.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
Not required in Oracle WebLogic Server.

More Information
None.

schema-minor-version

OC4J Definition
The minor version number of the `orion-web.xml` XSD.

Equivalent Entry in weblogic.xml
None.

Upgrade Advice
None.

More Information
None.

servlet-webdir

OC4J Definition

Use this attribute, in conjunction with a "true" setting for the OC4J system property `http.webdir.enable`, to enable servlet invocation by class name in standalone OC4J.

After the system property is set, any `servlet-webdir` setting that starts with a slash ("/") enables this feature and specifies a special URL portion to insert after the context path to instruct OC4J to invoke a servlet by class name. Anything appearing after this path in a URL is assumed to be a class name, including the package.

This feature is typically used in an OC4J standalone environment during development and testing; it presents a significant security risk and should not be used in a production environment.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

Configure standard servlet and servlet-mapping declarations from the servlet specification to configure `weblogic.servlet.ServletServlet`.

More Information

"Creating and Configuring Servlets" in *Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*

simple-jsp-mapping

OC4J Definition

Set this to "true" if "*.jsp" is mapped to only the `oracle.jsp.runtimev2.JspServlet` front-end JSP servlet.

This would be specified in the `<servlet>` elements of any Web descriptors affecting your application (`global-web-application.xml`, `web.xml`, and `orion-web.xml`).

Enabling this attribute improves performance for JSP pages.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

This is performance tuning flag for OC4J JSP container and is not required in WebLogic Server.

More Information

None.

source-directory

OC4J Definition

For situations in which the development attribute is set to "true", the `source-directory` setting specifies where to look for servlet source files to auto-compile.

If you use the default location, OC4J keeps track of the location of the `/WEB-INF` directory of your application after deployment. Note that modified source files will be

found anywhere under the `source-directory` directory, according to package name.

Equivalent Entry in weblogic.xml

- `page-check-seconds` attribute of the `jsp-descriptor` element
- `servlet-reload-check-secs` attribute of the `container-descriptor` element

Upgrade Advice

WebLogic Server does not auto-compile java sources.

- To enable dynamic class loading for JSP files, configure `page-check-seconds` attribute of the `jsp-descriptor` element.
- To enable dynamic class loading for servlets, configure the `servlet-reload-check-secs` attribute of the `container-descriptor` element.

In development mode, both these flags are set with the default value 1 second. In Production environment, these flags are disabled and need to be set explicitly."

More Information

"weblogic.xml Deployment Descriptor Elements" in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

temporary-directory

OC4J Definition

This is the path to a temporary directory that can be used by servlets and JSP pages for scratch files. The path can be either absolute, or relative to the deployment directory.

Equivalent Entry in weblogic.xml

None.

Upgrade Advice

Configure the `javax.servlet.context.tempdir` attribute from Servlet Specification.

Note that applications may not work properly, if at all, when using a temporary work directory that Servlets and other classes can use to store information.

More Information

"ServletContext" Java servlet interface in the *Java 2 Platform, Enterprise Edition, v 1.3 API Specification* on the java.sun.com Web site

orion-ejb-jar.xml

The OC4J-specific application-level EJB descriptor, `orion-ejb.xml`, is distributed in the `/WEB-INF` directory of your WAR files. It is used to add OC4J-specific settings, or override any settings in `ejb-jar.xml`.

In Oracle WebLogic Server, the equivalent vendor specific deployment descriptor is called `weblogic-ejb-jar.xml`, and it resides in the `/WEB-INF` directory of the web module.

When redeploying your 10*g* applications on Oracle WebLogic Server, you must convert any specific OC4J settings you have set in your EJB module to the WebLogic Server equivalents in the `weblogic-ejb-jar.xml` file. For more information, refer to the information provided for the key elements and element attributes and in the `orion-ejb.xml` file in this appendix.

<session-deployment>

OC4J Definition

Provides additional, customized deployment information for a session bean deployed within this JAR file. The existence of this element indicates that the application contains customized deployment settings for a session bean.

Equivalent Entry in weblogic-ejb-jar.xml

- `stateless-session-descriptor`
- `stateful-session-descriptor`

Upgrade Advice

Session bean customization in Oracle WebLogic Server is accomplished via the `stateless-session-descriptor` and `stateful-session-descriptor` elements.

More Information

"stateless-session-descriptor" and "stateful-session-descriptor" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

copy-by-value

OC4J Definition

This attribute of the `<session-deployment>` element indicates whether or not to copy (clone) all the incoming and outgoing parameters in EJB calls. Set to `false` if you are certain that your application does not assume `copy-by-value` semantics for a speed-up. The default value is true.

Equivalent Entry in weblogic-ejb-jar.xml

enable-call-by-reference

Upgrade Advice

Specify the copy by value semantics in `weblogic-ejb-jar.xml` using the `enable-call-by-reference` element.

More Information

"enable-call-by-reference" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

idletime

OC4J Definition
Use this attribute of the `<session-deployment>` element to set an idle timeout for each bean. When this timeout expires, passivation occurs. Set this attribute to the appropriate number of seconds. Default: 300 seconds. (5 minutes). To disable, specify any negative number.

Equivalent Entry in weblogic-ejb-jar.xml
stateful-session-cache/idle-timeout-seconds

Upgrade Advice
Use the `stateful-session-cache/idle-timeout-seconds` element of `stateful-session-descriptor` to set the idle timeout in `weblogic-ejb-jar.xml`.

More Information
"stateful-session-cache" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

min-instances

OC4J Definition
This attribute of the `<session-deployment>` element is the number of minimum bean implementation instances to be kept instantiated or pooled. The default is zero. This setting is valid for stateless session beans only.

The presence of this attribute indicates that the application contains customized bean pooling settings.

Equivalent Entry in weblogic-ejb-jar.xml
initial-beans-in-free-pool

Upgrade Advice
Set the minimum size of a bean pool in `weblogic-ejb-jar.xml` using the `initial-beans-in-free-pool` element.

More Information
"Pooling for Stateless Session EJBs" Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server

max-instances

OC4J Definition
This attribute of the `<session-deployment>` element is the presence of this attribute indicates that the application contains customized bean pooling settings

Equivalent Entry in weblogic-ejb-jar.xml
max-beans-in-free-pool

Upgrade Advice
Set the maximum size of a bean pool in `weblogic-ejb-jar.xml` using the `max-beans-in-free-pool` element

More Information
"Pooling for Stateless Session EJBs" in the Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server

max-instances-threshold

OC4J Definition
Use this attribute of the `<session-deployment>` element to set the percentage of `max-instances` number of beans that can be in memory before passivation occurs.

Equivalent Entry in weblogic-ejb-jar.xml
max-beans-in-cache

Upgrade Advice
Use the `stateful-session-cache/max-beans-in-cache` element of `stateful-session-descriptor` to set the idle timeout in `weblogic-ejb-jar.xml`.

More Information
"stateful-session-cache" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

max-tx-retries

OC4J Definition
This attribute of the `<session-deployment>` element specifies the number of times to retry a transaction that was rolled back due to system-level failures. The default is 0.

For a stateful session bean, if a RuntimeException, Error, or RemoteException is thrown, the OC4J does not do a retry.

Equivalent Entry in weblogic-ejb-jar.xml
retry-methods-on-a-rollback/retry-count

Upgrade Advice
To enable retries for all beans in an EJB module in `weblogic-ejb-jar.xml`, use the `retry-methods-on-rollback/retry-count` element.

Note, however, that the behavior differs between OC4J and Oracle WebLogic Server. OC4J specifies retries on a per-EJB basis while WebLogic Server configures retries on a per-EJB module basis.

More Information
"retry-methods-on-rollback" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

resource-check-interval

OC4J Definition
Use this attribute of the `<session-deployment>` element to check all resources at this time interval. At this time, if any of the thresholds have been reached, passivation occurs. Default: 180 sec. (3 min.).

To disable, specify any negative number.

Equivalent Entry in weblogic-ejb-jar.xml
None.

Upgrade Advice
Not supported by Oracle WebLogic Server.

More Information
N/A

passivate-count

OC4J Definition
Use this attribute of the `<session-deployment>` element to define the number of beans to be passivated if any of the resource thresholds have been reached.

Passivation of beans is performed using the least recently used algorithm. Default: one-third of the max-instances attribute. You can disable this attribute by setting the count to zero or a negative number.

Equivalent Entry in weblogic-ejb-jar.xml
None.

Upgrade Advice
Not supported in Oracle WebLogic Server.

More Information
N/A

persistence-filename

OC4J Definition
Use this attribute of the `<session-deployment>` element to define the path to the file where sessions are stored across restarts.

Equivalent Entry in weblogic-ejb-jar.xml
None.

Upgrade Advice
This is not supported by Oracle WebLogic Server; however, you might be able to use an Oracle WebLogic Server custom persistent store.

More Information
"Specifying the Persistent Store Directory for Passivated Beans" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

pool-cache-timeout

OC4J Definition
This attribute of the `<session-deployment>` element specifies how long to keep stateless sessions cached in the pool.

For stateless session beans, if you specify a `pool-cache-timeout`, then at every `pool-cache-timeout` interval all beans of the corresponding bean type in the pool are removed. If the value specified is zero or negative, then the `pool-cache-timeout` is disabled and beans are not removed from the pool.

Equivalent Entry in weblogic-ejb-jar.xm
idle-timeout-seconds

Upgrade Advice
Set the timeout value of a bean pool in `weblogic-ejb-jar.xml` using the `idle-timeout-seconds` element.

More Information

"Pooling for Stateless Session EJBs" in the Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server

timeout

OC4J Definition

Use this attribute of the `<session-deployment>` element to set the maximum number of seconds that a stateful session bean may be inactive before being subject to pool clean-up. If the value is zero or negative, then all timeouts are disabled.

Every 30 seconds the pool clean up logic is invoked. Within the pool clean up logic, only the sessions that timed out, by passing the timeout value, are deleted.

Equivalent Entry in weblogic-ejb-jar.xml

stateful-session-cache

Upgrade Advice

Use the `stateful-session-cache/session-timeout-seconds` element of `stateful-session-descriptor` to set the session timeout in `weblogic-ejb-jar.xml`.

More Information

"stateful-session-cache" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

transaction-timeout

OC4J Definition

This attribute of the `<session-deployment>` element indicates the maximum number of seconds that OC4J will wait for a transaction started by this stateless or stateful session bean to commit or rollback. If the value is zero or negative, the timeout is disabled.

Equivalent Entry in weblogic-ejb-jar.xml

`transaction-descriptor` element and its `trans-timeout-seconds` child element

Upgrade Advice

Set the transaction timeout for an EJB in `weblogic-ejb-jar.xml` using the `transaction-descriptor/trans-timeout-seconds` element and child element.

More Information

"transaction-descriptor" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

<ejb-ref-mapping>

OC4J Definition

This element maps any EJB references to JNDI names.

Before one enterprise bean, acting in the role of a client (call it the source enterprise bean), can access another enterprise bean (call it the target enterprise bean), you must define an EJB reference to the target enterprise bean in the deployment descriptor of the source enterprise bean.

Equivalent Entry in weblogic-ejb-jar.xml
ejb-reference-description

Upgrade Advice
Set the JNDI location mapping for an EJB reference using the `ejb-reference-description` element in `weblogic-ejb-jar.xml`.

More Information
"ejb-reference-description" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

<resource-ref-mapping>

OC4J Definition
This element maps any EJB references to JNDI names.

You can define an environment reference to resource manager connection factories that provide connections to such services as a JDBC data source, JMS topic or queue, Java mail, or an HTTP URL. These references are logical names that OC4J binds at deployment time to the actual resource manager connection factories that it provides.

Equivalent Entry in weblogic-ejb-jar.xml
resource-description

Upgrade Advice
Set the JNDI location mapping for a resource reference using the `resource-description` element in `weblogic-ejb-jar.xml`.

More Information
"resource-description" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

<resource-env-ref-mapping>

OC4J Definition
The `<resource-env-ref-mapping>` element is used to map an administered object for a resource.

For example, to use JMS, the bean must obtain both a JMS factory object and a destination object. These objects are retrieved at the same time from JNDI. The `<resource-ref>` element declares the JMS factory and the `<resource-env-ref>` element is used to declare the destination. Thus, the `<resource-env-ref-mapping>` element maps the destination object.

Equivalent Entry in weblogic-ejb-jar.xml
resource-env-description

Upgrade Advice
Set the JNDI location mapping for a resource environment reference using the `resource-env-description` element in `weblogic-ejb-jar.xml`.

More Information
"resource-env-description" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

<message-destination-ref-mapping>

OC4J Definition
The `<message-destination-ref-mapping>` element is only used if you are using JMS 1.1.

Use this element to map the `message-destination-ref-name` in the client deployment descriptor to another location that is available in the OC4J environment. It provides means of linking message consumers and producers to one or more common logical destinations.

Equivalent Entry in weblogic-ejb-jar.xml
message-destination-descriptor

Upgrade Advice
Set the JNDI location mapping for a message destination mapping using the `message-destination-descriptor` element in `weblogic-ejb-jar.xml`.

More Information
"message-destination-descriptor" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

<session-type>Stateful</session-type>

OC4J Definition
This value for the `session-type` element indicates that the application contains a stateful session bean.

Equivalent Entry in weblogic-ejb-jar.xml
stateful-session-cache/cache-type

Upgrade Advice
The default passivation strategy for stateful session beans differs between OC4J and WebLogic Server.

By default, WebLogic Server uses a "Not Recently Used" passivation model, where passivation only occurs when resource limits have been reached. OC4J follows a strict "Least Recently Used" model where passivation occurs as soon as the idle timeout for a bean is reached.

If eager passivation semantics are required, then set the `stateful-session-cache/cache-type` element to LRU in the `weblogic-ejb-jar.xml` stateful-session-descriptor to preserve OC4J semantics.

More Information
"Stateful Session EJB Passivation" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

<message-driven-deployment>

OC4J Definition
This section of the `orion-ejb-jar.xml` provides additional deployment information for a message driven bean deployed within this JAR file.

The presence of this element indicates that the application contains customized deployment settings for a message driven bean.

Equivalent Entry in weblogic-ejb-jar.xml
message-driven-descriptor

Upgrade Advice
Use the `message-driven-descriptor` element in `weblogic-ejb-jar.xml`.

More Information
"message-driven-descriptor" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

connection-factory-location

OC4J Definition
Use this attribute of the `<message-driven-deployment>` element to define the JNDI location of the connection factory to use. The JMS Destination Connection Factory is specified in this attribute. The syntax is `java:comp/resource` + resource provider name + `TopicConnectionFactories` OR `QueueConnectionFactories` + user defined name. The `nnnConnectionFactories` details what type of factory is being defined.

Equivalent Entry in weblogic-ejb-jar.xml
connection-factory-jndi-name

Upgrade Advice
Use `connection-factory-jndi-name` of the `message-driven-descriptor` element in `weblogic-ejb-jar.xml`.

More Information
"message-driven-descriptor" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

dequeue-retry-count

OC4J Definition
Use this attribute of the `<message-driven-deployment>` element to specify how often the listener thread tries to re-acquire the JMS session once database failover has occurred. This is applicable to only container-managed transactions in an MDB.

Equivalent Entry in weblogic-ejb-jar.xml
None.

Upgrade Advice
A `dequeue-retry-count` is specified for an MDB to configure number of attempts to restart the listener thread to the destination when OC4J detects the destination is down.

There is no direct equivalent in Oracle WebLogic Server. This feature was intended to support failover for Oracle AQ and would be a resource-adapter-specific setting in Oracle WebLogic Server.

More Information
N/A

dequeue-retry-interval

OC4J Definition
Use this attribute of the `<message-driven-deployment>` element to specify the interval between retries

Equivalent Entry in weblogic-ejb-jar.xml
None.

Upgrade Advice
A `dequeue-retry-interval` is specified for an MDB to configure the time interval between attempts to restart the listener thread to the destination when OC4J detects the destination is down.

There is no direct equivalent in Oracle WebLogic Server. This feature was intended to support failover for Oracle AQ and would be a resource-adapter-specific setting in Oracle WebLogic Server.

More Information
N/A

destination-location

OC4J Definition
Use this attribute of the `<message-driven-deployment>` element to define the JNDI location of the destination (queue/topic) to use.

The JMS Destination is specified in the `destination-location` attribute. The syntax is `java:comp/resource` + resource provider name + Topics OR Queues + Destination name. The Topic or Queue details what type of Destination is being defined. The Destination name is the actual queue or topic name defined in the database.

Equivalent Entry in weblogic-ejb-jar.xml
destination-jndi-name

Upgrade Advice
Use the `destination-jndi-name` element of the `message-driven-descriptor` in `weblogic-ejb-jar.xml`.

More Information
"message-driven-descriptor" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

listener-threads

OC4J Definition
Use this attribute of the `<message-driven-deployment>` element to concurrently consume JMS messages. The default is one thread. Topics can only have one thread. Queues can have more than one.

Equivalent Entry in weblogic-ejb-jar.xml
max-beans-in-free-pool

Upgrade Advice
Oracle WebLogic Server supports a variety of approaches for controlling thread management depending on the resource adapter in use.

More Information

"MDBs and Concurrent Processing" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

max-delivery-count

OC4J Definition

Use this attribute of the `<message-driven-deployment>` element to set the maximum number of times OC4J will attempt the immediate redelivery of a message to a message-driven bean's `onMessage` method if that method returns failure (fails to invoke an acknowledgment operation, throws an exception, or both).

After this number of redeliveries, the message is deemed undeliverable and is handled according to the policies of your message service provider. For example, OEMS JMS will put the message on its exception queue (`jms/Oc4jJmsExceptionQueue`).

Equivalent Entry in weblogic-ejb-jar.xml

None.

Upgrade Advice

A `max-delivery-count` is specified for an MDB to specify the maximum number of attempts to deliver the same message in the event of failure. Oracle WebLogic Server does not support setting this option on a per-MDB basis.

More Information

Oracle Fusion Middleware Programming JMS for Oracle WebLogic Server

resource-adapter

OC4J Definition

Use this attribute of the `<message-driven-deployment>` element to define the name of the resource adapter instance that this MDB uses. Applicable only if this MDB is using a J2CA message service provider. In order for the MDB to be activated by messages received by the resource adapter, the MDB and resource adapter must be connected.

Equivalent Entry in weblogic-ejb-jar.xml

resource-adapter-jndi-name

Upgrade Advice

Use `resource-adapter-jndi-name` of `message-destination-descriptor` element in `weblogic-ejb-jar.xml`.

More Information

"message-driven-descriptor" in the *Oracle Fusion Middleware Programming Enterprise JavaBeans for Oracle WebLogic Server*

subscription-name

OC4J Definition

Use this attribute of the `<message-driven-deployment>` element to define the name of the topic to which this message-drive bean subscribes.

Equivalent Entry in weblogic-ejb-jar.xml

None.

Upgrade Advice
If a resource adapter is used, and it supports the `subscription-name` property, then it must be specified using an `activation-config-property` element in `ejb-jar.xml`. Oracle WebLogic Server does not support setting this property in `weblogic-ejb-jar.xml`.

More Information
N/A

wrapper-class

OC4J Definition
Use this attribute of the `<message-driven-deployment>` element to .

Equivalent Entry in weblogic-ejb-jar.xml

Upgrade Advice
A `dequeue-retry-count` is specified for MDB to configure number of attempts to restart the listener thread to the destination when OC4J detects the destination is down.

There is no direct equivalent in Oracle WebLogic Server. This feature was intended to support failover for Oracle AQ and would be a resource-adapter-specific setting in Oracle WebLogic Server.

More Information
N/A

<config-property>

OC4J Definition
The `<config-property>` element is only used if you are using a J2CA message service provider. Use this element to set J2CA resource adapter configuration properties. When OC4J deploys an MDB configured to use a J2CA message service provider, OC4J provides the MDB's activation specification to the resource adapter. This specification includes the properties you set in the `<config-property>` element.

Alternatively, for an EJB 3.0 message-driven bean, you can set J2CA resource adapter configuration properties using @MessageDriven attribute `configProperty` and @ActivationConfig annotation.

You can use the `orion-ejb-jar.xml` file `<config-property>` configuration to override @MessageDriven configuration.

Equivalent Entry in weblogic-ejb-jar.xml
None.

Upgrade Advice
Oracle WebLogic Server does not support overriding activation config properties via `weblogic-ejb-jar.xml`. These settings must be merged with the normal activation config properties located in the `ejb-jar.xml` file.

More Information
N/A

Index

Numerics

10g Release 3 (10.1.3.1.0)
  components included in, 2-1

A

access-mask
  element of orion-web.xml, A-3
ADF applications, 4-2
admin_client.jar
  equivalent in WebLogic Server, 3-8
Administration Console, 3-2
  starting and stopping components, 3-9
administration server, 3-2, 3-5
administration tools
  for OC4J and WebLogic Server, 3-8
AJP, 3-6, 7-2
  no support in WebLogic Server, 3-7
  using to integrate Oracle HTTP Server with OC4J, 7-1
AJPS, 3-6
Ant
  See Apache Ant
Apache Ant
  using for WebLogic administration tasks, 3-9
API support
  when upgrading to WebLogic Server, 4-4, 4-5
application clients
  changes required when JMS provider is upgraded to WebLogic Server, 6-4
  changes required when using OC4J JMS provider, 6-5
  EJB clients, 6-3
  modifying to use WebLogic Server JNDI provider, 6-1
  remote standalone EJB clients, 6-3
  running in an OC4J server instance, 6-2
  upgrade, 6-1
  using OC4J-based EJB interfaces
    impact of upgrade on, 6-4
Application Server Control, 3-3, 3-6, 3-8, 3-9
  compared to WebLogic Server Administration Console, 3-2
application-client.xml, 4-3
applications
  See Java EE applications
application.xml, 4-2, 4-3
architecture
  comparing OC4J and Oracle WebLogic Server, 3-2
autojoin-session
  attribute of the web-app-class-loader element, A-9

C

classpath
  element of orion-web.xml, A-2
cluster topology
  compared with WebLogic domain, 3-4
  compared with WebLogic Server, 3-3
Cluster Topology page
  in Application Server Control, 3-3
clustering
  comparing OC4J and WebLogic Server, 3-3
  comparison of WebLogic and OC4J features, 3-4
clusters
  WebLogic Server clusters, 3-4
config-property
  element in orion-ejb-jar.xml, A-27
Configuration Wizard
  introduction for OC4J users, 3-7
Connection Pooling
  when upgrading to WebLogic Server JMS provider, 6-5
connection-factory-location
  attribute of the message-driven-deployment element, A-24
context-param
  element of web.xml, A-2
contextParamMappingFinding
  element of orion-web.xml, A-2
copy-by-value
  attribute of the session-deployment element, A-17
custom Java EE applications, 3-5

D

default-buffer-size
  attribute of the web-app-class-loader

element, A-9
default-charset
attribute of the web-app-class-loader element, A-10
default-mime-type
attribute of the web-app-class-loader element, A-10
default-web-site.xml, 7-2
deployment descriptors
comparison beween OC4J and WebLogic Server, 4-2
guidelines and resources for upgrading, 4-3
security elements in, 4-4
deployment plans
generating with weblogic.PlanGenerator, 4-4
upgrading, 4-4
dequeue-retry-count
attribute of the message-driven-deployment element, A-24
dequeue-retry-interval
attribute of the message-driven-deployment element, A-25
destination-location
attribute of the message-driven-deployment element, A-25
development
attribute of the web-app-class-loader element, A-11
development tools
selecting for upgrade, 4-1
diagnostics
comparison between OC4J and WebLogic Server, 3-9
Diagnostics and Logging Framework (ODL), 4-5
directory structures
comparison between OC4J and WebLogic Server, 3-4
directory-browsing
attribute of the web-app-class-loader element, A-11
DMS
compared with WebLogic Server diagnostics, 3-9
domain
comparison with 10*g* cluster topology, 3-4
content and organization of, 3-5
creating in the Middleware home, 3-4
domains
configuring multiple, 3-6
creating multiple, 3-6
default location of domain files, 3-6
introduction for OC4J users, 3-5
viewing domain log files, 3-10
Dynamic Monitoring System (DMS), 4-5

E

EJB
comparison of support in OC4J and WebLogic Server, 3-11
EJB clients
See also Enterprise Java Bean Clients, 6-3
ejb-jar.xml, 4-3, 4-4
ejb-ref-mapping
element in orion-ejb-jar.xml, A-21
element of orion-web.xml, A-6
enable-jsp-dispatcher-shortcut
attribute of the web-app-class-loader element, A-11
Enterprise Java Bean clients
impact of upgrade on, 6-3
remote standalone EJB clients, 6-3
running within an OC4J server instance, 6-3
env-entry-mapping
element of orion-web.xml, A-6
environment-naming-url-factory-enabled
server.xml attribute, 6-2
to allow use of multiple JNDI providers, 6-2
expiration-setting
element of orion-web.xml, A-7

F

file-modification-check-interval
attribute of the web-app-class-loader element, A-12
flow chart
of the Java EE upgrade process, 1-1
foreign server
WebLogic JMS feature, 6-5
Fusion Middleware Control
compared to Application Server Control, 3-8
using to verify a Web tier upgrade, 7-11

H

HTTP, 3-6, 7-2
support in WebLogic Server, 3-7
HTTP session state replication, 6-1
in WebLogic Server, 3-4
HTTPS, 3-6, 7-2

I

idletime
attribute of the session-deployment element, A-18
include-war-manifest-class-path
attribute of the web-app-class-loader element, A-9
installation and configuration
separation of in WebLogic server, 3-7
Integrated Web Server and OC4J
10*g* installation type, 3-2

J

j2ee directory, 3-4, 3-6
Java components, 3-5
Java EE
comparison of support in OC4J and WebLogic Server, 3-10

upgrading your applications, 4-1
Java EE and Web Server
OC4J installation type
upgrading, 7-1
Java EE applications
selecting development tools for, 4-1
upgrading, 4-1
upgrading clients, 6-1
upgrading deployment descriptors, 4-2
upgrading Web services, 4-6
verifying on OC4J before upgrade, 4-1
verifying with supported JDK, 4-2
Java EE Deployment
comparison of support in OC4J and WebLogic Server, 3-11
Java EE Management
comparison of support in OC4J and WebLogic Server, 3-11
Java EE upgrade
process summary, 1-1
Java Messaging Service
WebLogic "Unit of Order" feature, 6-4
WebLogic foreign server feature, 6-5
Java Messaging Service clients
upgrading, 6-4
Java Naming and Directory Interface clients
impact of upgrading on, 6-1
Java Required Files domain template
APIs available in, 4-4
Java SE
comparison of support in OC4J and WebLogic Server, 3-10
Java Server Pages clients
impact of upgrading on, 6-1
Java Standard Tag Library (JSTL), 4-6
Java standards
support in OC4J and WebLogic Server, 3-10
Java Virtual Machine
comparison between JVMs in WebLogic and OC4J, 3-6
configuring multiple in OC4J, 3-6
JAX-RPC, 4-6
comparison of support in OC4J and WebLogic Server, 3-11
JAX-WS, 4-6
comparison of support in OC4J and WebLogic Server, 3-11
jazn-web-app
element of orion-web.xml, A-7
JCA
comparison of support in OC4J and WebLogic Server, 3-11
JDBC
comparison of support in OC4J and WebLogic Server, 3-11
JDK
supported version for Oracle Fusion Middleware, 4-2
JMS, 3-7
comparison of support in OC4J and WebLogic Server, 3-11
JMS Clients
See also Java Messaging Service clients, 6-4
JMS provider
client impact when upgraded, 6-4
JMX
comparison of support in OC4J and WebLogic Server, 3-11
JNDI
comparison of support in OC4J and WebLogic Server, 3-11
JNDI clients
See also Java Naming and Directory Interface clients, 6-1
JNDI Namespace
understanding scope of WebLogic Server, 6-2
JRF domain template
See also Java Required Files domain template, 4-4
JSF
comparison of support in OC4J and WebLogic Server, 3-10
JSP
comparison of support in OC4J and WebLogic Server, 3-10
JSP clients
See also Java Server Pages clients, 6-1
jsp-cache-directory
attribute of the web-app-class-loader element, A-12
jsp-cache-tlds
attribute of the web-app-class-loader element, A-12
jsp-print-null
attribute of the web-app-class-loader element, A-13
jsp-taglib-locations
attribute of the web-app-class-loader element, A-13
jsp-timeout
attribute of the web-app-class-loader element, A-13
jstl.jar, 4-5
JTA
comparison of support in OC4J and WebLogic Server, 3-11
JVM
See Java Virtual Machine

L

library-ref
element of weblogic-application.xml, A-2
listener-threads
attribute of the message-driven-deployment element, A-25
Load Balancing
when upgrading EJB clients, 6-3
log files
location of domain log files, 3-10
viewing for a domain, 3-10

logging
comparison between OC4J and WebLogic Server, 3-10
lookup-context
element of orion-web.xml, A-5

M

managed servers, 3-5, 7-2
introduction to, 3-2
max-delivery-count
attribute of the message-driven-deployment element, A-26
max-instances
attribute of the session-deployment element, A-18
max-instances-threshold
attribute of the session-deployment element, A-19
max-tx-retries
attribute of the session-deployment element, A-19
Message ordering
when upgrading to WebLogic Server JMS provider, 6-4
message-destination-ref-mapping
element in orion-ejb-jar.xml, A-23
message-driven-deployment
element in orion-ejb-jar.xml, A-23
Middle Tier Instance
option in the Upgrade Assistant, 7-7
Middleware home
introduction to, 3-4
mimeMappings
element of orion-web.xml, A-3
min-instances
attribute of the session-deployment element, A-18
mod_oc4j, 7-2
mod_wl_ohs
using to configure Oracle HTTP Server with WebLogic, 7-3
My Oracle Support, 2-1

N

network channels, 7-2
WebLogic Server feature, 3-7
Network Connections
when upgrading to WebLogic Server JMS provider, 6-5
numproc
OC4J configuration property, 3-6

O

OC4J, 1-1
clustering, 3-3
comparing with WebLogic architecture, 3-1
comparison with WebLogic clustering features, 3-4
introduction to Oracle WebLogic Server for users of, 3-1
introduction to WebLogic installation and configuration for users of, 3-7
managing with Application Server Control, 3-2
Oracle WebLogic Server concepts for users of, 3-1
standalone, 3-2
verifying that applications deploy on, 4-1
Web sites
equivalent in WebLogic Server, 7-1
WSDL
generating Web services from, 4-7
OC4J application clustering, 3-3
comparison with WebLogic HTTP session state replication, 3-4
OC4J groups, 3-3
compared with WebLogic clusters, 3-4
OC4J JMX MBeans, 4-6
OC4J Job Scheduler, 4-6
OC4J Support for JSP, 4-6
OC4J-Based EJB Interfaces, 6-4
oc4jclient.jar, 6-4
oc4j-ra.xml, 4-3
ODL
See Oracle Diagnostics Logging
OEMS JMS Connector, 6-5
ojsputil.jar, 4-5
OPMN, 3-9
See also Oracle Process Management and Notification, 6-2
opmnctl, 3-8
Oracle Business Intelligence Discoverer, 7-3
Oracle Coherence, 6-1
Oracle Containers for Java EE
See OC4J
Oracle Diagnostics Logging
support in OC4J and WebLogic Server, 3-10
Oracle Enterprise Manager Application Server Control
See Application Server Control
Oracle Enterprise Manager Fusion Middleware Control
See Fusion Middleware Control, 3-8
Oracle Fusion Middleware
configuring a Web tier middle tier, 7-3
Java components, 3-5
Oracle Globalization Development Kit, 4-5
Oracle home, 3-4, 3-6
Oracle HTTP Client, 4-5
Oracle HTTP Server, 3-7, 7-8
comparison between using with OC4J and WebLogic, 7-1
configuring to work with WebLogic Server, 7-2
integrated with OC4J, 7-1
modifying the listening port after upgrade, 7-9
routing requests to OC4J, 3-7
understanding how to configure with Oracle WebLogic Server, 7-2
upgrading, 7-1
using Web servers other than, 7-3

Oracle Identity Federation, 7-3
Oracle Identity Management, 7-3
Oracle Java Object Cache, 4-5
Oracle JAZN (Java Authorization), 4-5
Oracle JSP Tag Libraries, 4-5
Oracle Platform Security Services (OPSS), 4-5
Oracle Portal, 7-3
Oracle Process Management and Notification (OPMN), 6-2
Oracle Security Developer Tools, 4-5
Oracle Service Registry, 4-6
Oracle Technology Network (OTN), 4-6
Oracle TopLink, 4-5
Oracle Web Cache, 7-3, 7-8
  verifying ports after upgrade, 7-10
  verifying the location of SSL wallet after upgrade, 7-9
Oracle Web Cache Invalidation, 4-6
Oracle Web Services Proxy, 4-6
Oracle Web Services SOAP, 4-6
Oracle Web Services UDDI Client, 4-6
Oracle Web Services UDDI Client API, 4-6
Oracle WebLogic Server
  additional facts for OC4J users, 3-6
  Administration Console, 3-2
  administration tools
    comparison with OC4J, 3-8
  API requirements, 4-5
  API support, 4-4
  associating a Web tier with, 7-4
  clustering, 3-3
  command-line scripting tool
    See also WLST, 3-8
  comparing with OC4J architecture, 3-1
  comparison to OC4J clustering features, 3-4
  concepts for OC4J users, 3-1
  configuration wizard, 3-6
  configuring Oracle HTTP Server to work with, 7-2
  configuring with Web tier components, 3-2
  directory structure, 3-4
  generating Web services from OC4J WSDL, 4-7
  HTTP session state replication, 3-4
  in-memory HTTP session state replication, 6-1
  installation and configuration concepts for OC4J users, 3-7
  introduction for OC4J users, 3-1
  introduction to domains for OC4J users, 3-5
  JAX-RPC Web services, 4-6
  JAX-WS Web Services, 4-6
  JMS "Unit of Order" feature, 6-4
  JNDI Namespace
    understanding scope of, 6-2
  JNDI provider, 6-3
    modifying clients to use, 6-1
  listening ports, 3-6
  managed server listening ports, 3-7
  managed servers compared to OC4J instances, 3-2
  network channels, 3-7
  standards support, 3-10
  support for ODL, 3-10
  typical administration tasks, 3-9
  using Oracle HTTP Server with, 7-1
  using with a Web server, 3-2
Oracle WebLogic Server installer
  introduction for OC4J users, 3-7
Oracle XML, 4-5
OracleAS Web Services, 4-6
OracleMetaLink, 2-1
oracle-webservices.xml, 4-3
orion-application-client.xml, 4-3
orion-application.xml, 4-3
orion-ejb-jar.xml, 4-3
  upgrade reference, A-1, A-17
orion-web.xml, 4-3
  upgrade reference, A-1

P

passivate-count
  attribute of the session-deployment element, A-20
patch sets
  applying the latest before upgrade, 2-1
persistence-filename
  attribute of the session-deployment element, A-20
persistence-path
  attribute of the web-app-class-loader element, A-14
pool-cache-timeout
  attribute of the session-deployment element, A-20
post-upgrade tasks
  for Web tier components, 7-9
  verifying the location of the Web Cache SSL wallet, 7-9
  verifying Web Cache ports, 7-10

R

ra.xml, 4-3
request-tracker
  element of orion-web.xml, A-4
resource-adapter
  attribute of the message-driven-deployment element, A-26
resource-check-interval
  attribute of the session-deployment element, A-19
resource-env-ref-mapping
  element in orion-ejb-jar.xml, A-22
  element of orion-web.xml, A-6
resource-ref-mapping
  element in orion-ejb-jar.xml, A-22
  element of orion-web.xml, A-5
RMI, 3-7, 7-2
RMI Protocol
  when upgrading EJB clients, 6-3

RMIInitialContextFactory, 6-4

S

SAML, 4-7
schema-major-version
  attribute of the web-app-class-loader element, A-14
schema-minor-version
  attribute of the web-app-class-loader element, A-14
search-local-classes-first
  attribute of the web-app-class-loader element, A-8
security role mappings, 4-4
security-role-mapping
  element of orion-web.xml, A-8
service-ref-mapping
  element of orion-web.xml, A-7
Servlet
  comparison of support in OC4J and WebLogic Server, 3-11
Servlet clients
  impact of upgrading on, 6-1
servlet-chaining
  element of orion-web.xml, A-4
servlet-webdir
  attribute of the web-app-class-loader element, A-15
session-deployment
  element in orion-ejb-jar.xml, A-17
session-tracker
  element of orion-web.xml, A-5
session-tracking
  element of orion-web.xml, A-4
session-type
  element in orion-ejb-jar.xml
    when set to stateful, A-23
simple-jsp-mapping
  attribute of the web-app-class-loader element, A-15
SmartUpgrade, 1-3, 4-2
SOA applications, 4-2
SOAP 1.1 and 1.2, 4-7
source-directory
  attribute of the web-app-class-loader element, A-15
Specify Destination Instance screen
  in the Upgrade Assistant, 7-7
Specify Operation screen
  of the Upgrade Assistant, 7-6
Specify Source Home screen
  in the Upgrade Assistant, 7-7
Specify Upgrade Options screen
  in the Upgrade Assistant, 7-8
Specify WebLogic Server screen
  in the Upgrade Assistant, 7-7
SSL, 7-2
  listening port in WebLogic Server, 3-7
standalone OC4J instances, 3-2
standards
  comparison of OC4J and WebLogic Server support for, 3-10
Start destination components after successful upgrade
  in the Upgrade Assistant, 7-8
starting points
  for Java EE upgrade, 2-1
starting servers
  in OC4J and WebLogic Server, 3-9
startManagedWebLogic
  WebLogic server script, 3-9
startWebLogic
  WebLogic server script, 3-9
stateful-session-descriptor
  in weblogic-ejb-jar.xml, A-17
stateless-session-descriptor
  in weblogic-ejb-jar.xml, A-17
Stop source components before upgrade option
  in the Upgrade Assistant, 7-8
stopping servers
  in OC4J and Oracle WebLogic Server, 3-9
subscription-name
  attribute of the message-driven-deployment element, A-26

T

t3 protocol
  used by WebLogic Server, 6-2
temporary-directory
  attribute of the web-app-class-loader element, A-16
thread pools
  comparison between OC4J and WebLogic Server, 3-10
timeout
  attribute of the session-deployment element, A-21
transaction-timeout
  attribute of the session-deployment element, A-21

U

ua
  command to start the Upgrade Assistant, 7-5
ua.bat
  command to start the Upgrade Assistant, 7-5
UDDI, 4-6
upgrade
  summary of the process, 1-1
Upgrade Assistant
  screens that require input during Web tier upgrade, 7-7
  Specify Destination Instance screen, 7-7
  Specify Operation screen, 7-6
  Specify Source Home screen, 7-7
  Specify Upgrade Options screen, 7-8
  Specify WebLogic Server screen, 7-7
  Start destination components after successful

upgrade option, 7-8
starting in preparation for a Web tier upgrade, 7-5
Stop source components before upgrade option, 7-8
upgrading Web tier components, 7-7
Use source Oracle home ports in destination option, 7-8
important notes when using, 7-8
Welcome screen, 7-5
Use source Oracle home ports in destination option, 7-8
important notes when using, 7-8
in the Upgrade Assistant, 7-8
user_projects directory, 3-4
user_projects/domains
default location of WebLogic domain files, 3-6

V

Verify Instance option
in Upgrade Assistant, 7-10
verifying a Web tier upgrade, 7-10
virtual-directory
element of orion-web.xml, A-3

W

Web Server
integrating with OC4J and Oracle WebLogic Server, 3-2
upgrading, 7-1
Web servers
other than Oracle HTTP Server
configuring with Oracle WebLogic, 7-3
Web services
generating Web service from OC4J WSDL, 4-7
JAX-RPC, 4-6
JAX-WS, 4-6
Oracle recommended upgrade path, 4-7
specifications supported by OC4J and WebLogic Server, 4-7
upgrading, 4-6
Web sites
in OC4J, 3-6
Web tier
associating with a WebLogic Server domain, 7-4
configuring to route requests to WebLogic Server, 7-9
configuring without a domain, 7-4
installation and configuration, 7-2
installing and configuring, 7-3
installing and configuring with Oracle WebLogic Server, 3-2
locating the installation and configuration documentation, 7-4
post-upgrade tasks, 7-9
selecting components, 7-3
upgrade screens that require input, 7-7
upgrading, 7-5
upgrading Web tier components, 7-1
verifying the upgrade, 7-10
Web Tier and Utilities CD-ROM, 3-2
web-app-class-loader
element of orion-web.xml, A-8
WebCenter applications, 4-2
WEB-INF/lib, 4-5
WEB-INF/tld directory, 4-5
WebLogic Diagnostics Framework
compared with Dynamic Monitoring Server (DMS), 3-9
WebLogic Server
See Oracle WebLogic Server
WebLogic Server JMX MBeans, 4-6
weblogic-appclient.xml, 4-3
resources for upgrading, 4-3
weblogic-application.xml, 4-3, A-2
resources for upgrading, 4-3
weblogic.deployer, 3-9
weblogic-ejb-jar.xml, 4-3, 6-3
resources for upgrading, 4-3
weblogic.PlanGenerator, 4-4
weblogic-ra.xml, 4-3
resources for upgrading, 4-3
weblogic-webservices.xml, 4-3
resources for upgrading, 4-4
weblogic.xml, 4-3
resources for upgrading, 4-3
webservices.xml, 4-3
web.xml, 4-2, 4-3, 4-4
Welcome screen
of the Upgrade Assistant, 7-5
WLDF
See WebLogic Diagnostics Framework
WLInitialContextFactory class
when upgrading application clients, 6-2
WLST, 3-8, 3-9
wrapper-class
attribute of the message-driven-deployment element, A-27
WS-Addressing, 4-7
WS-Conversation, 4-7
WSDL 1.1, 4-7
WS-I 1.0 and 1.1, 4-7
WS-Policy, 4-7
WS-PolicyAttachment, 4-7
WS-Reliability, 4-7
WS-ReliableMessaging, 4-8
WS-SecureConversation, 4-8
WS-SecurePolicy, 4-7
WS-Security, 4-7
WS-Trust, 4-7

X

XML Encryption, 4-7
XML Signature, 4-7