

Oracle® Application Server Containers for J2EE

User's Guide

10g Release 2 (10.1.2)

Part No. B14011-02

July 2005

Oracle Application Server Containers for J2EE User's Guide, 10g Release 2 (10.1.2)

Part No. B14011-02

Copyright © 2002, 2005, Oracle. All rights reserved.

Primary Authors: Sheryl Maring, Dan Hynes

Contributing Authors: Brian Wright, Timothy Smith

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software—Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Intended Audience.....	ix
Documentation Accessibility.....	ix
Related Documentation.....	x
Conventions.....	x
1 OC4J Overview	
Introduction to OC4J	1-1
Examples Provided with OC4J.....	1-2
JDK 1.4 Considerations: Cannot Invoke Classes Not in Packages	1-2
Migrating Source Code from JDK 1.4 to JDK 1.3.....	1-3
Migrating Source Code from JDK 1.3 to JDK 1.4.....	1-3
Navigating the OC4J Documentation Set	1-4
OC4J Installation	1-4
Using OC4J in an Enterprise or Standalone Environment	1-4
Managing Multiple OC4J Instances in an Enterprise Environment.....	1-5
Managing a Single OC4J Instance.....	1-5
Using JDK With OC4J.....	1-6
OC4J Documentation Set Assumptions.....	1-6
OC4J Communication	1-7
HTTP Communication.....	1-7
Requirements.....	1-7
2 Configuration and Deployment	
OC4J Home Page Overview	2-1
Applications Page.....	2-2
Administration Page.....	2-3
Starting and Stopping OC4J	2-3
Testing the Default Configuration.....	2-4
Creating the Development Directory	2-5
Configuring the FAQ Application Demo	2-6
Environment Setup for FAQ Demo.....	2-7
Oracle Database.....	2-7
OC4J System Configuration for FAQ Demo.....	2-7
Data Source Configuration.....	2-7
Security Configuration.....	2-8

Deploy the FAQ Demo	2-9
Deployment Details Explained	2-9
Deploying Applications	2-11
Basic Deployment.....	2-11
Select Application	2-12
Provide The URL Mappings For All Web Modules	2-13
IIOP Stub Generation	2-14
Provide Any Resource Reference Mappings	2-14
Specify Any User Manager.....	2-16
Provide Any Security Role Mappings	2-16
Publish Web Services	2-17
Deployment Review	2-17
Post-Deployment Application Modifications.....	2-18
Impact of Undeploying/Redeploying an Application.....	2-18
Impact of "Hot Deploying" an Application.....	2-19
What Happens When You Deploy?	2-19
OC4J Tasks During Deployment	2-19
Verifying the Configuration of Deployed J2EE Applications	2-20
Recovering From Deployment Errors	2-21
Undeploying Web Applications	2-21

3 Advanced Configuration and Development

Configuring OC4J Using Enterprise Manager	3-1
OC4J Instance Level Configuration.....	3-1
Configuring Server Properties	3-2
Configure a Web Site.....	3-4
Configure JSP Container Parameters.....	3-5
Configure Replication Parameters	3-6
Advanced Configuration Through XML Files.....	3-6
Configure Data Sources	3-7
Data Source Field Page	3-8
Configure Security	3-10
Configure JMS	3-10
Edit the JMS Section	3-11
Edit the JMS XML File.....	3-12
Configure Global Web Application Parameters.....	3-12
Configure RMI.....	3-13
Application Level Configuration.....	3-13
Configuring Application General Parameters.....	3-14
Configuring Local J2EE Services.....	3-14
Modifying XML Files Included in the Deployed Application EAR File.....	3-14
Overview of OC4J and J2EE XML Files	3-14
XML Configuration File Overview	3-15
XML File Interrelationships.....	3-17
Sharing Libraries	3-19
Understanding and Configuring OC4J Listeners	3-20

HTTP Requests	3-20
RMI Requests	3-21
Disabling Access Logging for a Web Application.....	3-21
Configuring Oracle HTTP Server With Another Web Context	3-22
Building and Deploying Within a Directory	3-22
Developing Startup and Shutdown Classes	3-25
OC4J Startup Classes	3-25
OC4J Shutdown Classes	3-27
Considerations for Startup and Shutdown Classes	3-28
Setting Performance Options	3-28
Performance Command-Line Options	3-28
Thread Pool Settings	3-29
Statement Caching	3-31
Task Manager Granularity.....	3-31
Enabling OC4J Logging	3-31
Viewing OC4J System and Application Log Messages	3-32
Oracle Diagnostic Logging (ODL) Log Files.....	3-32
Text Log Files.....	3-34
Redirecting Standard Out and Standard Error	3-35
OC4J Debugging	3-36
Servlet Debugging Example	3-38
Remote Debugging Using Oracle JDeveloper	3-39

4 OC4J Clustering

The OC4J Instance in a Cluster.....	4-1
The OC4J Process in a Cluster	4-2
Replicating Application State.....	4-3
Islands	4-3
J2EE Applications Involved in a Cluster	4-4
Instance-Specific Parameters	4-5
OC4J Clustering Examples	4-5
Software Failure	4-5
Hardware Failure	4-6
State Replication.....	4-7
OC4J Cluster Configuration.....	4-7
OC4J Instance Configuration.....	4-8
Configuring Islands and Processes	4-8
Configuring Web Application State Replication.....	4-9
Configuring EJB Application State Replication	4-10
Stateful Session Bean Replication Configuration in the Application JAR.....	4-12
JVM Termination Replication	4-12
End of Call Replication	4-12
Configuring OC4J Instance-Specific Parameters	4-13

A Troubleshooting OC4J

Problems and Solutions	A-1
-------------------------------------	------------

Unable to Start OC4J When Using JDK 1.3	A-1
Unable to Restart OC4J After Abnormal Termination When OracleAS JMS is Active	A-2
Stateful Replication Not Consistent Across OC4J instances	A-2
Using A Non-Certified Version of the JDK for OC4J Only	A-2
java.lang.OutOfMemory Errors Thrown When Running OC4J.....	A-2
Connection Timeouts Through a Stateful Firewall Affect System Performance	A-3
OPMN-Managed OC4J Unable to Access EJB Resources Via the Default RMI Port	A-4
Application Performance Impacted by JVM Garbage Collection Pauses	A-4
Invalid or Unneeded Library Elements Degrade Performance.....	A-5
JSP Error: Tag Not Registered	A-5
JSP Error: Zero-Length Class File	A-5
JSP Error: Illegal use of <when>-style tag without <choose> as its direct parent	A-6
Need More Help?	A-6

B Additional Information

Description of XML File Contents	B-1
OC4J Configuration XML Files	B-1
server.xml	B-1
default-web-site.xml	B-2
jazn-data.xml	B-2
principals.xml	B-3
data-sources.xml	B-3
jms.xml.....	B-3
rmi.xml	B-3
J2EE Deployment XML Files	B-4
The J2EE application.xml File	B-4
The OC4J-Specific orion-application.xml File.....	B-4
The J2EE ejb-jar.xml File	B-4
The OC4J-Specific orion-ejb-jar.xml File.....	B-5
The J2EE web.xml File.....	B-5
The OC4J-Specific orion-web.xml File	B-5
The J2EE application-client.xml File	B-6
The OC4J-Specific orion-application-client.xml File.....	B-6
Elements in the server.xml File	B-6
Configure OC4J	B-6
Reference Other Configuration Files.....	B-6
<application-server> Element Description	B-6
Elements Contained Within <application-server>	B-7
Elements in the application.xml File	B-14
<application> Element Description	B-15
Elements Contained Within <application>.....	B-15
Elements in the orion-application.xml File	B-16
<orion-application> Element Description	B-16
Elements Contained Within <orion-application>.....	B-16
Elements in the application-client.xml File	B-22
<application-client> Element Description	B-22
<application-client>	B-22

Elements Contained Within <application-client>.....	B-22
Elements in the orion-application-client.xml File	B-24
<orion-application-client> Element Description	B-24
Elements Contained Within <orion-application-client>	B-25
Configuration and Deployment Examples	B-26
Configuration File Examples	B-26
application.xml Example	B-26
web.xml Example.....	B-27
ejb-jar.xml Example	B-27
server.xml Addition.....	B-28
default-web-site.xml Addition.....	B-29
Client Example	B-29
JNDI Properties for the Client.....	B-29
Client Module—Standalone Java Client Invoking EJBs.....	B-30
Manifest File for the Client	B-30
Executing the Client	B-30
OC4J Command-Line Options and System Properties	B-30

C Third Party Licenses

Third-Party Licenses	C-1
Apache HTTP Server	C-1
The Apache Software License	C-1

Index

Preface

This preface introduces you to the *Oracle Application Server Containers for J2EE User's Guide*, discussing the intended audience, structure, and conventions of this document. It also provides a list of related Oracle documents.

Intended Audience

This manual is intended for anyone who is interested in using Oracle Application Server Containers for J2EE (OC4J). It assumes that you have basic knowledge of the following:

- Java and J2EE
- XML
- JDBC

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. 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.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documentation

For more information on OC4J, see the following documentation available from other OC4J manuals:

- *Oracle Application Server Containers for J2EE Services Guide*
- *Oracle Application Server Containers for J2EE Servlet Developer's Guide*
- *Oracle Application Server Containers for J2EE Security Guide*
- *Oracle Application Server Containers for J2EE Support for JavaServer Pages Developer's Guide*
- *Oracle Application Server Containers for J2EE JSP Tag Libraries and Utilities Reference*
- *Oracle Application Server Containers for J2EE Servlet Developer's Guide*
- *Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide*

The following documentation may also be helpful in understanding OC4J:

- *Oracle Application Server Administrator's Guide*
- *Oracle Application Server Performance Guide*
- *Oracle Application Server High Availability Guide*
- *Oracle9i JDBC Developer's Guide and Reference*
- *Oracle HTTP Server Administrator's Guide*
- *Oracle Application Server DMS API Reference*

Conventions

The following text conventions are used in this document:

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

OC4J Overview

This chapter describes OC4J and demonstrates how to install OC4J with the Oracle Application Server installation.

This chapter includes the following topics:

- [Introduction to OC4J](#)
- [JDK 1.4 Considerations: Cannot Invoke Classes Not in Packages](#)
- [Navigating the OC4J Documentation Set](#)
- [OC4J Installation](#)
- [Using OC4J in an Enterprise or Standalone Environment](#)
- [OC4J Communication](#)

Introduction to OC4J

Oracle Application Server provides a complete Java 2 Enterprise Edition (J2EE) environment written entirely in Java that executes in the Java virtual machine (JVM) of the standard Java Development Kit (JDK). You can run Oracle Application Server Containers for J2EE (OC4J) on the standard JDK that exists on your operating system. Refer to the certification matrix on <http://www.oracle.com/technology/index.html>.

OC4J is J2EE certified and provides all the containers, APIs, and services that J2EE specifies. OC4J is based on technology licensed from Ironflare Corporation, which develops the Orion server—one of the leading J2EE containers. Although OC4J is integrated with the Oracle Application Server infrastructure, the product and some of the documentation still contains some reference to the Orion server.

OC4J supports and is certified for the standard J2EE APIs, as listed in [Table 1-1](#).

Table 1-1 OC4J J2EE Support

J2EE 1.3 Standard APIs	Version Supported
JavaServer Pages (JSP)	1.2
Servlets	2.3
Enterprise JavaBeans (EJB)	2.0
Java Transaction API (JTA)	1.0
Java Message Service (JMS)	1.0
Java Naming and Directory Interface (JNDI)	1.2

Table 1–1 OC4J J2EE Support (Cont.)

J2EE 1.3 Standard APIs	Version Supported
Java Mail	1.1.2
Java Database Connectivity (JDBC)	2.0 Extension
JAAS	1.0
J2EE Connector Architecture (JCA)	1.0
JAXP	1.1

The OC4J documentation assumes that you have a basic understanding of Java programming, J2EE technology, and Web and EJB application technology. This includes deployment conventions such as the `/WEB-INF` and `/META-INF` directories.

Examples Provided with OC4J

Oracle Application Server 10g (9.0.4) includes one demo per component. Additional demos are available for download from OTN at the following URL:

<http://www.oracle.com/technology/tech/java/oc4j/demos/>

Drill down from this index page to the component areas that contain the additional demos that you need.

JDK 1.4 Considerations: Cannot Invoke Classes Not in Packages

Among the migration considerations in moving to a Sun Microsystems JDK 1.4 environment, which is the environment that is shipped with Oracle Application Server 10g, there is one of particular importance to servlet and JSP developers.

As stated by Sun Microsystems, "The compiler now rejects import statements that import a type from the unnamed namespace." (This was to address security concerns and ambiguities with previous JDK versions.) Essentially, this means that you cannot invoke a class (a method of a class) that is not within a package. Any attempt to do so will result in a fatal error at compilation time.

This especially affects JSP developers who invoke JavaBeans from their JSP pages, as such beans are often outside of any package (although the JSP 2.0 specification now requires beans to be within packages, in order to satisfy the new compiler requirements). Where JavaBeans outside of packages are invoked, JSP applications that were built and executed in an OC4J 9.0.3 / JDK 1.3.1 environment will no longer work in an Oracle Application Server 10g / JDK 1.4 environment.

Until you update your application so that all JavaBeans and other invoked classes are within packages, you have the alternative of reverting back to a JDK 1.3.1 environment to avoid this issue.

Notes:

- The `javac -source` compiler option is intended to allow JDK 1.3.1 code to be processed seamlessly by the JDK 1.4 compiler, but this option does not account for the "classes not in packages" issue.
 - Only the JDK 1.3.1 and JDK 1.4 compilers are supported and certified by OC4J. It is possible to specify an alternative compiler by adding a `<java-compiler>` element to the `server.xml` file, and this might provide a workaround for the "classes not in packages" issue, but no other compilers are certified or supported by Oracle for use with OC4J. (Furthermore, do *not* update the `server.xml` file directly in an Oracle Application Server environment. Use the Oracle Enterprise Manager 10g.)
-

For more information about the "classes not in packages" issue and other JDK 1.4 compatibility issues, refer to the following Web site:

<http://java.sun.com/j2se/1.4/compatibility.html>

In particular, click the link "Incompatibilities Between Java 2 Platform, Standard Edition, v1.4.0 and v1.3".

Migrating Source Code from JDK 1.4 to JDK 1.3

Java does not support compiling Java code under JDK 1.4 and running it under JDK 1.3 (bug 2811379). If you try anyway, Java may throw one of the following error messages at runtime:

- Unsupported major.minor version 48.0.
- The major.minor version '48.0' is too recent for this tool to understand.

However, if you must generate class files using JDK 1.4 and run them using JDK 1.3, then you can use the following command to force a JDK 1.4 compiler to generate a class file that is compatible with JDK 1.3:

```
% javac -target 1.3 hello.java
```

Migrating Source Code from JDK 1.3 to JDK 1.4

When you upgrade from JDK 1.3 to JDK 1.4, put all classes into packages. The JDK 1.4 specification explicitly prohibits a class within a package from invoking a class not within a package. For this reason, some Java source code that compiles in JDK 1.3 will not compile in JDK 1.4. This error is identified by the following compiler message:

```
',' expected import myClass
```

This message means that the Java compiler expects to find a package and class name and cannot parse a line that contains only a class name.

For details, see the Sun Microsystems compatibility document supplied with the JDK.

Navigating the OC4J Documentation Set

Most of the location of J2EE subject matter is obvious. For example, you can find out how to implement and use servlets within the *Oracle Application Server Containers for J2EE Servlet Developer's Guide*. Table 1–2 shows each J2EE subject matter and where you can find the information in the OC4J documentation set.

Table 1–2 Location of Information for J2EE Subjects

J2EE Subject	The Subject is Documented in this OC4J Documentation Book
JSP	<i>Oracle Application Server Containers for J2EE Support for JavaServer Pages Developer's Guide</i>
JSP Tag Libraries	<i>Oracle Application Server Containers for J2EE JSP Tag Libraries and Utilities Reference</i>
Servlet	<i>Oracle Application Server Containers for J2EE Servlet Developer's Guide</i>
EJB	<i>Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide</i>
JTA	<i>Oracle Application Server Containers for J2EE Services Guide</i>
Data Source	<i>Oracle Application Server Containers for J2EE Services Guide</i>
JNDI	<i>Oracle Application Server Containers for J2EE Services Guide</i>
JMS	<i>Oracle Application Server Containers for J2EE Services Guide</i>
RMI and RMI/IIOP	<i>Oracle Application Server Containers for J2EE Services Guide</i>
Security	<i>Oracle Application Server Containers for J2EE Security Guide</i>
CSiV2	<i>Oracle Application Server Containers for J2EE Security Guide</i>
JCA	<i>Oracle Application Server Containers for J2EE Services Guide</i>
Java Object Cache	<i>Oracle Application Server Containers for J2EE Services Guide</i>
Web Services	<i>Oracle Application Server Web Services Developer's Guide</i>
HTTPS	<i>Oracle Application Server Containers for J2EE Services Guide</i>

OC4J Installation

OC4J is a lightweight container that is J2EE-compliant. It is configured with powerful and practical defaults and is ready to execute after installation. OC4J is installed with Oracle Application Server; therefore, see the *Oracle Application Server Installation Guide* for details on OC4J installation.

Using OC4J in an Enterprise or Standalone Environment

OC4J is installed within Oracle Application Server with the goal of managing J2EE enterprise systems. Oracle Application Server can manage multiple clustered OC4J processes. Oracle Application Server, which includes OC4J, is managed and configured through the Oracle Enterprise Manager 10g, which can manage and configure your OC4J processes across multiple application server instances and hosts. Thus, you cannot locally manage your OC4J process using the `admin.jar` tool or by hand editing a single OC4J process' configuration files. This undermines the enterprise management provided by the Oracle Enterprise Manager 10g.

You can still execute OC4J as you have in the past. For those who want a single OC4J instance for development environments or simple business needs, you can download OC4J in standalone mode—complete with documentation.

The following sections discuss both management options in the following sections:

- [Managing Multiple OC4J Instances in an Enterprise Environment](#)

- [Managing a Single OC4J Instance](#)
- [Using JDK With OC4J](#)

Also, the following section describes how to understand the OC4J documentation set:

- [OC4J Documentation Set Assumptions](#)

Managing Multiple OC4J Instances in an Enterprise Environment

You manage Oracle Application Server, including OC4J, using Oracle Enterprise Manager 10g within an enterprise system. This includes clustering, high availability, load balancing, and failover.

You configure each OC4J instance and its properties—within the context of an application server instance—using Oracle Enterprise Manager 10g. After configuration, you start, manage, and control all OC4J instances through Oracle Enterprise Manager 10g. You can group several OC4J processes in a cluster. You must use either the Oracle Enterprise Manager 10g management tool or its command-line tools for starting, stopping, restarting, configuring, and deploying applications.

Note: You cannot use the OC4J standalone tool—`admin.jar`—for managing OC4J instances created in an application server instance.

You can modify the XML files locally. If you do so, you must notify Oracle Enterprise Manager 10g that these files have been hand edited through the Distributed Configuration Management (DCM) component tool—`dcmctl`. The following is the command that you execute after hand editing an XML file:

```
dcmctl updateconfig -ct oc4j
```

DCM controls and manages configuration for Oracle Application Server instances and its Oracle HTTP Server and OC4J components. For more information on DCM, see the Distributed Configuration Management Administrator's Guide.

This book discusses how to start, stop, manage, and configure OC4J in an enterprise environment.

Managing a Single OC4J Instance

You can still use a single OC4J—outside of the Oracle Application Server environment. After downloading OC4J in `oc4j_extended.zip` from OTN, you can start, manage, and control all OC4J instances through `oc4j.jar` and the `admin.jar` command-line tool. You configure either through the `admin.jar` command or by modifying the XML files by hand.

Any standalone OC4J process is not managed by Oracle Enterprise Manager 10g and cannot be used within an Oracle Application Server enterprise environment. Typically, you would use standalone for development or for a simple single OC4J instance Web solution.

Download the *OC4J Standalone User's Guide* for information on how to start, stop, configure, and manage your standalone process.

Using JDK With OC4J

You must have only one Java Developer's Kit (JDK) installed on your system. Make sure that it is a version that OC4J supports:

- JDK 1.3.1
- JDK 1.4.1
- JDK 1.4.2

OC4J Standalone does not include a JDK. If you are using OC4J Standalone, then you must provide your own JDK installation.

Some suppliers bundle a JDK with their products. If necessary, remove any old version(s), replace them with a supported version, and update the appropriate environment variables. Make sure that your `PATH`, `CLASSPATH`, and `LD_LIBRARY_PATH` (or `LIB` on Windows) variables are all set to the officially supported version of JDK.

Important: Due to a logging implementation dependency issue, OC4J fails to start when using JDK 1.3. The solution to this problem is to remove or comment out the following entry in the `ORACLE_HOME/j2ee/home/config/server.xml` configuration file:

```
<j2ee-logging-config path="./j2ee-logging.xml" />
```

OC4J Documentation Set Assumptions

Aside from this book, the rest of the OC4J documentation set was written with a standalone mindset. These other books may refer to modifying XML files by hand for managing the instance. This book provides a good overview and familiarization of the Oracle Enterprise Manager 10g configuration pages. It also guides you to understand the relationship of each Oracle Enterprise Manager 10g page to its XML counterpart. Use the familiarity of the Oracle Enterprise Manager 10g when reading the other OC4J books. You should be able to look at an XML representation and match it to the relevant Oracle Enterprise Manager 10g field names.

Also, the Distributed Configuration Management (DCM) utility, `dcmctl`, provides a command-line alternative to using Oracle Enterprise Manager 10g for some management tasks. The `dcmctl` tool uses the same distributed architecture and synchronization features as Oracle Enterprise Manager 10g, thereby providing identical functionality in a format that is ideal for scripting and automation.

The following functions can be managed through DCM:

- administration
- managing application server instances
- managing components
- managing clusters
- deploying applications

For other DCM commands that relate to OC4J, see the Distributed Configuration Management Administrator's Guide.

OC4J Communication

For HTTP applications, OC4J is preconfigured to execute behind the Oracle HTTP Server (OHS). You use the Oracle HTTP Server as a front-end listener and OC4J as the back-end J2EE application server.

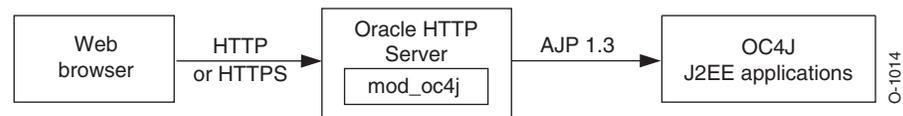
However, for RMI-based applications—such as EJB and JMS—clients should send their requests directly to OC4J. See "[Understanding and Configuring OC4J Listeners](#)" on page 3-20 for directions.

HTTP Communication

For all incoming HTTP communication within the application server environment, you use the OHS as a front-end listener and OC4J as the back-end J2EE application server. [Figure 1-1](#) illustrates this as follows:

1. A browser accesses the OHS listener for all HTTP requests. The Oracle HTTP Server is an Apache server. The default port number is 7777.
2. OHS, through the `mod_oc4j` module, passes the request to the OC4J server. The connection between the OHS and OC4J uses the Apache JServ Protocol (AJP) on a port number negotiated during OC4J startup. AJP is faster than HTTP, through the use of binary formats and efficient processing of message headers.

Figure 1-1 HTTP Application Listener



The `mod_oc4j` module is preconfigured to direct all incoming HTTP requests under the `j2ee/` Web context to OC4J. Thus, if you want to use the default routing, you can deploy your Web application into a servlet context that includes as its prefix `j2ee/`. However, any URL mapping you provide in the deployment wizard is automatically added to the `mod_oc4j` module. See "[Configuring Oracle HTTP Server With Another Web Context](#)" on page 3-22 for information on what is added to `mod_oc4j` for you during deployment. For additional information on the `mod_oc4j` module, see the *Oracle HTTP Server Administrator's Guide*.

Notes: In Oracle9iAS version 1.0.2.2, the default OC4J Web site did not use the Oracle HTTP Server as a front-end, and it listened using the HTTP protocol on port 8888.

Requirements

For optimum performance, run OC4J with the JDK that is installed with Oracle Application Server Release 2, which is JDK 1.4.2.

It is not necessary to add anything to your `CLASSPATH` to run OC4J, because it loads the Java JAR and class files directly from the installation directory, from the `lib/` subdirectory, and from the deployed application EAR files.

Configuration and Deployment

This chapter demonstrates how to configure and execute OC4J as simply and quickly as possible. You installed OC4J with the Oracle Application Server installation.

Within OC4J, you can execute servlets, JSP pages and EJBs. As an example of deploying an application to OC4J, this chapter describes how to configure the FAQ application demo.

This chapter includes the following topics:

- [OC4J Home Page Overview](#)
- [Starting and Stopping OC4J](#)
- [Creating the Development Directory](#)
- [Configuring the FAQ Application Demo](#)
- [Deploying Applications](#)
- [What Happens When You Deploy?](#)
- [Recovering From Deployment Errors](#)
- [Undeploying Web Applications](#)

OC4J Home Page Overview

Most of the configuration and management of your OC4J instance occurs off its OC4J Home Page. When you create an OC4J instance off of the Oracle Application Server Instance Home Page, it creates an OC4J Home Page for configuration and management of your OC4J instance. Each OC4J instance has its own OC4J Home Page.

Off the Oracle*9i*AS Console, you can drill down to any of the running OC4J instances by selecting the name of the instance (home, for example) in the System Components table. The Oracle*9i*AS Console displays the OC4J Home Page for that instance.

[Figure 2–1](#) shows portions of the OC4J Home Page for the home instance.

Figure 2-1 Oracle Application Server Console OC4J Home Page

OC4J: home

[Home](#) [Applications](#) [Administration](#)

General

 Status **Up** [Stop](#) [Restart](#)
 Start Time **Jul 2, 2003 6:47:35 PM**
 Virtual Machines **1**

Status

CPU Usage (%) **0.13**
 Memory Usage (MB) **67.98**
 Heap Usage (MB) **26.04**

JDBC Usage

Open JDBC Connections **0**
 Total JDBC Connections **0**
 Active Transactions **0**
 Transaction Commits **0**
 Transaction Rollbacks **0**

Response - Servlets and JSPs

Active Sessions **0**
 Active Requests **1**
 Request Processing Time (seconds) **0.005**
 Requests per Second **0.16**

Response - EJBs

Active EJB Methods **0**
 Method Execution Time (seconds) **0.00**
 Method Execution Rate (per second) **0.00**

[Home](#) [Applications](#) [Administration](#)

The OC4J Home Page shows metrics on your OC4J instance and its applications. In addition, you can start, stop, and restart all OC4J processes configured to this instance.

From the OC4J Home Page, you can navigate to the following pages:

- Click **Applications** to access the Oracle9iAS Console Applications Page. See "[Applications Page](#)" on page 2-2 for more information.
- Click **Administration** to access the Oracle9iAS Console Administration Page. See "[Administration Page](#)" on page 2-3 for more information.

Applications Page

Figure 2-2 shows the Deployed Applications section. In this section, you can deploy applications using the **Deploy EAR file** or **Deploy WAR file** buttons. After deployment, you can modify configuration for each application. See "[Deploying Applications](#)" on page 2-11 for more information.

Figure 2–2 Deployed Applications

OC4J: home

Home Applications Administration

Page Refreshed Jul 3, 2003 3:03:51 PM

Default Application Name [default](#)
 Default Application Path [application.xml](#)

Deployed Applications

Deploy EAR file Deploy WAR file

Edit Undeploy Redeploy

Select	Name	Path	Parent Application	Active Requests	Request Processing Time (seconds)	Active EJB Methods
<input checked="" type="radio"/>	FAQAPP_SB	../applications/FAQAPP_SB.ear	default	0	0.00	0

Home Applications Administration

As an example, you can see how the FAQ application demo is configured and deployed to OC4J in "Configuring the FAQ Application Demo" on page 2-6.

Administration Page

Figure 2–3 shows the Administration page. You can modify the following:

- Instance Properties, which are global configuration values for a specific OC4J instance. This includes configuration of OC4J services, such as RMI, JMS, and Web sites.
- Application defaults, including default data sources and security that can be used by all deployed applications in this OC4J instance.

Figure 2–3 Administration Section

OC4J: home

Home Applications Administration

Page Refreshed Jul 3, 2003 3:02:48 PM

Instance Properties

[Server Properties](#)
[Website Properties](#)
[JSP Container Properties](#)
[Replication Properties](#)
[Advanced Properties](#)

Application Defaults

[Data Sources](#)
[Security](#)
[JMS Providers](#)
[Global Web Module](#)

Home Applications Administration

OC4J Inheritance

OC4J applications have a hierarchical parent-child relationship to facilitate administration through inheritance. A child application inherits certain attributes from its parent application such as principals and JNDI objects including data sources, JMS providers and EJBs. When an OC4J application is deployed, you specify the parent application. The Default Application is the top of the parent hierarchy.

Details for each of these options are covered in "Configuring OC4J Using Enterprise Manager" on page 3-1.

Starting and Stopping OC4J

OC4J is installed with a default configuration that includes a default Web site and a default application. Therefore, you can start OC4J immediately without any additional configuration.

Important: Due to a logging implementation dependency issue, OC4J fails to start when using JDK 1.3. The solution to this problem is to remove or comment out the following entry in the `ORACLE_HOME/j2ee/home/config/server.xml` configuration file:

```
<j2ee-logging-config path="./j2ee-logging.xml" />
```

From the Oracle Enterprise Manager Web site, you can start, stop, and restart OC4J with one of the following methods:

- Drill down to the Oracle Application Server Instance Home Page, start the entire Oracle Application Server instance, which includes any configured OC4J instances, by clicking the **Start All** button in the General section. In addition, **Stop All** and **Restart All** buttons are included for these purposes.
- Drill down to the Oracle Application Server Instance Home Page, start a specific OC4J instance by selecting the radio button next to the OC4J instance. Click the **Start** button. Click **Stop**, **Restart**, or **Delete** to stop, restart, or delete the specified OC4J instance.
- From the Oracle Application Server Instance Home Page, drill down to the OC4J Home Page. Click the **Start** button in the General section on this page. In addition, **Stop** and **Restart** buttons are included for these purposes. [Figure 2-1](#) displays the General section of the OC4J Home Page.

OC4J automatically detects changes made to deployed applications and reloads these applications automatically. Therefore, you do not need to restart the server when redeploying an application. However, you may have to restart OC4J if you modify fields in any of the options off of the Administration page.

You can also start, stop, and restart using the DCM control command. See the *Distributed Configuration Management Administrator's Guide* for directions.

Testing the Default Configuration

Start OC4J with the defaults, as follows:

1. From the Oracle Application Server Instance Page, start either the whole Oracle Application Server instance or—at least—the Oracle HTTP Server and OC4J components. To start, click the **Start All** button for the Oracle Application Server instance or select the components and click the **Start** button.

2. Test OC4J by specifying the following from a Web browser:

```
http://<ohs_host>:7777/j2ee/j2ee-index.html
```

Substitute the name of the host where the OHS is installed for `<ohs_host>`.

3. Test a servlet deployed in OC4J during installation by specifying the following in a Web browser:

```
http://<ohs_host>:7777/j2ee/servlet/HelloWorldServlet
```

This command returns a "Hello World" page. The `HelloWorldServlet` is automatically deployed with the OC4J installation.

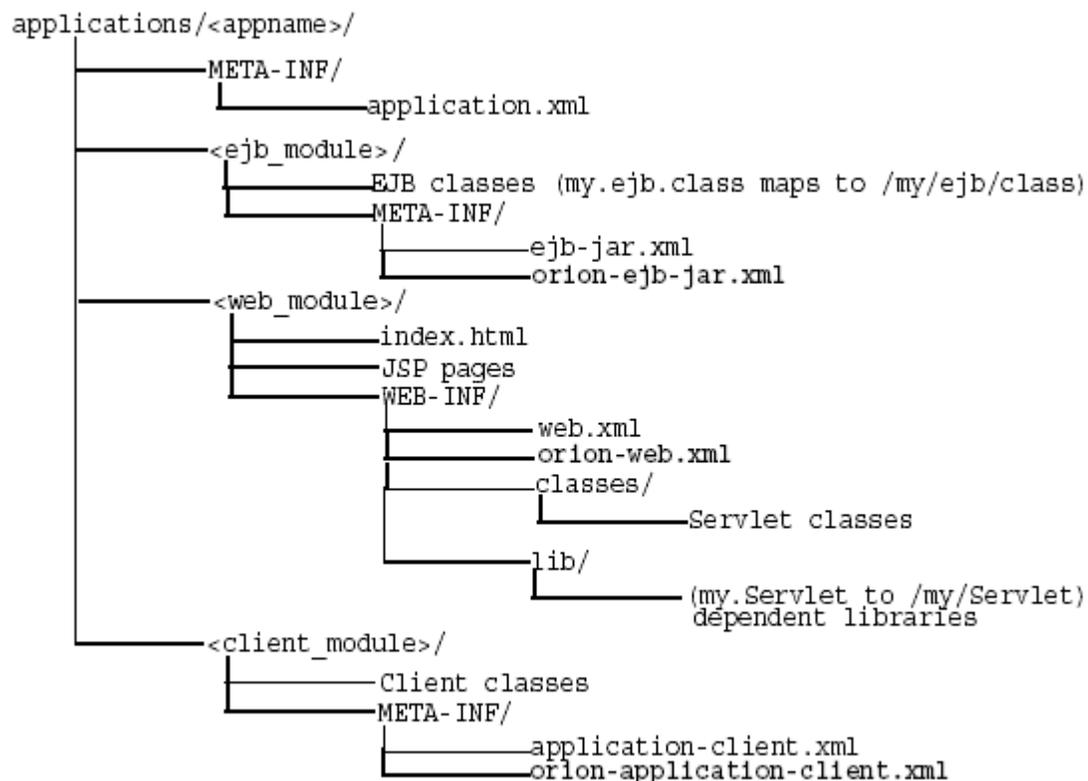
Note: The examples and URLs in this guide use port 7777, which is the default port for the OHS Web listener. If you change the default port number of the OHS, then specify the new port number after the hostname, as follows:

`http://<ohs_host>:<ohs_port>/j2ee/`

Creating the Development Directory

When developing your application, Oracle recommends that you use consistent and meaningful naming conventions. As an example, you could develop your application as modules within a directory named after your application. All the subdirectories under this directory could be consistent with the structure for creating JAR, WAR, and EAR archives. Thus, when you have to archive the source, it is already in the required archive format. [Figure 2-4](#) demonstrates this structure.

Figure 2-4 Development Application Directory Structure



Consider the following points regarding [Figure 2-4](#):

- You cannot change the following directory names and XML filenames: META-INF, WEB-INF, application.xml, ejb-jar.xml, web.xml, and application-client.xml.
- Separate directories clearly distinguish modules of the enterprise Java application from each other. The application.xml file, which acts as the standard J2EE application descriptor file, defines these modules.

- The directories containing the separate modules (`<ejb_module>`, `<web_module>`, and `<client_module>`) can have arbitrary names. However, these names must match the values in the standard J2EE application descriptor file—the local `application.xml` file.
- The top of the module represents the start of a search path for classes. As a result, classes belonging to packages are expected to be located in a nested directory structure beneath this point. For example, a reference to an EJB package class `'myapp.ejb.Demo'` is expected to be located in `<appname>/<ejb_module>/myapp/ejb/Demo.class`.

Configuring the FAQ Application Demo

This section describes how to configure the FAQ J2EE demo application, which provides support for managing Frequently Asked Questions (FAQs) and storing/retrieving these FAQs from an Oracle database. You must have a working Oracle database and an OC4J installation. You should use this installation for demonstration purposes only and not in a production environment.

FAQs are broadly categorized into *Specialization Areas*. Each Specialization Area is further sub-categorized into *Topics*. Each FAQ can be associated with multiple Specialization Areas, where each area has one or more Topics associated with them.

You can generate a list of FAQs (in HTML format) for a given Specialization Area for internal or external publication.

- Internal: FAQs that are published for internal users only. These include all external and internal FAQs.
- External: FAQs that are published on external forums.

Within the demo, Areas, Topics, and FAQs are entered or updated in the database through Input/Update screens or through a Web service interface. Each Area, Topic and FAQ is uniquely identified by a primary key, which is automatically generated by the system.

This application is a J2EE 1.3 compliant application, developed utilizing the following technologies:

- HTML (including MS-HTML for creating a Rich-Text Editor)
- JavaScript
- Cascade Style Sheets
- Java Server Pages 1.2
- Servlet 2.3
- JSP Standard Tag Library (JSTL) 1.0
- Oracle JSP 1.2 Utility Tag Libraries
- Enterprise JavaBeans 2.0 (using Local Interfaces, Abstract Classes, CMR and EJB-QL)
 - Entity Bean (CMP)
 - Session (Facade) Bean (stateless)
- Oracle Application Server Java Authentication and Authorization Service (JAAS) Provider
- Oracle Application Server Web Services

The following sections detail how to configure and deploy the FAQ demo application. In addition, the last section demonstrates how these steps relate to any application that you may wish to configure and deploy to OC4J:

- [Environment Setup for FAQ Demo](#)
- [OC4J System Configuration for FAQ Demo](#)
- [Deploy the FAQ Demo](#)
- [Deployment Details Explained](#)

Environment Setup for FAQ Demo

In order to run the FAQ demo, you must modify the back-end database to contain tables that the demo uses.

Oracle Database

1. Add the `faq` user with password of `faq` in your database.
2. Create the database tables for the FAQ demo by executing the SQL table creation script `CreateTables.sql`, which is located at `<FAQApp_home>/faq/sql/CreateTables.sql` or can be downloaded with the rest of the FAQ application from OTN at <http://www.oracle.com/technology/tech/java/oc4j/demos/> in the `FAQApp.zip` file.

In an Oracle database environment, you can execute the SQL script through SQL*Plus, connecting to the database and schema where you want the tables to be installed and executing `@CreateTables`. Please refer to the Oracle database documentation for further instructions on how to use SQL*Plus, running install scripts, creating database users/schemas, and so on.

OC4J System Configuration for FAQ Demo

In order for the FAQ demo to execute properly, the following system modification must be implemented:

- Modify the default data source, `OracleDS`, to point to the back-end database.
- Add the FAQ user to the `jazn.com` realm and assign it to the `users` role.

The directions for each of these steps are covered in the following sections:

- [Data Source Configuration](#)
- [Security Configuration](#)

Data Source Configuration

In order to execute the FAQ application, you must have an Oracle database with the corresponding FAQ application database schema installed on it. The FAQ Application uses the default global data source named `OracleDS` that ships with the application server, which must be configured so that it connects to the database in which you created the FAQ tables.

Note: An I/O exception is thrown if you do not update the global `OracleDS` data source appropriately.

1. Navigate to the OC4J Home Page on the Oracle Enterprise Manager Web site.
2. Select the Administration tab at the top of the page.
3. Select **Data Sources** under the Application Defaults section. The default application is the automatic parent of each application and it holds global configuration for all deployed applications, such as the data sources used. You are going to modify the default data source that the FAQ application uses.
4. For the OracleDS data source, click the **Edit** button. This brings up the configuration information for this data source. Modify the JDBC URL, driver, username, and password to point to your back-end database.

When finished, click the **Apply** button.

If your back-end database uses the thin JDBC driver, is located at myhost:1521:ORCL, and uses the username/password of faq/faq, then the j2ee/home/config/data-sources.xml file is modified to point to the database service at the URL jdbc:oracle:thin:@myhost:1521/MYSERVICE, as follows:

```
<data-source
  class="com.evermind.sql.DriverManagerDataSource"
  name="OracleDS"
  location="jdbc/OracleCoreDS"
  xa-location="jdbc/xa/OracleXADS"
  ejb-location="jdbc/OracleDS"
  connection-driver="oracle.jdbc.driver.OracleDriver"
  username="faq"
  password="faq"
  url="jdbc:oracle:thin:@myhost:1521/MYSERVICE"
  inactivity-timeout="30" />
```

Security Configuration

The FAQ demo uses Oracle's JAAS Provider, JAZN, for authentication and user access control capabilities.

1. Select the Administration tab at the top of the OC4J Home Page.
2. Select **Security** under the Application Defaults section. The default configures global configuration for all deployed applications. You are going to add the user that the FAQ application uses to the jazn.com realm.
3. On the Security page, scroll down to the Users section
4. Click **Add User**. A configuration screen appears that allows you to add information about the new user. Supply the following information:
 - Name and password of the user
 - Check the box next to the jazn.com/users realm

When finished, click the **Apply** button.

Alternatively, an application user is added to the default jazn.com realm through the jazn.jar command line tool, as follows:

```
> java -jar jazn.jar -adduser jazn.com <username> <passwd>
> java -jar jazn.jar -grantrrole users jazn.com <username>
```

The previous adds your user (given the username and password) to the jazn.com realm and then grants the users role to the new user. See the Oracle Application Server

Containers for J2EE Security Guide for complete information on using JAZN as your security provider.

Deploy the FAQ Demo

Download the FAQ demo application from OTN at <http://www.oracle.com/technology/tech/java/oc4j/demos/index.html> in the `FAQApp.zip` file.

1. Unzip this file to a working directory, which is referred to as `<FAQApp_Home>`.
2. Navigate to the OC4J Home Page on the Oracle Enterprise Manager Web site.
3. Select the Applications tab at the top of the screen.
4. Click on the **Deploy EAR File** button. This starts the application deployment wizard.
5. Provide the EAR file and the name of your application in the Select Application page. Click the **Browse** button to find the `FAQApp.ear` file that you unzipped on your system. Type "FAQApp" in the application name field. Click the **Continue** button.
6. Provide the URL mappings for the servlet context on all Web modules in the FAQ application. The FAQApp demo contains a Web module, which should be mapped to the `/FAQApp` servlet context. Type `/FAQApp` in the URL mapping field and click the **Next** button.
7. At this point, the FAQApp demo does not need any additional configuration through the wizard. You can jump to the Summary page by clicking **Finish**.
8. Read the summary of the FAQApp application deployment. Click the **Deploy** button to complete the application deployment.
9. On the OC4J Home Page, select "FAQApp" in the Name column of the Applications section. This shows the configuration and all deployed modules of the FAQApp demo application. If the OC4J server is started, the application is automatically started.
10. Execute the FAQApp application in your browser by accessing the OHS, where the default port is 7777.

`http://<ohs_host>:7777/FAQApp`

The FAQApp screen appears.

Deployment Details Explained

Although the development of J2EE applications is standardized and portable, the non-application (server) configuration is not. The necessary server configuration depends on the services that your application uses. For example, if your application uses a database, you must configure its `DataSource` object.

For basic applications, such as the FAQ demo, you configure the following:

- `META-INF/application.xml`—The standard J2EE application descriptor for the application is contained within the `application.xml` file. This file must be properly configured and included within the J2EE EAR file that is to be deployed.
- `DataSource` objects—You must configure the `DataSource` object in the Data Source configuration page for each database used within the application.

- When you deploy any application, you configure the name of the application, the Web contexts, and any application-level data sources or security, as well as other services.

To create and deploy simple J2EE applications, perform the following basic steps:

Basic Step	FAQ Application Step Description
1. Create or obtain the application.	Download the <code>FAQApp.zip</code> from OTN
2. Make any necessary server environment changes.	Set the <code>JAVA_HOME</code> variable
3. Modify any global configuration modifications.	Modified the default data source, <code>OracleDS</code> , and added a user to the default JAZN security.
4. Provide the application deployment descriptors.	The deployment descriptors, such as <code>web.xml</code> and <code>ejb-jar.xml</code> , are provided in the <code>FAQApp.ear</code> file.
5. Update the application standard J2EE application descriptor file.	The <code>application.xml</code> file is included in the <code>FAQApp.ear</code> file.
6. Build an EAR file including the application—if one does not already exist.	If you want to modify the FAQ demo, modify within the <code>src</code> directory, and use ANT to build an EAR file.
7. Register the application in the appropriate server XML files.	The application is registered when you deploy the application through the deployment wizard.

The following steps describe what modifications to make to deploy the FAQ demo application into OC4J.

Note: Displays of the screens for each step of the deployment wizard are shown in "[Deploying Applications](#)" on page 2-11.

1. We asked you to download the FAQ demo application from the Oracle OTN site.
2. Make any necessary server environment changes. You must set the `JAVA_HOME` variable to the base directory of the Java 2 SDK.
3. Modify the global configuration:
 - a. Configure the OC4J `DataSource` for an Oracle database. Modify the default data source, `OracleDS`, to point to your back-end database, with the correct URL, username, and password.
 - b. Add the OracleAS JAAS Provider user to the `jazn.com` realm.
4. All of the application XML files, such as `web.xml`, are provided for you in the ZIP file, correctly configured.
5. Update the standard J2EE application descriptor file. The `application.xml` in the FAQ demo application is provided for you in the ZIP file. OC4J uses the `application.xml` file as the standard J2EE application descriptor file.
6. Build an EAR file including the application. You can modify the FAQ demo application and rebuild it using the ANT command. To rebuild and deploy the FAQ demo, execute the following:

```
ant deploy
```

The `ANT build.xml` is included in the FAQ ZIP download. To learn more about the ANT file, go to the following Jakarta site:

<http://jakarta.apache.org/ant/>

If you do not want to rebuild, you can copy the `FAQApp.ear` from the ZIP file into `j2ee/home/applications`. This step places the FAQ application in the OC4J server.

1. Register the J2EE application and its Web application within the deployment wizard. When you deploy, the wizard asks for information that is necessary to register the application, such as the name of the application and the Web context.
2. Start OC4J. For a complete description of all the OC4J starting options, see ["Starting and Stopping OC4J"](#) on page 2-3.
3. Open your Web browser and then specify the following URL:

`http://oc4j_host:8888/FAQApp`

Deploying Applications

This section describes how to deploy a J2EE application to the OC4J server. When you deploy an application using the deployment wizard, the application is deployed to the OC4J instance and any Web application is bound to a URL context so that you can access the application from OC4J.

This section includes the following topics:

- [Basic Deployment](#)
- [Impact of Undeploying/Redeploying an Application](#)
- [Impact of "Hot Deploying" an Application](#)

To deploy your application, drill down to the OC4J Home Page and scroll to the Deployed Applications section. [Figure 2-2](#) shows this section.

Note: You can also deploy simple applications with the `dcmctl` command. See the *Distributed Configuration Management Administrator's Guide* for directions.

Basic Deployment

Your J2EE application can contain the following modules:

- Web applications
The Web applications module (WAR files) includes servlets and JSP pages.
- EJB applications
The EJB applications module (EJB JAR files) includes Enterprise JavaBeans (EJBs).
- Client application contained within a JAR file

Archive the JAR and WAR files that belong to an enterprise Java application into an EAR file for deployment to OC4J. The J2EE specifications define the layout for an EAR file.

The internal layout of an EAR file should be as follows:

Archive Directory Format

Archive these files using the JAR command in the <appname> directory, as follows:

```
% jar cvfM <appname>.ear .
```

Note that the `application.xml` file acts as a standard J2EE application descriptor file.

- To deploy a J2EE application packaged within an EAR file, click the **Deploy Ear File** button on the Applications page.
- To deploy a J2EE Web application packaged within a WAR file, click the **Deploy WAR File** button on the Applications page.

Both of these buttons start an eight-step application deployment wizard, which guides you through deploying an application. In the case of the WAR file, the `application.xml` file is created for the Web application. Whereas, you must create the `application.xml` file within the EAR file. Thus, deploying a WAR file is an easier method for deploying a Web application.

Note: You must still provide configuration for J2EE services, such as data source and security configuration.

Select Application

Figure 2–5 shows the first page, which enables you to provide the following:

- Browse your system for the EAR file to be deployed.
- Provide a name to be identified with this application. The application name is user-created and will be the identifier for the application in the Applications page.
- Designate a parent application. A child application can see the namespace of its parent application. Thus, setting up an application as a parent is used to share services among children. The default parent is the global application. The parent application must already be deployed to be seen in the pull-down menu.

Figure 2–5 Designate EAR File

Deploy Application

For a J2EE application to be successfully deployed on the OC4J container, the application has to be assembled correctly as an Enterprise Archive (ear) file, with all the needed application and module deployment descriptors. The OC4J container generates default OC4J specific deployment descriptors when the application is deployed. If you have custom OC4J specific deployment descriptors that you wish to use, you need to include these in the ear file.

Select the J2EE application (.ear file) to be deployed.

J2EE Application

Specify a unique application name for this application.

Application Name

Select the parent for this application.

Parent Application

When the application is deployed, the information in this step enables the following:

1. Copies the EAR file to the `/applications` directory.
2. Creates a new entry in `server.xml` for the application, as follows:

```
<application name=<app_name> parent="applicationWithCommonClasses" path=<path_
EARfile> auto-start="true" />
```

where

- The `name` variable is the name of the application you provided.
- The `parent` is the name of the optional parent application. The default is the global application. Children see the namespace of its parent application. This setting is used to share services, such as EJBs among multiple applications.
- The `path` indicates the directory and filename where the EAR file is deployed.
- The `auto-start` variable indicates if this application should be automatically restarted each time OC4J is restarted.

For a description of the elements in `server.xml`, see ["Elements in the server.xml File"](#) on page B-6.

Click the **Continue** button to go to the next step in the wizard deployment process. The wizard uploads your EAR file and examines the application. Based on your configuration files and deployment descriptors, the wizard dynamically presents only the configuration screens your application needs. These screens are a subset of the steps presented in the following sections:

- [Provide The URL Mappings For All Web Modules](#)
- [IIOP Stub Generation](#)
- [Provide Any Resource Reference Mappings](#)
- [Specify Any User Manager](#)
- [Provide Any Security Role Mappings](#)
- [Publish Web Services](#)
- [Deployment Review](#)

Provide The URL Mappings For All Web Modules

Map any Web modules in your application to a specific URL for its servlet context. All OC4J servlet contexts must be prefixed with a slash `/`. When you try to access any Web applications, you provide the host, port, and Web context.

For all Web modules, your URL mapping for this module includes the URL you bind in this screen. Thus, for the URL `http://<host>:<port>/url_name`, provide `/url_name` in the URL mapping screen of the wizard.

Figure 2–6 URL Mapping

Deploy Application: URL Mapping for Web Modules

A web module needs to be mapped to an URL pattern in the default web site before it can be accessed. The following table lists all the web modules found in your application. Specify the URL mapping for each of these modules.

Name	URL Binding
WebTier	* <input type="text"/>

Click the **Next** button to go to the next step in the wizard deployment process.

IIOB Stub Generation

The EJBs in your application can have IIOB stubs generated by selecting **Generate IIOB Stubs**. See the "Interoperability and RMI Tunneling" chapter in the *Oracle Application Server Containers for J2EE Services Guide* for information on IIOB stubs.

Figure 2–7 IIOB Stub Generation

Deploy Application: IIOB Stub Generation

This application contains EJB's. Please confirm if you wish to generate IIOB stubs.

Generate IIOB Stubs

Click the **Next** button to go to the next step in the wizard deployment process.

Provide Any Resource Reference Mappings

Map any references resources in your application, such as data sources or mail queues, to physical entities currently present on the OC4J container. Note that if you need a specific resource, you must have already added this to the OC4J container before you deploy your application in order for you to match them in this step.

For most applications, the resource reference you must designate is the data source JNDI name. This screen does not configure the data source information, it only designates an already configured data source or a data source that you will be configuring later. Designate the JNDI location name of the data source that the application will be using.

Figure 2–8 Resource Reference Mappings

Deploy Application: Resource Reference Mappings

The table below lists all resource references found in your application. Resource references need to be associated with the JNDI names of physical entities on the system where the selected instance/cluster is running.

Resource Reference	Type	Referenced By	Authentication	JNDI Location
jdbc/EstoreDataSource	javax.sql.DataSource	Web Module: WebTier	Container	<input type="text"/>
jdbc/EstoreDataSource	javax.sql.DataSource	EJB: TheProfileMgr	Container	<input type="text"/>
jdbc/InventoryDataSource	javax.sql.DataSource	EJB: TheInventory	Container	<input type="text"/>
jdbc/SignOnDataSource	javax.sql.DataSource	EJB: TheSignOn	Container	<input type="text"/>
jdbc/EstoreDataSource	javax.sql.DataSource	EJB: TheAccount	Container	<input type="text"/>
jdbc/EstoreDataSource	javax.sql.DataSource	EJB: TheOrder	Container	<input type="text"/>
mail/MailSession	javax.mail.Session	EJB: TheMailer	Container	<input type="text"/>
jdbc/EstoreDataSource	javax.sql.DataSource	EJB: TheCatalog	Container	<input type="text"/>

If you have any MDBs in your EAR file, you may be required to add information about the subscriptions or topics. If you are defining DataSource objects for CMP entity beans, you are given the option to add a JNDI location for those DataSource objects.

Figure 2–9 Resource Reference Mappings for CMP Entity Beans

Deploy Application: Resource Reference Mappings

The table below lists all resource references found in your application. Resource references need to be associated with the JNDI names of physical entities on the system where the selected instance/cluster is running.

Resource Reference	Type	Referenced By	Authentication	JNDI Location
jms/logTopic	javax.jms.Topic	Web Module:	Container	<input type="text"/>
jms/logTopicConnectionFactory	javax.jms.TopicConnectionFactory	Web Module:	Container	<input type="text"/>

Data Sources for CMP Entity Beans

Your application includes CMP entity beans. CMP entity beans require a Data Source be associated with them to deal with persistence. If the table associated with this entity bean does not exist, OC4J will create one on deployment.

Entity Bean	EJB Module	Data Source
com.evermind.logger.LogMessage		jdbc/OracleDS
com.evermind.ejb.Counter		<input type="text"/>

Click the **Next** button to go to the next step in the wizard deployment process.

Specify Any User Manager

You can specify what User Manager to use for security. For complete security, we recommend that you choose the JAZN XML User Manager.

Figure 2–10 *User Manager Choices*

Deploy Application: User Manager

Specify a user manager to be associated with the application. Note that all web modules in your application will be automatically SSO enabled, when you use JAZN LDAP as your user manager.

Use JAZN XML User Manager

Default Realm

XML Data File

Use XML User Manager

Path to principals file

Use Custom User Manager

Name

Class Name

Description

Initialization Parameters for Class

Select	Name	Value
<input type="checkbox"/>	No initialization parameters	

As [Figure 2–10](#) demonstrates, you must already have your User Manager set up and configured. Most of the entries requires an XML file that designates the security roles, users, and groups for your security mappings.

- JAZN XML User Manager—This is the recommended User Manager. It requires a default realm and a `jazn-data.xml` file.
- XML User Manager—This is not the most secure option. It requires a `principals.xml` file.
- Custom User Manager—This User Manager must be programmed; provide the class name in this field.

For more information on security and User Managers, see the Oracle Application Server Containers for J2EE Security Guide.

Provide Any Security Role Mappings

Map any security roles defined in your application to existing users and groups. If you have defined a security role within your application, you can map this role to a security group or role. You do not define security groups and users in this screen. Users and groups are obtained from your user manager.

Figure 2–11 Security Role Mappings

Deploy Application: Security Role Mappings

Your application exposes the following security roles. You may assign these roles to users/groups present on the OC4J container. To do this, select a role and then click on the Map Role button. You will be directed to a new page where you can map this role to users/groups. Click on OK in that page to get back to this screen and map another role.

Select	Name	Description	Assigned Users	Assigned Groups
	No security roles found in this application			

Click the **Next** button to go to the next step in the wizard deployment process.

For more information on security roles, see the *Oracle Application Server Security Guide* and the *Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide*.

Publish Web Services

Publish any Web services defined in your application. This feature requires the UDDI registry. Web services are not installed with a core install.

If you have defined any Web services, they are shown in the following screen:

Figure 2–12 Web Services

Deploy Application: Publish Web Services

The table below lists all of the web services found in your application. Each web service that you wish to access must be published to the UDDI registry in an appropriate category. To do this, select a web service and then click on the Publish button. You will be directed to a new page where you can enter details and select the category. Click on OK in that page to get back to this screen and publish another web service.

Web Services

Select	Web Service	Web Module	Status
	No Web Services found		

⊖ Previous Next ⊕

If you want to publish these Web services, then click on the Publish button. This leads you through the process of publishing your Web services. When finished, it brings you back to this screen.

Click the **Next** button to go to the next step in the wizard deployment process.

Deployment Review

At this point, you will receive a review of your application deployment modules and configuration, as follows:

Figure 2–13 Review

Deploy Application: Review

Ear File to Deploy	petstore.ear
Deployment Destination	Instance OC4J_EM
URLs Mapped to Application	/estore

TIP The HTTP listener will be restarted after deployment, to pick up the new web module mappings.

Step 4 of 4

In order to deploy this application, click on the **Deploy** button. A message will be displayed that tells you that your application deployed.

Post-Deployment Application Modifications

You can modify any fields and add additional configuration by returning to the OC4J Home page, select the application name in the Applications section. This brings you to a screen with the details of the deployed application.

From within this screen, you can view the Web and EJB modules. In addition, you can add and modify application-specific properties, resources, and security options in the Administration section. It is in this Administration section, that you can add application-specific data sources or security groups or users mentioned in the deployment wizard.

Impact of Undeploying/Redeploying an Application

Undeploying a J2EE application from an OC4J instance results in the following:

- The application is removed from the OC4J runtime and is no longer accessible to clients.
- All bindings for the Web applications are removed from all the Web sites to which the Web modules were bound.
- All application files are removed from both the `applications/` and `application-deployments/` directories.
- In an Oracle Application Server environment, Oracle HTTP Server will be restarted to remove the mount point defined for the application. This will result in the loss of existing HTTP sessions.

During a redeployment, OC4J removes the existing application (EAR/WAR) before redeploying the new EAR. This means, for example, that attempts to access an HTML file that was included in the previous application, but not the new one, will result in "File Not Found" errors.

Also note that a redeployed WAR file overlays the previously expanded WAR, meaning that some older files may persist in the new deployment and will need to be deleted. For example, static HTML files from the previous deployment that are not included in the new WAR may continue to reside in the expanded WAR directory structure, and we would have to be manually deleted.

Impact of "Hot Deploying" an Application

The term *hot deployment* refers to the process of deploying archive files - EARs, WARs, JARs, etc. - and their associated XML descriptor files on a production application server without re-starting or "bouncing" the server.

When an EAR is redeployed or "hot redeployed" on a running OC4J instance, the status of the classes loaded in the JVM from the previous application may vary. In some cases a classloader may recognize that a class or JAR file in the file system has changed, and reload the class or library. In other cases whether a new class definition is loaded may depend on whether the JVM tuning allows the garbage collector to flush the existing class definition.

Issues may also exist with respect to serialized objects containing session data. If the class related to a session object changes, it may not be possible to cast the generic session object back to the class, since the class has changed and its variables may occupy a different memory footprint. This may result in lost session data.

Deploying a new Web module to an active OC4J instance also has a negative impact on existing sessions. Specifically, the HTTP sessions for every Web application running within the server instance will be lost by default.

You can avoid this issue on non-clustered OC4J instances by defining a "persistence directory" in each Web application's `orion-web.xml` file. Existing HTTP sessions will be stored in this temporary location across application deployments.

Specify a relative path to this directory as the value of the `persistence-path` attribute in the root `<orion-web-app>` element within each `orion-web.xml` file. For example:

```
<orion-web-app ...
  persistence-path="persistDir"
  ...>
</orion-web-app>
```

Note that this feature is not applicable for OC4J instances within a clustered environment. For guidelines on addressing this issue in a clustered environment, see the *Oracle Application Server High Availability Guide*.

What Happens When You Deploy?

When you become more proficient with OC4J and deploying applications, you should acquaint yourself with what OC4J does for you. The following sections help you understand these tasks:

- [OC4J Tasks During Deployment](#)
- [Verifying the Configuration of Deployed J2EE Applications](#)

OC4J Tasks During Deployment

When you deploy your application, the following occurs:

OC4J opens the EAR file and reads the descriptors.

1. OC4J opens, parses the `application.xml` that exists in the EAR file. The `application.xml` file lists all of the modules contained within the EAR file. OC4J notes these modules and initializes the EAR environment.
2. OC4J reads the module deployment descriptors for each module type: Web module, EJB module, connector module, or client module. The J2EE descriptors

are read into memory. If OC4J-specific descriptors are included, these are also read into memory. The JAR and WAR file environments are initialized.

3. OC4J notes any unconfigured items that have defaults and writes these defaults in the appropriate OC4J-specific deployment descriptor. Thus, if you did not provide an OC4J-specific deployment descriptor, you will notice that OC4J provides one written with certain defaults. If you did provide an OC4J-specific deployment descriptor, you may notice that OC4J added elements.
4. OC4J reacts to the configuration details contained in both the J2EE deployment descriptors and any OC4J-specific deployment descriptors. OC4J notes any J2EE component configurations that require action on OC4J's part, such as wrapping beans with their interfaces.
5. After defaults have been added and necessary actions have been taken, OC4J writes out the new module deployment descriptors to the `application-deployments/` directory. These are the descriptors that OC4J uses when starting and restarting your application. But do not modify these descriptors. Always change your deployment descriptors in the "master" location.
6. OC4J copies the EAR file to the "master" directory. This defaults to the `applications/` directory. You can change the "master" directory in the Server Properties page off of the OC4J Home Page. In the General section, modify the Application Directory field to the new location of the "master" directory. The location of the directory is relative to `/j2ee/home/config`.

Note: Each time you deploy this EAR file without removing the EAR file from the `applications/` directory, the new deployment renames the EAR file prepended with an underscore. It does not copy over the EAR file. Instead, you can copy over the EAR file. OC4J notices the change in the timestamp and redeploys the EAR.

7. Finally, OC4J updates the `server.xml` file with the notation that this application has been deployed.

Verifying the Configuration of Deployed J2EE Applications

After deployment, you can see your application configuration in the `server.xml` and `default-web-site.xml` files, as follows:

- In `server.xml`, each existing application contains a line with an `<application name=... path=... auto-start="true" />` entry. The `auto-start` attribute designates that you want this application automatically started when OC4J starts. The path is either the location of the EAR file to be deployed or the exploded directory where the application has been built. See ["Basic Deployment"](#) on page 2-11 or ["Building and Deploying Within a Directory"](#) on page 3-22 for more information.
- In `default-web-site.xml`, a `<web-app...>` entry exists for each Web application that is bound to the Web site upon OC4J startup. Because the name attribute is the WAR filename (without the `.WAR` extension), there is one line for each WAR file included in your J2EE application.

For each Web application binding included in a WAR file, the following line has been added:

```
<web-app application="myapp" name="myapp-web" root="/myapp" />
```

- The `application` attribute is the name provided in the `server.xml` as the application name.
- The `name` attribute is the name of the WAR file, without the `.WAR` extension.
- The `root` attribute defines the root context for the application off of the Web site. For example, if you defined your Web site as `"http://<ohs_host>:7777/j2ee"`, then to initiate the application, point your browser at `"http://<ohs_host>:7777/j2ee/myapp"`.

Note: Wait for automatic startup to complete before trying to access the client. The client fails on lookup if it tries to access before the completion of these processes.

Recovering From Deployment Errors

If the deployment process is interrupted for any reason, you may need to clean up the temp directory, which by default is `/var/tmp`, on your system. The deployment wizard uses 20 MB in swap space of the temp directory for storing information during the deployment process. At completion, the deployment wizard cleans up the temp directory of its additional files. However, if the wizard is interrupted, it may not have the time or opportunity to clean up the temp directory. Thus, you must clean up any additional deployment files from this directory yourself. If you do not, this directory may fill up, which will disable any further deployment. If you receive an Out of Memory error, check for space available in the temp directory.

To change the temp directory, set the command-line option for the OC4J process to `java.io.tmpdir=<new_tmp_dir>`. You can set this command-line option in the Server Properties page. Drill down to the OC4J Home Page. Scroll down to the Administration Section. Select **Server Properties**. On this page, Scroll down to the Command Line Options section and add the `java.io.tmpdir` variable definition to the OC4J Options line. All new OC4J processes will start with this property.

Undeploying Web Applications

You can remove a J2EE Web application from the OC4J Web server by selecting the application in the Applications section of the OC4J Home Page (see [Figure 2-2](#)) and clicking the **Undeploy** button. This command removes the deployed J2EE application and results in the following:

- The application is removed from the OC4J runtime.
- All bindings for the Web modules are removed from all the Web sites to which the Web modules were bound.
- The application files are removed from both the `applications/` and `application-deployments/` directories.

Note: You can also undeploy applications with the DCM command. See the *Distributed Configuration Management Administrator's Guide* for directions.

Advanced Configuration and Development

Chapter 2, "Configuration and Deployment", discusses basic configuration, development, and deployment of a J2EE application. This chapter discusses both global J2EE service configuration and advanced J2EE application configuration.

This chapter discusses the following topics:

- [Configuring OC4J Using Enterprise Manager](#)
- [Overview of OC4J and J2EE XML Files](#)
- [Sharing Libraries](#)
- [Understanding and Configuring OC4J Listeners](#)
- [Configuring Oracle HTTP Server With Another Web Context](#)
- [Building and Deploying Within a Directory](#)
- [Developing Startup and Shutdown Classes](#)
- [Setting Performance Options](#)
- [Enabling OC4J Logging](#)
- [OC4J Debugging](#)

Configuring OC4J Using Enterprise Manager

You can configure J2EE services, J2EE applications, and Oracle Application Server clusters with Enterprise Manager. Some aspects are configured at the OC4J instance level; thus, they provide a global configuration for all deployed applications in the instance. Other aspects are configured at the application level; thus, this type of configuration is local and applicable only to the application.

The following sections provide you with an overview of advanced configuration within Enterprise Manager for OC4J:

- [OC4J Instance Level Configuration](#)
- [Application Level Configuration](#)

OC4J Instance Level Configuration

Off of the OC4J Home page, you can configure global services that apply to all applications by drilling down to the Administration page, which allows you to configure the following:

- [Configuring Server Properties](#)
- [Configure a Web Site](#)
- [Configure JSP Container Parameters](#)
- [Configure Replication Parameters](#)
- [Advanced Configuration Through XML Files](#)
- [Configure Data Sources](#)
- [Configure Security](#)
- [Configure JMS](#)
- [Configure Global Web Application Parameters](#)
- [Configure RMI](#)

Configuring Server Properties

To configure OC4J properties, scroll down to the Instance Properties section and select Server Properties. The General section of this page contains the following fields:

Figure 3–1 General Section of the Instance Properties Page

General

Name	home
Server Root	/private/j2ee/home/config
Configuration File	/private/j2ee/home/config/server.xml
Default Application	default
Name	
Default Application	application.xml
Path	
Default Web Module	<input type="text" value="global-web-application.xml"/>
Properties	
Application Directory	<input type="text" value="../applications"/>
Deployment Directory	<input type="text" value="../application-deployments"/>

The following information about the default server properties is displayed in the upper half of the page.

- Default application—The default application is what most deployed applications used as its parent. Thus, these deployed applications can see the classes within the default application. See "[Select Application](#)" on page 2-12 for more information on parent applications.
- Default application path—There exists a file named `application.xml`, which is separate from the `application.xml` included with each EAR file. This `application.xml` file is known as the global `application.xml` file. It defines

properties that are used across all deployed applications within this OC4J instance.

In this section, you can modify OC4J server defaults. These are listed below:

- Default Web module properties—These are specified in an XML file called `global-web-application.xml`. If you want it to refer to another XML file, you can change the pointer to this file here. However, this file must conform to the DTD that Oracle specifies. The directory is relative to `j2ee/home/config`.

If you want to actually modify elements contained within this file, update entries in either the Web Site Properties or Advanced Properties section. These are discussed more in ["Configure a Web Site"](#) on page 3-4 and ["Modifying XML Files Included in the Deployed Application EAR File"](#) on page 3-14.

- Application and deployment directories—The default directory to place the "master" EAR file of the deployed application is the `/applications` directory. The default directory is where OC4J places modified module deployment descriptors with added defaults. Currently, this location is in the `/application-deployments` directory. You can change the locations of the default directories in these fields. The directory is relative to `j2ee/home/config`. See ["What Happens When You Deploy?"](#) on page 2-19 for more information on the usage of these directories.

The next section, Multiple VM Configuration, is dedicated as part of the cluster configuration. The following details what each field means; however, the context of how to configure clusters using this section is discussed fully in [Chapter 4, "OC4J Clustering"](#).

Figure 3–2 Clustering and Ports

Multiple VM Configuration

 **TIP** If OC4J is running, newly added islands and associated processes will be automatically started.

Islands

Island ID	Number of Processes	Related Links	Virtual Machine Metrics
default_island	1		
<input type="button" value="Add Another Row"/>			

Ports

RMI Ports	<input type="text" value="3101-3200"/>
JMS Ports	<input type="text" value="3201-3300"/>
AJP Ports	<input type="text" value="3000-3100"/>

RMI-IIOP Ports

IIOP Ports	<input type="text" value="3401-3500"/>
IIOP SSL (Server only)	<input type="text"/>
IIOP SSL (Server and Client)	<input type="text"/>

- Islands—Designate the number of islands within the cluster. Each island is created when you click on the Add Another Row button. You can supply a name for each island within the Island ID field. You can designate how many OC4J processes should be started within each island by the number configured in the Number of Processes field. For more information on configuring islands for clustering, see ["OC4J Cluster Configuration"](#) on page 4-7.

- Ports—This section enables you to configure what the port ranges should be for RMI, JMS, AJP, and IIOP. For more information on these services, see the *Oracle Application Server Containers for J2EE Services Guide*.

Figure 3–3 Command-Line Options and Environment Variables

Command Line Options

Java Executable	<input type="text"/>	Related Links Tracing Properties
OC4J Options	<input type="text"/>	
Java Options	<input type="text"/>	

Environment Variables

Select	Name	Value
<input checked="" type="radio"/>	DISPLAY	:0.0
<input type="radio"/>	LD_LIBRARY_PATH	/ade/plaquerr_emdw4_nedc/oracle/lib
<input type="radio"/>	ORACLE_HOME	/ade/plaquerr_emdw4_nedc/oracle

- Command Line Options—This section enables you to configure the following:
 - the Java executable command that should be used, such as javac
 - any OC4J options to include when starting a new OC4J process
 - any Java options to include when executing 'java'
 - any OC4J system properties

See "OC4J Command-Line Options and System Properties" on page B-30 for a list of command-line options and system properties.
- Environment Variables—This section enables you to configure environment variables for OC4J. To add a new environment variable, click **Add Environment Variable**. A new row is added for you to define the variable name in the left column and the value in the right column.

Configure a Web Site

To configure your Web site, select Website Properties in the Instance Properties column on the Administration page.

The Web site page has two sections. In the first section, you can see what is the default Web application. In the second section—URL Mappings for Web Modules—you can specify whether each Web application is to be loaded upon startup. These parameters are discussed in more detail in the *Oracle Application Server Containers for J2EE Servlet Developer's Guide* and are stored in the default-web-site.xml file.

Figure 3–4 Website Properties

Website Properties

Default Web Module

Name **defaultWebApp**
 Application **default**
 Load on startup **true**

URL Mappings for Web Modules

Name	Application	URL Mapping	Load on startup
dms	default	/dmsoc4j	<input type="checkbox"/>
FAQAppWeb	FAQAPP_SB	/FAQApp	<input checked="" type="checkbox"/>
FAQAppWebService	FAQAPP_SB	/FAQAppWebService	<input checked="" type="checkbox"/>

Configure JSP Container Parameters

You can configure global JSP Container parameters. These apply to all JSPs deployed in this OC4J instance. To configure JSP Container parameters, select JSP Container Properties in the Instance Properties column on the Administration page. This brings you to the following page:

Figure 3–5 Oracle JSP Container Properties

Oracle JSP Container Properties

The following properties may be used to configure the Oracle JSP Container.

Debug Mode	<input type="button" value="No"/>	Emit Debug Info	<input type="button" value="No"/>
External Resource for Static Content	<input type="button" value="No"/>	When a JSP Changes	<input type="button" value="Recompile JSP"/>
Generate Static Text as Bytes	<input type="button" value="No"/>	Precompile Check	<input type="button" value="No"/>
Tags Reuse Default	<input type="button" value="No"/>	Validate XML	<input type="button" value="No"/>
Reduce Code Size for Custom Tags	<input type="button" value="No"/>		
SQLJ Command	<input type="text"/>		
Alternate Java Compiler	<input type="text"/>		

Revert

Apply

Most of the properties indicated here are described in Chapter 3 of the *Oracle Application Server Containers for J2EE Support for JavaServer Pages Developer's Guide*.

These properties can be included within the `global-web-application.xml` file within the `<servlet>` element.

Configure Replication Parameters

For clustering servlet or EJBs, you may need to configure replication parameters. See ["OC4J Instance Configuration"](#) on page 4-8 for details.

Advanced Configuration Through XML Files

In OC4J version 1.0.2.2, you configured the OC4J server and all deployed application parameters through XML files. Since the OC4J server existed on a single node and did not need high availability and clustering management, this worked well. However, with the integration of OC4J with Oracle Application Server, increased enterprise management abilities with clustering and high availability options, all configuration must be accomplished through Enterprise Manager.

When you select Advanced Properties off of the Administration page, you can modify the OC4J Server XML files. These include the XML files that configure the server and its services. The files that are in this group are `server.xml`, `global-web-application.xml`, `rmi.xml`, `jms.xml`, and `default-web-site.xml`. Modify any of these XML files in the Advanced Properties page off of the OC4J Home Page.

Other XML configuration files can be modified in other areas of the Oracle9iAS Console.

- **Global application XML files:** These include XML files that apply to all applications deployed in the OC4J instance. These include the `global-application.xml`, `data-sources.xml`, the security XML file and `oc4j-connectors.xml`. To modify these XML files, select Applications off of the OC4J Home Page. On the Applications page, select default. On the default application page, scroll down to the Administration section and choose Advanced Properties.
- **Local application XML files.** You can modify XML files that configure the overall application. These include local data sources, local security, and OC4J-specific application configuration. These XML files include `data-sources.xml`, `orion-application.xml`, and security XML files. To modify these files, drill down to the specific application from the Deployed Applications section on the Applications page. On the specified application screen, scroll down to the Administration section and choose Advanced Properties.
- **Application module XML files:** When the EAR or WAR file is deployed, you provided module deployment descriptors, such as `web.xml`, `orion-web.xml`, `ejb-jar.xml`, and `orion-ejb-jar.xml`. You can modify parameters only in the OC4J-specific (`orion-xxx.xml`) XML files. You cannot modify the J2EE XML files, such as `web.xml` or `ejb-jar.xml`. For more information on modifying these XML files, see ["Modifying XML Files Included in the Deployed Application EAR File"](#) on page 3-14.

As an example, the `server.xml` page is shown. Notice that you can hand edit the XML elements.

Figure 3–6 Edit Server.XML Parameters

Edit server.xml

This configuration file is located at server.xml

```
<?xml version = '1.0'?>
<!DOCTYPE application-server PUBLIC "-//Evermind//DTD Orion Application-
server//EN" "http://xmlns.oracle.com/ias/dtds/application-server.dtd">
<application-server localhostIsAdmin="true" application-directory="../applications"
deployment-directory="../application-deployments" connector-directory="../connectors">
  <library path="../../../../jdk/lib/tools.jar"/>
  <rmi-config path="/rmi.xml"/>
  <jms-config path="/jms.xml"/>
  <log>
    <file path="/log/server.log"/>
  </log>
  <transaction-config timeout="30000"/>
  <global-application name="default" path="application.xml"/>
  <application name="petstore" path="../applications/petstore.ear" auto-start="true"/>
```

Revert

Apply

For an overview of the OC4J XML files, see ["Overview of OC4J and J2EE XML Files"](#) on page 3-14 for a discussion of these files and their relation to each other. Other books in the OC4J documentation set describe the elements within each of these files.

Configure Data Sources

You can configure global or local data sources. A global data source is available to all deployed applications in this OC4J instance. A local data source is configured within the deployed application and can only be used by that application.

See *Oracle Application Server Containers for J2EE Services Guide* for a full explanation of how to configure a data source and the elements within the `data-sources.xml` file.

To configure global data sources, select one of the following off of the OC4J Home Page:

- Data Sources under the Application Defaults column on the Administration page—This page allows you to add data source definitions one field at a time. See ["Data Source Field Page"](#) on page 3-8 for a description of this page.
- Off the default application page—You can either modify or add a data source either through a GUI or by directly accessing the `data-sources.xml` file.
 1. Select Applications off of the OC4J Home Page.
 2. On the Applications page, select default next to the Default Application Name.
 3. On the default application page, scroll down to the Administration section and do one of the following:
 - Select Data Sources under the Resources column. This allows you to add or modify a data source using a GUI form.

- Select Advanced Properties under the Properties column. Select `data-sources.xml` on this page. This allows you to add data sources using the XML definitions. This is useful if you have been provided the XML. You can just copy in the already configured data sources.

To configure local data sources, you perform the same selection off of the application page. You drill down to the particular application to which this data source is local. On the application page, choose Data Source under the Resources column. It displays the same data source field page that is discussed in "Data Source Field Page" on page 3-8.

Data Source Field Page To configure a new Data Source, click **Add Data Source**. This brings you to a page where you can enter all configuration details about the data source. This page is divided up into four sections.

Figure 3-7 shows the General section.

Figure 3-7 General Section of Data Source Definition

Use this page to configure a data source to connect to Oracle or non-Oracle databases. To connect to Oracle databases, configure either a non-emulated (pure Oracle) Data Source or an emulated (wrappers around Oracle Data Sources) Data Source. To connect to non-Oracle databases, use the `com.evermind.sql.DriverManagerDataSource` with the Merant JDBC drivers. Please refer to the online help for additional information.

General

Name	<input type="text"/>
Description	<input type="text"/>
* Data Source Class	<input type="text"/>
JDBC URL	<input type="text"/>
JDBC Driver	<input type="text"/>
	<small>This field is required if you are using a generic Orion Data Source Class.</small>
Schema	<input type="text"/>

The General section enables you to define the following aspects about a data source:

- Name—A user-defined name to identify the data source.
- Description—A user-defined description of the data source.
- Data Source Class—This is the class, such as `com.evermind.sql.DriverManagerDataSource`, that the data source is instantiated as.
- JDBC URL—The URL to the database service represented by this data source. For example, if using an Oracle Thin driver, the URL could be the following:
`jdbc:oracle:thin:@my-lap:1521:<service_name>`.
- JDBC Driver—The JDBC driver to use. One example of a JDBC driver is `oracle.jdbc.driver.OracleDriver`.
- Schema—This is an optional parameter. Input the file name that contains the Java to database mappings for a particular database.

Figure 3–8 shows the username and password.

Figure 3–8 Username and Password

Datasource Username and Password

Cleartext passwords may pose a security risk, especially if the permissions on the data-sources.xml configuration file allows it to be read by any user. You can specify an indirect password to avoid this risk. An indirect password is used to do a look up in the User Manager to get the password.

Username

Use Cleartext Password

 Password

Use Indirect Password

 Indirect Password

example: Scott, customers/Scott

Username/Password—The username and password used to authenticate to the database that this data source represents. The password can either be entered as clear text, or you can provide a username for an indirect password. For details, see the *Oracle Application Server Containers for J2EE Services Guide*.

Figure 3–9 shows the JNDI Locations section.

Figure 3–9 JNDI Locations

JNDI Locations

[Return to Top](#)

For an emulated datasource, please specify all three location attributes. It is recommended that you reference the EJB Location attribute in your code to look up this datasource. For a non-emulated datasource, the location attribute is all that is needed.

The Location field is required

Location

Transactional(XA) Location

For emulated data sources, retrieve the data source using the JNDI value in this field.

EJB Location

The JNDI Locations section enables you to define the JNDI location string that the data source is bound with. This JNDI location is used within the JNDI lookup for retrieving this data source.

Figure 3–10 shows the Connection Attributes section.

Figure 3–10 Connection Attributes

Connection Attributes

Connection Retry Interval (secs)	<input type="text" value="1"/>
Max Connection Attempts	<input type="text"/>
Cached Connection Inactivity Timeout(secs)	<input type="text"/>
The following attributes only apply if you are using pooled data sources	
Maximum Open Connections	<input type="text"/>
Minimum Open Connections	<input type="text"/>
Wait For Free Connection Timeout(secs)	<input type="text"/>

This section enables you to modify connection tuning parameters, including the retry interval, pooling parameters, timeout parameters, and maximum attempt parameter.

Figure 3–11 shows the Properties section for the data source.

Figure 3–11 Properties

Properties

Properties may be set when configuring a custom or 3rd-party data source.

⊖ Previous ▾ Next ⊕

Select Name	Value
(No items found in J2EE deployment descriptor)	
<input type="button" value="Add a Property"/>	

If your data source is a third party data source, you may need to set certain properties. These properties would be defined in the third-party documentation. In addition, properties must be set for JTA transactions for the two-phase commit coordinator.

Configure Security

The user manager, employing the user name and password, verifies the user's identity based on information in the user repository. The user manager defines what type of authentication you will be using. It contains your definitions for users, groups, or roles. The default user manager is the JAZNUserManager. You can define a user manager for all applications or for specific applications.

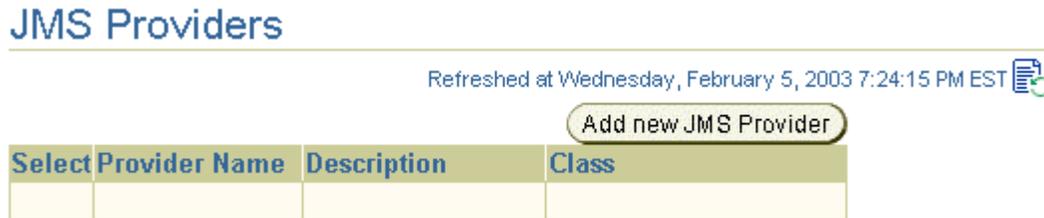
See the Oracle Application Server Containers for J2EE Security Guide for a full description of OC4J security, including user managers.

Configure JMS

JMS can be configured either within the JMS section or directly within the `jms.xml` file.

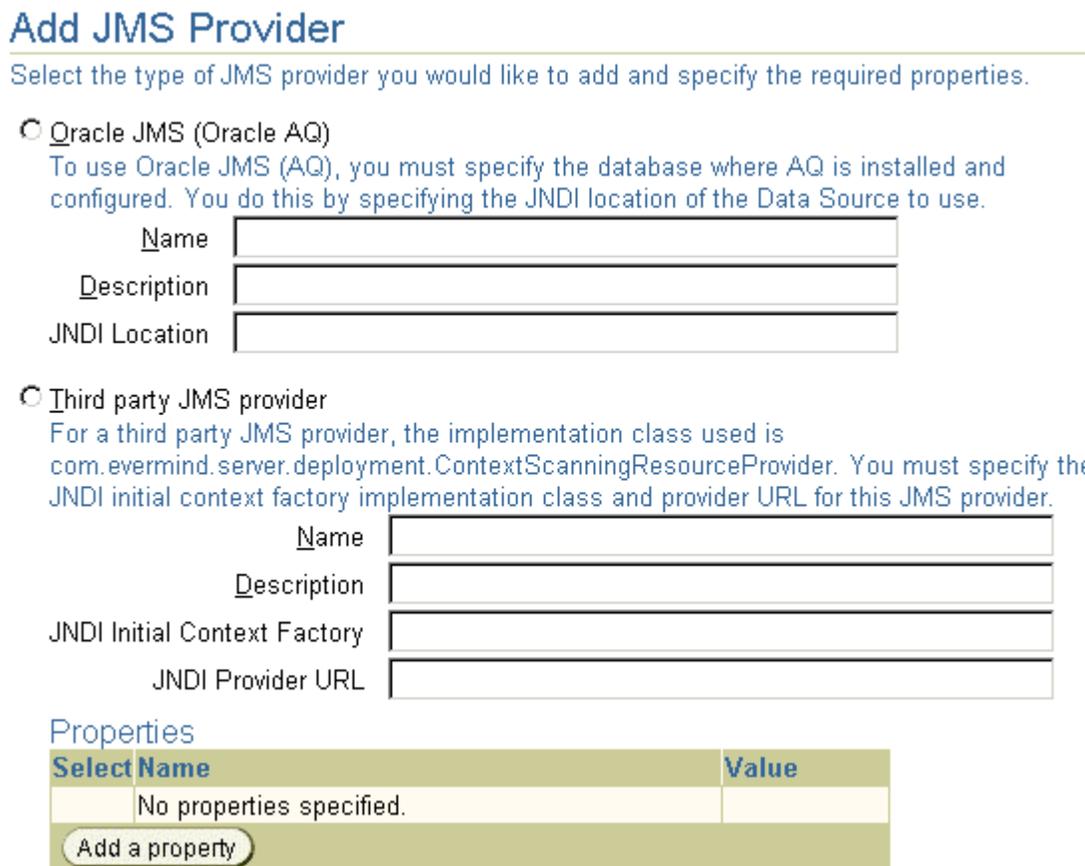
Edit the JMS Section To add Oracle JMS or third-party JMS providers, select JMS Providers under the Application Defaults column on the Administration page. This brings you to the following page:

Figure 3–12 Defining JMS Providers



Click the **Add new JMS Provider** button to configure each JMS provider, which brings up the following page:

Figure 3–13 Adding a JMS Provider



This page enables you to configure either Oracle JMS or a third-party JMS provider. OracleAS JMS is always provided and preconfigured, except for the topics and queues, with the OC4J installation.

Once you choose the type of JMS provider, you must provide the following:

- Oracle JMS: Provide the data source name and JNDI location for the database where Oracle JMS is installed and configured.

- Third-party JMS provider: Provide the name, JNDI initial context factory class, and JNDI URL for the third-party provider. To add JNDI properties for this JMS provider, such as `java.naming.factory.initial` and `java.naming.provider.url`, click **Add a property**. A row is added where you can add the name for each JNDI property and its value.

This only configures the providers; it does not configure the `Destination` objects (topic, queue, and subscription). See the *Oracle Application Server Containers for J2EE Services Guide* or more information on JMS providers.

To configure a JMS provider that is only for a specific application, select the application from the Applications page, scroll down to the Resources column, and select JMS Providers. The screens that appear are the same as for the default JMS provider.

Edit the JMS XML File To add queues and topics for OracleAS JMS, you can edit the `jms.xml` file directly as follows: select the Advanced Server Properties section under the Instance Properties column on the Administration page. In this section, choose `jms.xml` to modify the straight XML file. See the *Oracle Application Server Containers for J2EE Services Guide* or descriptions of the elements in the `jms.xml` file.

Configure Global Web Application Parameters

To configure Web parameters that apply to all deployed Web applications, select Global Web Module in the Application Defaults column on the Administration page. This brings you to the following page:

Figure 3–14 Defining Web Modules

Web Module: Global Web Module

Refreshed at Tuesday, February 4, 2003 5:49:33 PM EST 

Servlets/JSPs

Name	Type	Source	Startup Priority
cgi	Servlet	com.evermind.server.http.CGIServlet	
jsp	Servlet	oracle.jsp.runtimev2.JspServlet	
perl	Servlet	com.evermind.server.http.CGIServlet	
php	Servlet	com.evermind.server.http.CGIServlet	
rmi	Servlet	com.evermind.server.rmi.RMIHttpTunnelServlet	
rmip	Servlet	com.evermind.server.rmi.RMIHttpTunnelProxyServlet	
ssi	Servlet	com.evermind.server.http.SSIServlet	

Administration

Properties

[General](#)

[Mappings](#)

[Filtering and](#)

[Chaining](#)

Security

[General](#)

The type of parameters that you can define for Web modules concern mappings, filtering, chaining, environment, and security. Drill down into each of the provided links under the Properties and Security columns to modify these parameters. Each of these parameters are discussed in detail in the *Oracle Application Server Containers for J2EE Servlet Developer's Guide*. These parameters are stored in the `global-web-application.xml` and `orion-web.xml` files. This guide discusses the elements in these files.

Configure RMI

RMI can only be defined within an XML definition. To edit the `rmi.xml` file, select Advanced Properties under the Instance Properties column on the Administration page. In this section, choose `rmi.xml` to modify the XML file. See the *Oracle Application Server Containers for J2EE Services Guide* for descriptions of the elements in the `rmi.xml` file.

Application Level Configuration

You can deploy, redeploy, or undeploy a J2EE application that exists in an EAR or WAR file archival format. To deploy an application, click the **Deploy EAR File** or **Deploy WAR File** buttons to deploy in the Deployed Applications section on the Applications page.

This starts the deployment wizard that is covered in "[Deploying Applications](#)" on page 2-11. If you deploy an EAR file, it must contain an `application.xml` that describes the application modules; if you deploy a WAR file, the `application.xml` file is created for you automatically.

To undeploy, click the **Select** radio button for the application and then click the **Undeploy** button.

To redeploy, click the **Select** radio button for the application and then click the **Redeploy** button.

Note: You can also deploy, undeploy, or redeploy simple applications with the DCM command. See the Distributed Configuration Management Administrator's Guide for directions.

Once you have deployed your application, you can modify most of the parameters for this application. To configure application-specific parameters, do the following:

1. On the OC4J Home Page, select the Applications page.
2. Select the application where you want to change the configuration using one of the following methods:
 - a. Click the **Select** radio button for the application and click the **Edit** button.
 - b. Select the application name in the Name column in the applications box.

This page is the initiating point for changing general application configuration as well as configuration specific to a certain part of your deployed application, such as a WAR file.

The following sections provide a brief overview of these configuration options:

- [Configuring Application General Parameters](#)
- [Configuring Local J2EE Services](#)

- [Modifying XML Files Included in the Deployed Application EAR File](#)

Configuring Application General Parameters

If you drill down to the application, scroll down to the Properties column and select the General link, you can configure a multitude of application details, as follows:

- persistence path
- data sources path
- library paths
- EJB properties
 - automatically create database tables for CMP beans
 - automatically delete old database tables for CMP beans
- default data source (JNDI name)
- User Manager configuration

Configuring Local J2EE Services

As described in "[Configure Data Sources](#)" on page 3-7 and "[Configure Security](#)" on page 3-10, you can configure data sources and security either for all deployed applications (global) or only for a specific application (local). See these sections for directions on how to configure your J2EE services for your application.

Modifying XML Files Included in the Deployed Application EAR File

You can modify only the OC4J-specific XML files of your application after deployment. This includes `orion-ejb-jar.xml`, `orion-web.xml`, and `orion-application-client.xml`. You cannot modify the J2EE XML files, such as `web.xml`, `ejb-jar.xml`, and `application-client.xml`.

In order to modify the OC4J-specific XML files, do the following:

1. From the application screen, select the JAR or WAR file whose configuration you are interested in modifying. The application screen is shown.
2. You can modify parameters for the application in one of the following manners:
 - Follow links in the Administration section for modifying parameters.
 - Select the bean or servlet in the section that details the beans, servlets, or JSPs deployed. This drills down to another level of configuration.
 - The Administration section contains either a Properties or Advanced Properties section that allows you to modify XML directly for the OC4J-specific deployment descriptors—`orion-ejb-jar.xml`, `orion-web.xml`, and `orion-application-client.xml`.

Overview of OC4J and J2EE XML Files

This section contains the following topics:

- [XML Configuration File Overview](#)
- [XML File Interrelationships](#)

XML Configuration File Overview

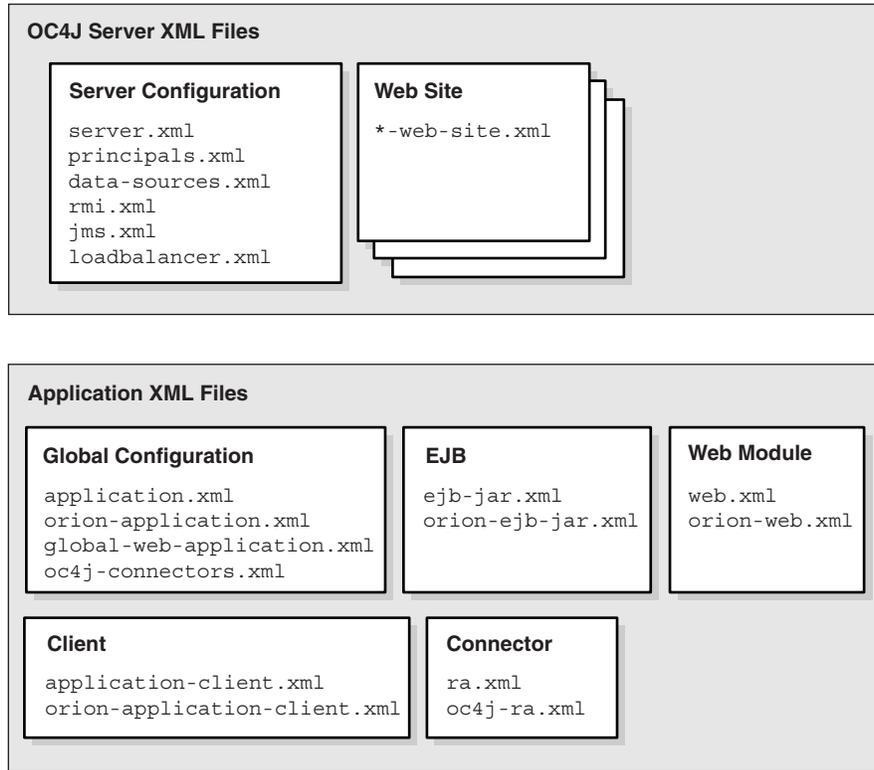
Each XML file within OC4J exists to satisfy a certain role; thus, if you have need of that role, you will understand which XML file to modify and maintain.

Figure 3–15 illustrates all the OC4J XML files and their respective roles.

- **OC4J server:** All XML files within this box are used to set up this instance of the OC4J server. These files configure things such as listening ports, administration passwords, security, and other basic J2EE services.

These files configure the OC4J server and point to other key configuration files. The settings in the OC4J configuration files are not related to the deployed J2EE applications directly, but to the server itself.
- **Oracle HTTP Server:** These files are configuration files within the Oracle HTTP Server. However, they are included in this diagram because you may need to modify these to change how requests are handed off to the OC4J server.
- **Web site:** These XML files configure listening ports, protocols, and Web contexts for the OC4J Web site.
- **Application XML files:** Each J2EE application type (EJB, servlet, JSP, connector) requires its own configuration (deployment) files. Each application type has one J2EE deployment descriptor and one OC4J-specific deployment descriptor, which is denoted with an "orion-" prefix. In addition, the following are global configuration files for all components in the application:
 - The `application.xml` as the global application configuration file that contains common settings for all applications in this OC4J instance.
 - The `orion-application.xml` file contains OC4J-specific global application information for all applications in this OC4J instance.
 - The `global-web-application.xml` file contains OC4J-specific global Web application configuration information that contains common settings for all Web modules in this OC4J instance.
 - The `oc4j-connectors.xml` file contains global connector configuration information.

Figure 3–15 OC4J and J2EE Application Files



Note: Each deployed application uses an `application.xml` as its standard J2EE application descriptor file. That XML file is local to the application and separate from the global `application.xml`, which configures options that are applied to all applications deployed in this OC4J server instance.

Table 3–1 describes the role and function for each XML file that was displayed in the preceding figure.

Table 3–1 OC4J Features and Components

XML Configuration File	Features/Components
<code>server.xml</code>	OC4J overall server configuration. Configures the server and points to the XML files that add to this file, such as <code>jms.xml</code> for JMS support. The listing of other XML files enables the services to be configured in separate files, but the <code>server.xml</code> file denotes that they be used for the OC4J configuration.
<code>jazn.xml</code> <code>jazn-data.xml</code>	OC4J security configuration for JAZN security required for accessing the server.
<code>data-sources.xml</code>	OC4J data source configuration for all databases used by applications within OC4J.
<code>rmi.xml</code>	OC4J RMI port configuration and RMI tunneling over HTTP.
<code>jms.xml</code>	OC4J JMS configuration for <code>Destination</code> topics and queues that are used by JMS and MDBs in OC4J.

Table 3–1 (Cont.) OC4J Features and Components

XML Configuration File	Features/Components
default-web-site.xml	OC4J Web site definition.
mod_oc4j.conf	The mod_oc4j module is an Oracle HTTP Server module that forwards OC4J requests. This file configures the mount point that denotes what contexts to be directed to OC4J.
application.xml orion-application.xml	J2EE application standard J2EE application descriptor file and configuration files. <ul style="list-style-type: none"> ■ The global application.xml file exists in the j2ee/home/config directory and contains common settings for all applications in this OC4J instance. This file defines the location of the security XML definition file—jazn-data.xml and the datasource XML definition file—data-sources.xml. This is a different XML file than the local application.xml files. ■ The local application.xml file defines the J2EE EAR file, which contains the J2EE application modules. This file exists within the J2EE application EAR file. ■ The orion-application.xml file is the OC4J-specific definitions for all applications.
global-web-application.xml web.xml orion-web.xml	J2EE Web application configuration files. <ul style="list-style-type: none"> ■ global-web-application.xml is an OC4J-specific file for configuring servlets that are bound to all Web sites. ■ web.xml and orion-web.xml for each Web application. <p>The web.xml files are used to define the Web application deployment parameters and are included in the WAR file. In addition, you can specify the URL pattern for servlets and JSPs in this file. For example, servlet is defined in the <servlet> element, and its URL pattern is defined in the <servlet-mapping> element.</p>
ejb-jar.xml orion-ejb-jar.xml	J2EE EJB application configuration files. The ejb-jar.xml files are used to define the EJB deployment descriptors and are included in the EJB JAR file.
application-client.xml orion-application-client.xml	J2EE client application configuration files.
oc4j-connectors.xml ra.xml oc4j-ra.xml	Connector configuration files. <ul style="list-style-type: none"> ■ The oc4j-connectors.xml file contains global OC4J-specific configuration for connectors. ■ The ra.xml file contains J2EE configuration. ■ The oc4j-ra.xml file contains OC4J-specific configuration.

XML File Interrelationships

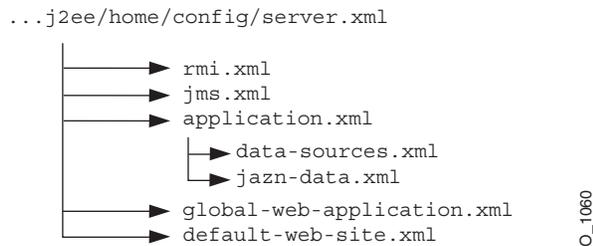
Some of these XML files are interrelated. That is, some of these XML files reference other XML files—both OC4J configuration and J2EE application (see [Figure 3–17](#)).

Here are the interrelated files:

- `server.xml`—contains references to the following:
 - All `*-web-site.xml` files for each Web site for this OC4J server, including the `default-web-site.xml` file.
 - The location of each of the other OC4J server configuration files, except `jazn-data.xml` and `data-sources.xml` which are defined in the global `application.xml`, shown in [Figure 3-15](#)
 - The location of each `application.xml` file for each J2EE application that has been deployed in OC4J
- `default-web-site.xml`—references applications by name, as defined in the `server.xml` file. And this file references an application-specific EAR file.
- `application.xml`—contains references to the `jazn-data.xml` and `data-sources.xml` files.

The `server.xml` file is the keystone that contains references to most of the files used within the OC4J server. [Figure 3-16](#) shows the XML files that can be referenced in the `server.xml` file:

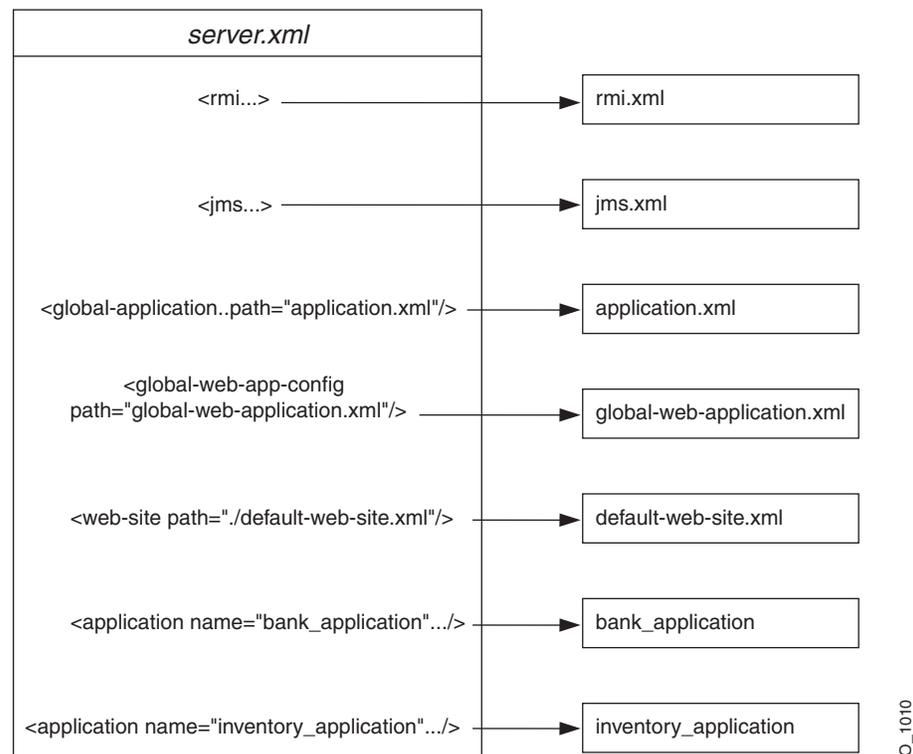
Figure 3-16 XML Files Referenced Within `server.xml`



[Figure 3-17](#) demonstrates how the `server.xml` points to other XML configuration files. For each XML file, the location can be the full path or a path relative to the location of where the `server.xml` file exists. In addition, the name of the XML file can be any name, as long as the contents of the file conform to the appropriate DTD.

- The `<rmi-config>` tag denotes the name and location of the `rmi.xml` file.
- The `<jms-config>` tag denotes the name and location of the `jms.xml` file.
- The `<global-application>` tag denotes the name and location of the global `application.xml` file.
- The `<global-web-app-config>` tag denotes the name and location of the `global-web-application.xml` file.
- The `<web-site>` tag denotes the name and location of one `*-web-site.xml` file. Since you can have multiple Web sites, you can have multiple `<web-site>` entries.

In addition to pointing to the OC4J server configuration files, the `server.xml` file describes the applications that have been deployed to this OC4J server. Each deployed application is denoted by the `<application>` tag.

Figure 3–17 Server.xml File and Related XML Files

Other tags for `server.xml` are described in ["Elements in the server.xml File"](#) on page 3-14.

Note: If you understand the OC4J XML files from previous releases of OC4J, you can simply change most of the OC4J server XML configuration files by drilling to the OC4J Home Page, scroll down to Administration, and click on Advanced Properties. From here, you can modify the XML files using an Enterprise Manager editor.

Sharing Libraries

If you have libraries that you want to share among applications, add a `<library>` element in the `global-application.xml` file, indicating the directory where you are placing the libraries, as follows:

Windows:

```
<library path="d:\oc4j\j2ee\home\applib\"/>
```

UNIX:

```
<library path="/private/oc4j/j2ee/home/applib\"/>
```

For each directory to be included, use a separate `<library>` element on a separate line, as follows:

```
<library path="/private/oc4j/j2ee/home/applib\"/>
<library path="/private/oc4j/j2ee/home/mylibrary\"/>
```

As a default, a `<library>` element exists in the global `application.xml` file with the `j2ee/home/applib` directory. Instead of modifying the `<library>` element to contain other directories, you could move your libraries into the `applib` directory. However, note that adding libraries to this directory increases the size of OC4J and effects the performance as all libraries are searched for unknown classes. Use this with discretion.

Note: The default `j2ee/home/applib` directory is not created when OC4J is installed. If you want to add shared libraries to this directory, you must first create it before adding your libraries.

If you can, you should keep your shared libraries local to the application through the `orion-application.xml` file deployed with the application. You can add `<library>` elements in the `orion-application.xml` file for the application to indicate where the libraries are located, which are used only within the application.

Understanding and Configuring OC4J Listeners

Incoming client requests use one of three protocols: AJP, HTTP, or RMI. AJP and HTTP are used for HTTP requests. AJP is used between the OHS and OC4J components. HTTP is used for HTTP requests directed past OHS to OC4J. RMI is used for incoming EJB requests.

HTTP Requests

All HTTP requests, whether through OHS or directly to OC4J, must have a listener configured in an OC4J Web site. You can have multiple Web sites for each OC4J instance, one for each protocol type. You cannot have one Web site listen for both types of protocols. Thus, OC4J provides two Web site configuration files:

- `default-web-site.xml`—This is the AJP protocol listener and the default for most HTTP requests that use Oracle Application Server. After installation, the Oracle HTTP Server front-end forwards incoming HTTP requests over the AJP port. The OC4J Web server configuration file (`default-web-site.xml`) indicates the AJP listener port. The `default-web-site.xml` file defines the default AJP port as zero. This enables OC4J and the Oracle HTTP Server to negotiate a port upon startup.

The range of port values that the AJP port can be is configured in the OPMN configuration. See the High Availability chapter in the *Oracle Application Server Administrator's Guide* for more information on OPMN.

The following shows the entry in the `default-web-site.xml` for the default AJP listener:

```
<web-site host="oc4j_host" port="0" protocol="ajp13"
  display-name="Default OC4J WebSite">
```

You can configure the AJP default Web site protocol in two places: Website Properties or Advanced Properties off of the OC4J Home Page.

- `http-web-site.xml`—This is the HTTP protocol listener. If you want to bypass OHS and go directly to OC4J - as in an OC4J standalone environment - use the port number defined in this file.

Be aware that the AJP port is chosen at random every time OC4J is started. If it chooses the same port number that is hard-coded in this XML file, there will be a

conflict. If you use this method for incoming requests, verify that the port number you choose is outside of the range for AJP port numbers, which is defined in the OPMN configuration.

The default HTTP port is 8888. The following shows the entry in the `http-web-site.xml` for an HTTP listener with a port number of 8888:

```
<web-site host="oc4j_host" port="8888" protocol="http"
  display-name="HTTP OC4J WebSite">
```

Note: In a UNIX environment, the port number should be greater than 1024, unless the process has administrative privileges.

RMI Requests

The RMI protocol listener is set up by OPMN in the RMI configuration—`rmi.xml`. It is separate from the Web site configuration. EJB clients and the OC4J tools access the OC4J server through a configured RMI port. OPMN designates a range of ports that the RMI listener could be using. When you use the `opmn:ormi: //` string in the lookup, the client retrieves automatically the assigned RMI port. See *Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide* for more.

Disabling Access Logging for a Web Application

In the OC4J 10.1.2 implementation, there is new functionality in Web site XML files to disable OC4J access logging (used to log requests to the Web site) on a per-module basis.

In general, text-based access logging is enabled for a Web site through the `<access-log>` subelement of the `<web-site>` element in the Web site XML file (such as `default-web-site.xml` or `http-web-site.xml`). Alternatively, Oracle Diagnostic Logging ("ODL-based access logging") is enabled for a Web site through the `<odl-access-log>` subelement of the `<web-site>` element.

As of the 10.1.2 release, you can disable text-based logging or ODL-based logging (as applicable) for a particular Web application (module) through the new `access-log` attribute of the `<web-app>` element for that Web application. The `<web-app>` element is another subelement of `<web-site>`. A setting of `access-log="false"` for a Web application overrides any `<access-log>` or `<odl-access-log>` element to disable logging during the operation of that Web application.

The following example disables logging for the `dms0` module of the `default` application, but leaves text-based logging enabled for the `admin_web` module:

```
<web-site ... >
...
  <web-app application="default" name="dms0" root="/dms0" access-log="false" />
  <web-app application="default" name="admin_web" root="/adminoc4j" />
  <access-log path="../log/http-web-access.log" />
...
</web-site>
```

Notes: The default setting is `access-log="true"`. With this setting, functionality is the same as for previous releases, with logging being determined solely through the presence or absence of an `<access-log>` or `<odl-access-log>` element.

If there is no `<access-log>` or `<odl-access-log>` element in a Web site XML file, then logging is disabled anyway and `access-log="false"` for an application has no additional effect.

See the 10.1.2 release of the *Oracle Application Server Containers for J2EE Servlet Developer's Guide* for related information about access logging.

Configuring Oracle HTTP Server With Another Web Context

The `mod_oc4j` module in the Oracle HTTP Server is configured at install time to direct all `j2ee/` context bound applications to the OC4J server. If you want to use a different context, such as `pubs/`, you can add another mount for this context in the `mod_oc4j.conf` configuration file.

To modify this file, drill down to the Oracle HTTP Server Page and select `mod_oc4j.conf`. The file is presented for edits in the right-hand frame.

1. Find the `Oc4jMount` directive for the `j2ee/` directory. Copy it to another line. The mount directive is as follows:

```
Oc4jMount /j2ee/* OC4Jworker
```

Note: The OC4Jworker is defined further down in the `mod_oc4j.conf` file to be the OC4J instance.

2. Modify the `j2ee/` context to your desired context. In our example, you would have another line with a `pubs/` mount configuration. Assuming that the OC4J worker name is `OC4Jworker`, then both lines would be as follows:

```
Oc4jMount /j2ee/* OC4Jworker
Oc4jMount /pubs/* OC4Jworker
```

3. Restart the Oracle HTTP Server to pick up the new mount point.

Then all incoming requests for the `pubs/` context will be directed to the OC4J server. Note that when you deploy an application using the deployment wizard, the wizard automatically adds a mount point as described here for your URL mapping.

See the *Oracle HTTP Server Administrator's Guide* for complete details on the `mod_oc4j` module configuration.

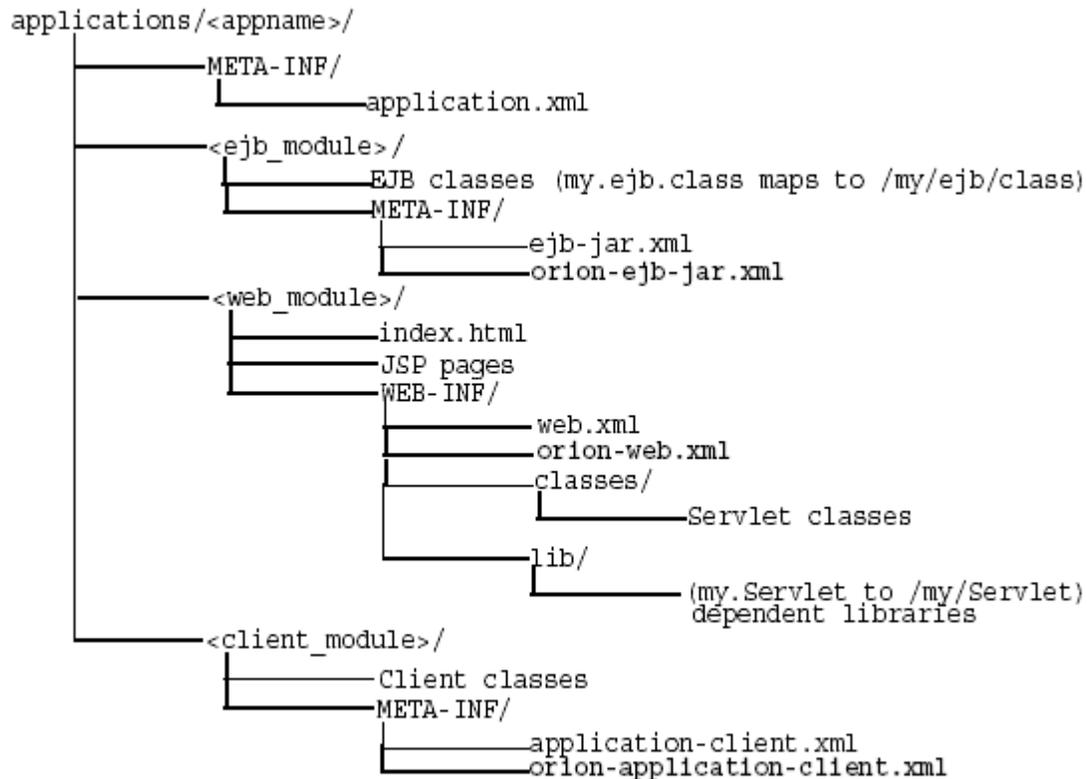
Building and Deploying Within a Directory

When developing applications, you want to quickly modify, compile, and execute your classes. OC4J can automatically deploy your applications as you are developing them within an expanded directory format. OC4J automatically deploys applications if the timestamp of the top directory, noted by `<appname>` in [Figure 3-18](#), changes. This is the directory that `server.xml` knows as the "master" location.

The application must be placed in the "master" directory in the same hierarchical format as necessary for JAR, WAR, and EAR files. For example, if `<appname>` is the

directory where your J2EE application resides, [Figure 3–18](#) displays the necessary directory structure.

Figure 3–18 Development Application Directory Structure



To deploy EJB or complex J2EE applications in an expanded directory format, complete the following steps:

1. Place the files in any directory. [Figure 3–18](#) demonstrates an application placed into `j2ee/home/applications/<appname>/`. The directory structure below `<appname>` is similar to that used within an EAR file, as follows:
 - a. Replace the EJB JAR file name, Web application WAR file name, client JAR file name, and Resource Adapter Archive (RAR) file name with a directory name of your choosing to represent the separate modules. [Figure 3–18](#) demonstrates these directory names by `<ejb_module>/`, `<web_module>/`, `<client_module>/`, and `<connector_module>/`.
 - b. Place the classes for each module within the appropriate directory structure that maps to their package structure.
2. Modify the `server.xml`, `application.xml`, and `*-web-site.xml` files. The `server.xml` and `*-web-site.xml` files are located in `j2ee/home/config` directory, while the `application.xml` is under `j2ee/home/applications/<appname>/META-INF` directory. Modify these files as follows:
 - In `server.xml`, add a new or modify the existing `<application name=... path=... auto-start="true" />` element for each J2EE application. The path points to the "master" application directory. In [Figure 3–18](#), this is `j2ee/home/applications/<appname>/`.

You can specify the path in one of two manners:

- * Specifying the full path from root to the parent directory.

In the example in [Figure 3–18](#), if `<appname>` is "myapp", then the fully-qualified path is as follows:

```
<application_name="myapp"
  path="/private/j2ee/home/applications/myapp"
  auto-start="true" />
```

- * Specifying the relative path. The path is relative to where the `server.xml` file exists to where the parent directory lives.

In the example in [Figure 3–18](#), if `<appname>` is "myapp", then the relative path is as follows:

```
<application_name="myapp" path="./applications/myapp"
  auto-start="true" />
```

- In `application.xml`, modify the `<module>` elements to contain the directory names for each module—not JAR or WAR files. You must modify the `<web-uri>`, the `<ejb>`, and the `<client>` elements in the `application.xml` file to designate the directories where these modules exist. The path included in these elements should be relative to the "master" directory and the parent of the `WEB-INF` or `META-INF` directories in each of these application types.

For example, if the `<web_module>/` directory in [Figure 3–18](#) was "myapp-web/", then the following example designates this as the Web module directory within the `<web-uri>` element as follows:

```
<module>
  <web>
    <web-uri>myapp-web</web-uri>
  </web>
</module>
```

- In the `*-web-site.xml` file, add a `<web-app...>` element for each Web application. This is important, because it binds the Web application within the Web site. The application attribute value should be the same value as that provided in the `server.xml` file. The name attribute should be the directory for the Web application. Note that the directory path given in the name element follows the same rules as for the path in the `<web-uri>` element in the `application.xml` file.

To bind the "myapp" Web application, add the following:

```
<web-app application="myapp" name="myapp-web" root="/myapp" />
```

Note: You achieve better performance if you are deploying with a JAR file. During execution, the entire JAR file is loaded into memory and indexed. This is faster than reading in each class from the development directory when necessary.

3. Add mount points for the Web application in the `mod_oc4j.conf` configuration file. Based on the `root="/myapp"` in this example, the `mod_oc4j.conf` should be updated to contain the following lines:

```
Oc4jMount /myapp home
```

Oc4jMount /myapp/* home

Developing Startup and Shutdown Classes

You can develop classes that are called after OC4J initializes or before OC4J terminates. Startup classes can start services and perform functions after OC4J initiates; shutdown classes can terminate these services and perform functions before OC4J terminates. The `oc4j.jar` must be in the Java CLASSPATH when you compile these classes.

OC4J deploys and executes the OC4J startup and shutdown classes based on configuration of these classes in the `server.xml` file.

- [OC4J Startup Classes](#)
- [OC4J Shutdown Classes](#)
- [Considerations for Startup and Shutdown Classes](#)

OC4J Startup Classes

Startup classes are executed only once after OC4J initializes. They are not re-executed every time the `server.xml` file is touched. Your startup class implements the `com.evermind.server.OC4JStartup` interface that contains two methods—`preDeploy` and `postDeploy`—in which you can implement code for starting services or performing other initialization routines.

- The `preDeploy` method executes before any OC4J application initialization.
- The `postDeploy` method executes after all OC4J applications initialize.

Each method requires two arguments—a `Hashtable` that is populated from the configuration and a `JNDI Context` to which you can bind to process values contained within the `Context`. Both methods return a `String`, which is currently ignored.

Once created, you must configure the startup class within the `<startup-classes>` element in the `server.xml` file. You access this file by selecting **Advanced Properties** on the OC4J home page. Each `OC4JStartup` class is defined in a single `<startup-class>` element within the `<startup-classes>` element. Each `<startup-class>` defines the following:

- The name of the class that implements the `com.evermind.server.OC4JStartup` interface.
- Whether a failure is fatal. If considered fatal, then when an exception is thrown, OC4J logs the exception and exits. If not considered fatal, then OC4J logs the exception and continues. Default is not fatal.
- The order of execution where each startup class receives an integer number that designates in what order the classes are executed.
- The initialization parameters that contain key-value pairs, of type `String`, which OC4J takes, which are provided within the input `Hashtable` argument. The names for the key-value pairs must be unique, as JNDI is used to bind each value to its name.

In the `<init-library path=" ../[xxx] " />` element in the `server.xml` file, configure the directory where the startup class resides, or the directory and JAR filename where the class is archived. The `path` attribute can be fully-qualified or relative to `j2ee/home/config`.

Example 3–1 Startup Class Example

The configuration for the `TestStartup` class is contained within a `<startup-class>` element in the `server.xml` file. The configuration defines the following:

- The `failure-is-fatal` attribute is `true`, so that an exception causes OC4J to exit.
- The `execution-order` is `0`, so that this is the first startup class to execute.
- Two initialization key-value pairs defined, of type `String`, which will be populated in the `Hashtable`, of the following:

```
"oracle.test.startup" "true"
"startup.oracle.year" "2002"
```

Note: The names of the key-value pairs must be unique in all startup and shutdown classes, as JNDI binds the name to its value.

Thus, configure the following in the `server.xml` file to define the `TestStartup` class:

```
<startup-classes>
  <startup-class classname="TestStartup" failure-is-fatal="true">
    <execution-order>0</execution-order>
    <init-param>
      <param-name>oracle.test.startup</param-name>
      <param-value>true</param-value>
    </init-param>
    <init-param>
      <param-name>startup.oracle.year</param-name>
      <param-value>2002</param-value>
    </init-param>
  </startup-class>
</startup-classes>
```

The container provides the two initialization key-value pairs within the input `Hashtable` parameter to the startup class.

The following example shows `TestStartup`, which implements the `com.evermind.server.OC4JStartup` interface. The `preDeploy` method retrieves the key-value pairs from the `Hashtable` and prints them out. The `postDeploy` method is a null method. The `oc4j.jar` must be in the Java CLASSPATH when you compile `TestStartup`.

```
import com.evermind.server.OC4JStartup;

import javax.naming.*;
import java.util.*;

public class TestStartup implements OC4JStartup {
    public String preDeploy(Hashtable args, Context context) throws Exception {
        // bind each argument using its name
        Enumeration keys = args.keys();
        while(keys.hasMoreElements()) {
            String key = (String)keys.nextElement();
            String value = (String)args.get(key);
            System.out.println("prop: " + key + " value: " + args.get(key));
            context.bind(key, value);
        }
    }
}
```

```

        return "ok";
    }

    public String postDeploy(Hashtable args, Context context) throws Exception {
        return null;
    }
}

```

Assuming that the `TestStartup` class is archived in `../app1/startup.jar`, modify the `<init-library>` element in the `server.xml` file as follows:

```
<init-library path="../app1/startup.jar" />
```

When you start OC4J, the `preDeploy` method is executed before any application is initialized. OC4J populates the JNDI context with the values from the `Hashtable`. If `TestStartup` throws an exception, then OC4J exits since the `failure-is-fatal` attribute was set to `TRUE`.

OC4J Shutdown Classes

Shutdown classes are executed before OC4J terminates. Your shutdown class implements the `com.evermind.server.OC4JShutdown` interface that contains two methods—`preUndeploy` and `postUndeploy`—in which you can implement code for shutting down services or perform other termination routines.

- The `preUndeploy` method executes before any OC4J application terminates.
- The `postUndeploy` method executes after all OC4J applications terminate.

Each method requires two arguments—a `Hashtable` that is populated from the configuration and a JNDI `Context` to which you can bind to process values contained within the `Context`.

The implementation and configuration is identical to the shutdown classes as described in "[OC4J Startup Classes](#)" on page 3-25 with the exception that the configuration is defined within the `<shutdown-classes>` and `<shutdown-class>` elements and there is no `failure-is-fatal` attribute. Thus, the configuration for a `TestShutdown` class would be as follows:

```

<shutdown-classes>
  <shutdown-class classname="TestShutdown">
    <execution-order>0</execution-order>
    <init-param>
      <param-name>oracle.test.shutdown</param-name>
      <param-value>>true</param-value>
    </init-param>
    <init-param>
      <param-name>shutdown.oracle.year</param-name>
      <param-value>2002</param-value>
    </init-param>
  </shutdown-class>
</shutdown-classes>

```

Assuming that the `TestShutdown` class is archived in `../app1/shutdown.jar`, add another `<init-library>` element in the `server.xml` file as follows:

```
<init-library path="../app1/shutdown.jar" />
```

Considerations for Startup and Shutdown Classes

Be aware of the following issues when developing startup/shutdown classes:

- You should separate your shutdown class from any archive files (WAR and EAR) that will be undeployed before shutdown.
- OC4J creates an instance of your startup class and an instance of your shutdown class. These are separate instances. If you implement the startup and shutdown interfaces in the same class, OC4J creates two instances of that class. In this case, you cannot share non static fields between your startup class and your shutdown class.

Setting Performance Options

Most performance settings are discussed in the *Oracle Application Server Performance Guide*.

You can manage these performance settings yourself from either the OC4J command-line option or by editing the appropriate XML file element.

- [Performance Command-Line Options](#)
- [Thread Pool Settings](#)
- [Statement Caching](#)
- [Task Manager Granularity](#)

Performance Command-Line Options

Each `-D` command-line option, except for the `dedicated.rmiContext` option, defaults to the recommended setting. However, you can modify these options by providing each `-D` command-line option as an OC4J option. See the "[OC4J Command-Line Options and System Properties](#)" on page B-30 for an example.

- `dedicated.rmiContext=true/false`. The default value is `false`. This replaces the deprecated `dedicated.connection` setting. When two or more clients in the same process retrieve an `InitialContext`, OC4J returns a cached context. Thus, each client receives the same `InitialContext`, which is assigned to the process. Server lookup, which results in server load balancing, happens only if the client retrieves its own `InitialContext`. If you set `dedicated.rmiContext=true`, then each client receives its own `InitialContext` instead of a shared context. When each client has its own `InitialContext`, then the clients can be load balanced.

This parameter is for the client. You can also set this in the JNDI properties.

- `oracle.dms.sensors=[none, normal, heavy, all]`. You can set the value for Oracle9iAS built-in performance metrics to the following: `None` (off), `normal` (medium amount of metrics), `heavy` (high number of metrics), or `all` (all possible metrics). The default is `normal`. This parameter should be set on the OC4J server. The previous method for turning on these performance metrics, `oracle.dms.gate=true/false`, is replaced by the `oracle.dms.sensors` variable. However, if you still use `oracle.dms.gate`, then setting this variable to `false` is equivalent to setting `oracle.dms.sensors=none`.
- `DefineColumnType=true/false`. The default is `false`. Set this to `true` if you are using an Oracle JDBC driver that is prior to 9.2. For these drivers, setting this variable to `true` avoids a round-trip when executing a `select` over the Oracle JDBC driver. This parameter should be set on the OC4J server.

When you change the value of this option and restart OC4J, it is only valid for applications deployed after the change. Any applications deployed before the change are not affected.

When true, the `DefineColumnType` extension saves a round trip to the database that would otherwise be necessary to describe the table. When the Oracle JDBC driver performs a query, it first uses a round trip to a database to determine the types that it should use for the columns of the result set. Then, when JDBC receives data from the query, it converts the data, as necessary, as it populates the result set. When you specify column types for a query with the `DefineColumnType` extension set to true, you avoid the first round trip to the Oracle database. The server, which is optimized to do so, performs any necessary type conversions.

Thread Pool Settings

Thread pools create and maintain a "queue" of threads for use by an OC4J process. Re-using existing threads rather than creating new threads on demand improves performance and reduces the burden on the JVM and underlying operating system.

By default, a single thread pool is created at OC4J startup. New threads are created and added to the pool on an as-needed basis. As each thread is released, it is returned to the pool to wait until it is needed by a new request.

Threads in the pool are automatically destroyed after 10 minutes of inactivity. There is no limit on the number of threads that can be created in this configuration.

The default configuration should be sufficient for most OC4J usage scenarios. However, you can optionally modify the single thread pool created by default through the `min`, `max`, `queue`, and `keepAlive` attributes of the `<global-thread-pool>` element in the `server.xml` file.

Alternatively, you can create two thread pools using `<global-thread-pool>`, with different types of threads divided among the pools:

- The *worker thread pool* contains worker threads used in processing RMI, HTTP and AJP requests, as well as MDB listener threads. These threads are process-intensive and use database resources.
- The *connection thread pool* contains threads such as listener threads, JDBC connection threads, RMI server and HTTP server connection threads, and background threads. These threads are typically not process intensive.

To create two pools, you must configure the `min`, `max`, `queue`, and `keepAlive` attributes for the worker thread pool and the `cx-min`, `cx-max`, `cx-queue`, and `cx-keepAlive` attributes for the connection thread pool. All of these attributes must be configured if creating pools; otherwise you will see the following error message:

```
Error initializing server: Invalid Thread Pool parameter: null
```

See [Table 3-2](#) on page 3-30 for descriptions of the attributes of `<global-thread-pool>`.

The following example initializes two thread pools for the OC4J process. Each contains a minimum of 10 threads and maximum of 100 threads. The number of requests outstanding in each queue can be 200 requests. Also, idle threads are kept alive for 700 seconds. The thread pool information is printed at startup.

```
<application-server ...>
...
<global-thread-pool min="10" max="100" queue="200" keepAlive="700000"
```

```

    cx-min="10" cx-max="100" cx-queue="200" cx-keepAlive="700000" debug="true"/>
    ...
</application-server>

```

Table 3–2 below describes the attributes of the `<global-thread-pool>` element. Note that this element is not included in `server.xml` by default.

Table 3–2 Attributes of `<global-thread-pool>`

Attribute	Description
<code>min</code>	<p>The minimum number of threads to create in the pool. By default, a minimum number of threads are preallocated and placed in the thread pool when the container starts.</p> <p>If you add the <code><global-thread-element></code> element to <code>server.xml</code>, the default value is set to 20. The minimum value that can be specified is 10.</p>
<code>max</code>	<p>The maximum number of threads that can be created in the pool. New threads are spawned if the maximum size is not reached and if there are no idle threads. Idle threads are used first before a new thread is spawned.</p> <p>The default is 40.</p>
<code>queue</code>	<p>The maximum number of requests that can be kept in the queue. The default is 80.</p>
<code>keepAlive</code>	<p>The length of time, in milliseconds, to keep a thread alive (idle) while waiting for a new request. After the timeout is reached, the thread is destroyed.</p> <p>To never destroy threads, set to -1. The default is 600000 milliseconds (10 minutes), which is also the minimum value allowed if not -1.</p>
<code>cx-min</code>	<p>The minimum number of threads to create in the connection thread pool.</p> <p>The minimum value that can be specified is 10.</p>
<code>cx-max</code>	<p>The maximum number of threads that can be created in the connection pool. The default is 40.</p>
<code>cx-queue</code>	<p>The maximum number of threads that can be kept in the queue in the connection pool. The default is 80.</p>
<code>cx-keepAlive</code>	<p>The length of time, in milliseconds, to keep a thread alive (idle) while waiting for a new request. After the timeout is reached, the thread is destroyed.</p> <p>To never destroy threads, set to -1. The default is 600000 milliseconds (10 minutes), which is also the minimum value allowed if not -1.</p>
<code>debug</code>	<p>If <code>true</code>, print the application server thread pool information to the console at startup. The default is <code>false</code>.</p>

Additional notes on thread pool configuration:

- The `queue` attributes should be at least twice the size of the maximum number of threads.
- The minimum and maximum number of worker threads should be a multiple of the number of CPUs installed on your machine. However, this number should be small; the more threads you have, the more burden you put on the operating system and the garbage collector.

- The `cx-min` and `cx-max` attributes are relative to the number of the physical connections you have at any point in time. The `cx-queue` handles bursts in connection traffic.

Statement Caching

You can cache database statements, which prevents the overhead of repeated cursor creation and repeated statement parsing and creation. In the `DataSource` configuration, you enable JDBC statement caching, which caches executable statements that are used repeatedly. A JDBC statement cache is associated with a particular physical connection. See *Oracle9i JDBC Developer's Guide and Reference* for more information on statement caching.

You can dynamically enable and disable statement caching programmatically through the `setStmtCacheSize()` method of your connection object or through the `stmt-cache-size` XML attribute in the `DataSource` configuration. An integer value is expected with the size of the cache. The cache size you specify is the maximum number of statements in the cache. The user determines how many distinct statements the application issues to the database. Then, the user sets the size of the cache to this number.

If you do not specify this attribute or set it to zero, this cache is disabled.

Example 3–2 Statement Caching

The following XML sets the statement cache size to 200 statements.

```
<data-source>
...
  stmt-cache-size="200"
</data-source>
```

Task Manager Granularity

The task manager is a background process that performs cleanup. However, the task manager can be expensive. You can manage when the task manager performs its duties through the `taskmanager-granularity` attribute in `server.xml`. This attribute sets how often the task manager is kicked off for cleanup. Value is in milliseconds. Default is 1000 milliseconds.

```
<application-server ... taskmanager-granularity="60000" ...>
```

Enabling OC4J Logging

OC4J logs messages both to standard error, standard out, and several log files for OC4J services and deployed applications.

- [Viewing OC4J System and Application Log Messages](#): This section describes the separate log files for OC4J sub-systems and deployed applications. You can manage how large these files can be and where they are located.
- [Redirecting Standard Out and Standard Error](#): This section describes how to forward standard out and standard error messages to a log file.

Note: Also, OC4J supports Jakarta `log4j`. See the "Open Source Frameworks and Utilities" appendix in the *Oracle Application Server Containers for J2EE Servlet Developer's Guide*.

Viewing OC4J System and Application Log Messages

Each OC4J process included in the Oracle9iAS environment has a set of log files, as shown in [Table 3–3](#). If there are multiple processes running for an OC4J instance, there is a multiple set of log files.

Table 3–3 List of Log Files Generated for OC4J

Default Log File Name	Description	Scope	Configuration File
application.log	All events, errors, and exceptions for a deployed application.	One log file for each application deployed.	orion-application.xml
global-application.log	All common events, errors, and exceptions related to applications.	All applications, including the default application.	application.xml
jms.log	All JMS events and errors.	JMS sub-system	jms.xml
rmi.log	All RMI events and errors.	RMI sub-system	rmi.xml
server.log	All events not associated with a particular sub-system or an application. This logs history of server startup, shutdown internal server errors.	server-wide	server.xml
web-access.log	Logs all accesses to the Web site.	Each Web site	default-web-site.xml

There are two types of log files:

- Oracle Diagnostic Logging (ODL) Log Files:** The messages logged in these files use an XML format that is read by the Enterprise Manager GUI. We recommend that you use this format for your logging, even though it is not the default, when you are using OC4J within Oracle9iAS.
- Text Log Files:** The messages logged in these files are not in XML and are simply for reading within any editor. This is the default. Normally, those who use OC4J standalone would benefit from viewing their log messages in a text format.

Oracle Diagnostic Logging (ODL) Log Files

The ODL log entries are each written out in XML format in its respective log file. Each XML message can be read through the Enterprise Manager GUI or through your own XML reader. The advantages for ODL logging is that the log files and the directory have a maximum limit. When the limit is reached, the log files are overwritten.

When you enable ODL logging, each new message goes into the current log file, named `log.xml`. When the log file is full—that is, the log file size maximum is reached—then it is copied to an archival log file, named `logN.xml`, where N is a number starting at one. When the last log file is full, the following occurs:

- The least recent log file is erased to provide space in the directory.
- The `log.xml` file is written to the latest `logN.xml` file, where N increments by one over the most recent log file.

Thus, your log files are constantly rolling over and do not encroach on your disk space.

Within each XML file listed in [Table 3–3](#), you enable ODL logging by either un-commenting an existing ODL configuration line or adding the ODL configuration line into the XML file, as follows:

- Add or un-comment the `<odl>` element within the `<log>` element in all XML files listed in [Table 3-3](#), except for the `default-web-site.xml` file.
- Add the `<odl-access-log>` element in the `default-web-site.xml` file.

The attributes that you can configure are:

- `path`: Path and folder name of the log folder for this area. You can use an absolute path or a path relative to where the configuration XML file exists, which is normally in the `j2ee/home/config` directory. This denotes where the log files will reside for the feature that the XML configuration file is concerned with. For example, modifying this element in the `server.xml` file denotes where the server log files are written.
- `max-file-size`: The maximum size in KB of each individual log file.
- `max-directory-size`: The maximum size of the directory in KB.

New files are created within the directory, until the maximum directory size is reached. Each log file is equal to or less than the maximum specified in the attributes.

Thus, to specify log files of 1000 KB and a maximum of 10,000 KB for the directory in the `ORACLE_HOME/j2ee/log/server` directory in the `server.xml` file, configure the following:

```
<log>
<odl path="../log/server/" max-file-size="1000" max-directory-size="10000" />
</log>
```

When OC4J is executing, all log messages that are server oriented are logged in the `ORACLE_HOME/j2ee/log/server` directory.

The XML message that is logged is of the following format:

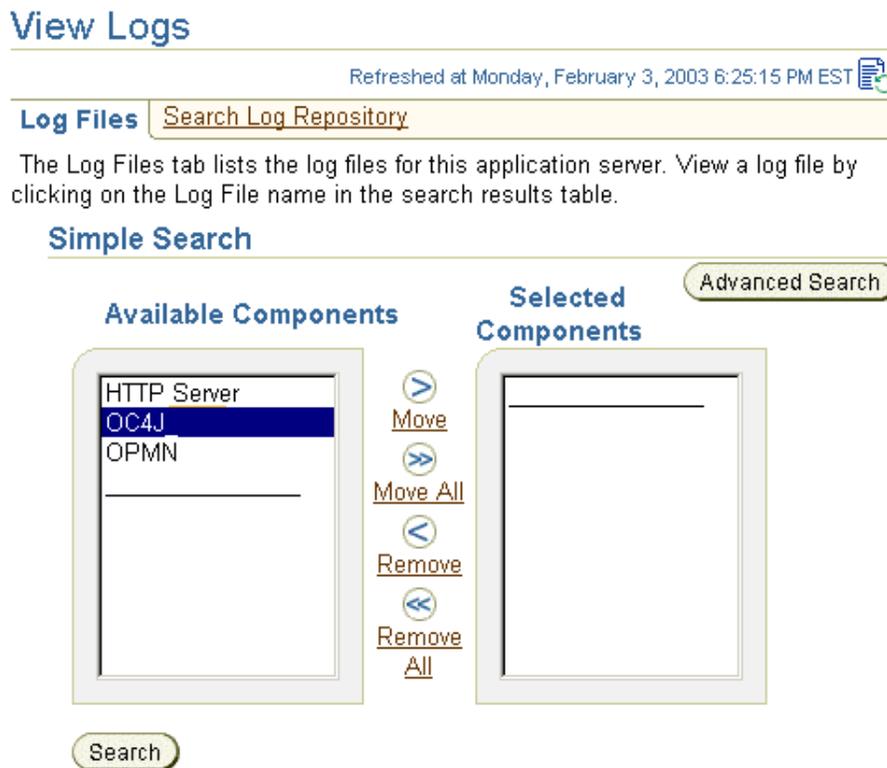
```
<MESSAGE>
<HEADER>
<TSTZ_ORIGINATING>2002-11-12T15:02:07.051-08:00</TSTZ_ORIGINATING>
<COMPONENT_ID>oc4j</COMPONENT_ID>
<MSG_TYPE TYPE="ERROR"></MSG_TYPE>
<MSG_LEVEL>1</MSG_LEVEL>
<HOST_ID>myhost</HOST_ID>
<HOST_NWADDR>001.11.22.33</HOST_NWADDR>
<PROCESS_ID>null-Thread[Orion Launcher,5,main]</PROCESS_ID>
<USER_ID>dpda</USER_ID>
</HEADER>
<PAYLOAD>
<MSG_TEXT>java.lang.NullPointerException at
com.evermind.server.ApplicationServer.setConfig(ApplicationServer.java:1070)
at com.evermind.server.ApplicationServerLauncher.run
(ApplicationServerLauncher.java:93) at java.lang.Thread.run(Unknown Source)
</MSG_TEXT>
</PAYLOAD>
</MESSAGE/>
```

You can have both the ODL and text logging turned on. To save on disk space, you should turn off one of these options. If you decide to enable ODL logging, turn off the text logging functionality by commenting out the `<file>` subelement of the `<log>` element for all XML files except the `default-web-site.xml` file. For the `default-web-site.xml` file, turn off the text logging by commenting out the `<access-log>` element.

You can view ODL log files by performing the following:

1. At the bottom of the Oracle Application Server Instance Home page, select **Logs**.
2. Select the OC4J Instances for which you want to view the log files.
 - a. Select the OC4J Instances in the Available Components column.
 - b. Select **Move** to transfer your select to the Selected Components column.
 - c. Select **Search** to view the log files for these OC4J Instances.

Figure 3–19 Viewing the Logs



See Also: "Managing Log Files" chapter in the *Oracle Application Server Administrator's Guide*

Text Log Files

Full text logging is still available in OC4J. Primarily, you should use text logging within OC4J standalone. It is easier to read within any editor, as it is not in XML format.

The text logging facility separates messages out in alignment with the XML files. However, instead of writing to multiple log files of the same size, all messages for that component are written into a single file. The text logging does not have any imposed limits or log rollover. Instead, the log files will continue to grow, unless you stop OC4J, remove the file, and restart OC4J to start the log files over. You can overrun your disk space if you do not monitor your log files. This is only feasible in a standalone, development environment.

Text messaging is the default and is configured in the same XML files as listed for ODL logging in [Table 3–3](#). Text messaging is enabled in the `<file>` subelement the `<log>` element of the XML files, except the `default-web-site.xml` file. For the `default-web-site.xml` file, the text messaging is enabled with the

<access-log> element. To turn off text messaging, eliminate or comment out the <file> or <access-log> element. If you do not remove this line and enable ODL logging, you will have both logging facilities turned on. The location and filename for text messaging does have defaults, as shown in [Table 3–4](#), but you can specify the location and filename within the path attribute of the <log> or <access-log> elements.

The following table shows the default location directory for text messaging log files of an OC4J process.

Table 3–4 OC4J Log File Locations

Log File	Default Location
application.log	<code>\$ORACLE_HOME/j2ee/<OC4J InstanceName>/application-deployments/<application-name>/<OC4J IslandName></code>
global-application.log	<code>\$ORACLE_HOME/j2ee/<OC4J InstanceName>/log/<OC4J IslandName>_<Process#></code>
jms.log	<code>\$ORACLE_HOME/j2ee/<OC4J InstanceName>/log/<OC4J IslandName>_<Process#></code>
rmi.log	<code>\$ORACLE_HOME/j2ee/<OC4J InstanceName>/log/<OC4J IslandName>_<Process#></code>
server.log	<code>\$ORACLE_HOME/j2ee/<OC4J InstanceName>/log/<OC4J IslandName>_<Process#></code>
web-access.log	The location is configurable from *-web-site.xml with the <access-log> element, as follows: <code><access-log path="..../log/http-web-access.log" /></code>
OPMN log file, named <OC4J_Instance_Name>.<Island_Name>.<Process#>	<code>\$ORACLE_HOME/opmn/logs/</code>

The location of all of the above log files can be specified, except the `web-access.log` file, using the <log> element in the respective configuration files. You can specify either absolute paths or paths relative to the `j2ee/home/config` directory. For example, specify the server log file in the `server.xml` configuration file, as follows:

```
<log>
<file path="..../log/my-server.log" />
</log>
```

You can also specify an absolute path for the location of the log file, as follows:

```
<log>
<file path="d:\log-files\my-server.log" />
</log>
```

Redirecting Standard Out and Standard Error

In an Oracle Application Server environment, the standard output and standard errors for OC4J are routed to the OPMN log for the OC4J instance. For example, if you have an OC4J instance called `OC4J_Demos`, and it has an island called `default_island`, and there is only one process in the island, the file that will contain the STDOUT and STDERR streams is `$ORACLE_HOME/opmn/logs/OC4J_Demos.default_island.1`

[Figure 3–20](#) shows how you specify the `-out` and `-err` parameters in the OC4J command-line options for your OC4J instance. When you specify these parameters

without specifying a specific directory, the log files are created in the `$J2EE_HOME/<OC4JInstanceName>_<IslandName>_<Process#>` file. For example, if you have an OC4J instance called `OC4J_Demos` and it has an island called `default_island`, then the log file will be created in the `$J2EE_HOME/OC4J_Demos_default_island_1`.

Note: In the preceding examples, it is assumed that there is only one OC4J process for the default island. If there are more processes for an island there are separate log files for each OC4J process.

In the Application Server Control, select Server Properties on the Administration page, which will bring you to the screen in [Figure 3–20](#).

Figure 3–20 EM Console to Modify Server Properties for an OC4J Instance

Multiple VM Configuration

✓ **TIP** If OC4J is running, newly added islands and associated processes will be automatically started.

Islands

Island ID	Number of Processes
default_island	1

Add Another Row

Ports

RMI Ports	3101-3200
JMS Ports	3201-3300
AJP Ports	3001-3100

Command Line Options

Java Executable	
OC4J Options	-properties -out oc4j.out -err oc4j.err
Java Options	-Dhttp.session.debug=true

OC4J Debugging

OC4J properties are configuration switches that can be set on the command-line. As shown in [Figure 3–20](#), the properties are prefaced with a `-D` in the Java Options line. OC4J provides several debug properties for generating additional information on the operations performed by the various sub-systems of OC4J. These debug properties can be set for a particular sub-system while starting up OC4J.

Note: Turning on excessive debug options can slow down the execution of your applications and use large amounts of disk space with the contents of the log files.

OC4J is started and managed by OPMN. You have to specify the Java system properties for your OC4J instance using Enterprise Manager as shown in [Figure 3–20](#). The supplied properties are saved in the OPMN configuration file. OPMN starts OC4J with these supplied properties when you shutdown and restart your OC4J Instance. See "[OC4J Command-Line Options and System Properties](#)" on page B-30 for OC4J general properties.

By default, the debug information is written to the OPMN log file for the OC4J Instance, which resides in the `$ORACLE_HOME/opmn/logs/` directory. For example, you have a OC4J instance named `OC4J_DEMOS` and there is only one island and the island has only process for this OC4J instance, then the debug information is logged in `$ORACLE_HOME/opmn/logs/OC4J_Demos.default_island.1`. However, if there is `-out` and `-err` command-line options specified with OC4J, then the debug information is redirected to the appropriate files.

The following tables provide useful debug options that available with OC4J. These debug options have two states either true or false. By default these are set to false. For a complete list of debug properties, see "[OC4J Command-Line Options and System Properties](#)" on page B-30.

Table 3–5 HTTP Debugging Options

HTTP Debugging	Description of Option
<code>http.session.debug</code>	Provides information about HTTP session events
<code>http.request.debug</code>	Provides information about each HTTP request
<code>http.cluster.debug</code>	Provides information about HTTP clustering events
<code>http.error.debug</code>	Prints all HTTP errors
<code>http.method.trace.allow</code>	Default: false. If true, turns on the <code>trace</code> HTTP method.

Table 3–6 JDBC Debugging Options

JDBC Debugging	Description of Option
<code>datasource.verbose</code>	Provides verbose information on creation of data source and connections using Data Sources and connections released to the pool, and so on,
<code>jdbc.debug</code>	Provides very verbose information when JDBC calls are made

Table 3–7 EJB Debugging Options

EJB Debugging	Description of Options
<code>ejb.cluster.debug</code>	Turns on EJB clustering debug messages

Table 3–8 RMI Debugging Options

RMI Debugging	Description of Options
<code>rmi.debug</code>	Prints RMI debug information
<code>rmi.verbose</code>	Provides very verbose information on RMI calls

Table 3–9 Web Services Debugging Options

Web Services Debugging	Description of Options
<code>ws.debug</code>	Turns on Web Services debugging

In addition to the specific sub-system switches, you can also start OC4J with a supplied verbosity level. The verbosity level is an integer between 1 and 10. The higher the verbosity level, the more information that is printed in the console. You specify the verbosity level with the `-verbosity OC4J` option in the Enterprise Manager in the OC4J command-line options section. The following examples show the output with and without verbosity:

Example 3–3 Error Messages Displayed Without Verbosity

```
D:\oc4j903\j2ee\home>java -jar oc4j.jar
Oracle Application Server Containers for J2EE initialized
```

Example 3–4 Error Messages Displayed With Verbosity Level of 10

```
D:\oc4j903\j2ee\home>java -jar oc4j.jar -verbosity 10
Application default (default) initialized...
Binding EJB work.ejb.WorkHours to work.ejb.WorkHours...
Application work (work) initialized...
Application serv23 (Servlet 2.3 New Features Demo) initialized...
Web-App default:defaultWebApp (0.0.0.0/0.0.0.0:8888) started...
Oracle Application Server Containers for J2EE initialized
```

Servlet Debugging Example

You deployed a Web application to OC4J that is having some problems with servlets. You are losing the client session when you use a pre-configured data source to make database connection. You want to know what OC4J is doing when the servlet is accessing the data source. In order to generate the debug information on HTTP Session and data source usage, you must set two debug options - `http.session.debug` and `datasource.verbose` to true.

Perform the following tasks

1. Logon to Enterprise Manager console as administrator.
2. Drill down to the OC4J Instance.
3. Select **Server Properties** for the OC4J Instance.
4. Enter Java Options as follows: `-Dhttp.session.debug=true`
`-Ddatasource.verbose=true`
5. Restart the OC4J instance.

After the OC4J Instance is restarted, you re-execute your servlet and see the following type of debug information in the standard output for the OC4J Instance:

```
DataSource logwriter activated... jdbc:oracle:thin:@localhost:1521/DEBU:
Started
jdbc:oracle:thin:@localhost:1521/DEBU: Started
Oracle Application Server Containers for J2EE initialized
Created session with id '4fa5eb1b9a564869a426e8544963754f' at Tue APR 23
16:22:56 PDT 2002, secure-only: false
Created new physical connection: XA XA Orion Pooled
jdbc:oracle:thin:@localhost:1521/DEBU
null: Connection XA XA Orion Pooled jdbc:oracle:thin:@localhost:1521/DEBU
allocated (Pool size: 0)
jdbc:oracle:thin:@localhost:1521/DEBU: Opened connection
Created new physical connection: Pooled
oracle.jdbc.driver.OracleConnection@5f18
Pooled jdbc:oracle:thin:@localhost:1521/DEBU: Connection Pooled
oracle.jdbc.driver.OracleConnection@5f1832 allocated (Pool size: 0)
```

```
Pooled jdbc:oracle:thin:@localhost:1521/DEBU: Releasing connection Pooled
oracle.jdbc.driver.OracleConnection@5f1832 to pool (Pool size: 1)
null: Releasing connection XA XA Orion Pooled
jdbc:oracle:thin:@localhost:1521/DEBU to pool (Pool size: 1)
Orion Pooled jdbc:oracle:thin:@localhost:1521/DEBU: Cache timeout, closing
connection (Pool size: 0)
com.evermind.sql.OrionCMTDataSource/default/jdbc/OracleDS: Cache timeout,
closing connection (Pool size: 0)
```

Remote Debugging Using Oracle JDeveloper

You can remotely debug your applications deployed in OC4J using any Java debugging facility that supports the JPDA (Java Platform Debugging Architecture). With OC4J embedded directly inside of the Oracle JDeveloper IDE, debugging both locally deployed and remote J2EE applications is simple. For more details, please see the Oracle JDeveloper documentation or read the "Remote Debugging How-To" document that is posted on OTN.

OC4J Clustering

This chapter discusses OC4J clustering, and provides instructions on how to manage OC4J processes and applications within a cluster.

It contains the following topics:

- [The OC4J Instance in a Cluster](#)
- [Instance-Specific Parameters](#)
- [OC4J Clustering Examples](#)
- [OC4J Cluster Configuration](#)

The full picture of clustering within Oracle Application Server is discussed in the Oracle Application Server High Availability Guide.

The OC4J Instance in a Cluster

The OC4J instance is the entity to which J2EE applications are deployed and configured. It defines how many OC4J processes exist within the application server and the configuration for these OC4J processes. The OC4J process is what executes the J2EE applications for the OC4J instance.

The OC4J instance has the following features:

- The configuration of the OC4J instance is valid for one or more OC4J executable processes. This way, you can duplicate the configuration for multiple OC4J processes by managing these processes in the OC4J instance construct. When you modify the cluster-wide configuration within the OC4J instance, the modifications are valid for all OC4J processes.
- Each OC4J instance can be configured with one or more OC4J processes.
- When you deploy an application to an OC4J instance, the OC4J instance deploys the application to all OC4J processes defined in the OC4J instance. The OC4J instance is also responsible for replicating the state of its applications.
- The number of OC4J processes is specific to each OC4J instance. This must be manually configured for each application server instance in the cluster. The OC4J process configuration provides flexibility to tune according to the specific hardware capabilities of the host. By default, each OC4J instance is instantiated with a single OC4J process.

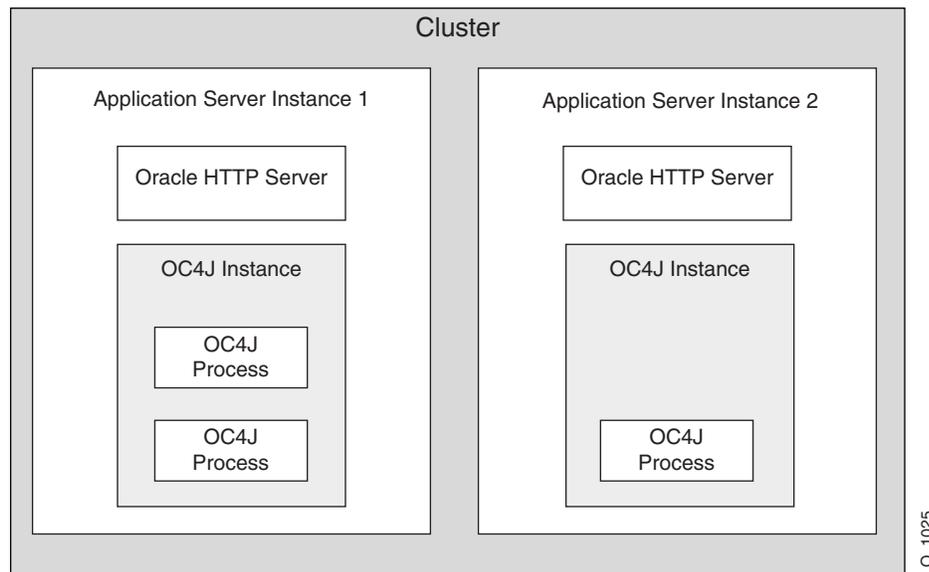
Within the application sever instance, you can configure multiple OC4J instances, each with its own number of OC4J processes. The advantage for this is for configuration management and application deployment for separate OC4J processes in your cluster.

Figure 4–1 demonstrates the home default OC4J instance. In the context of a cluster, the OC4J instance configuration is part of the cluster-wide configuration. Thus, the home instance, configured on the first application instance, is replicated on all other application server instances.

The number of processes in each home instance is an instance-specific parameter, so you must configure the home instance separately on each application server instance for the number of OC4J processes that exist on each application server instance.

Figure 4–1 shows that the home instance on application server instance 1 contains two OC4J processes; the home instance on application server instance 2 contains only one OC4J process. Each OC4J instance defaults to having one OC4J process.

Figure 4–1 OC4J Processes in a Cluster



The OC4J Process in a Cluster

The OC4J process is the JVM process that executes J2EE applications. Each OC4J process is associated to an OC4J instance and inherits its configuration from that OC4J instance. All applications deployed to an OC4J instance are started in all the OC4J processes associated to that OC4J instance.

You can define one or more OC4J processes within an OC4J instance, so that J2EE requests can be load balanced and have failover capabilities.

The configuration for the number of OC4J processes is instance-specific. Thus, you must configure each OC4J instance in each application server instance with the number of OC4J processes you want to start up for that OC4J instance. The default is one OC4J process.

Each host that you install the application server instances on has different capabilities. To maximize the hardware capabilities, configure the number of OC4J processes in each OC4J instance that will use these capabilities properly. For example, you can configure a single OC4J process on host A and five OC4J processes on host B.

When you define multiple OC4J processes, you enable the following:

- You can serve multiple users with multiple OC4J processes.

- You can provide failover if the state of the application is replicated across multiple OC4J processes.
- OHS provides load balancing for all OC4J processes in the OC4J instance. The OPMN component notifies each OHS when a new OC4J process is initiated. Thus, each OHS in the cluster knows of each OC4J process in the cluster.

Note: `mod_oc4j`, which sits between OHS and OC4J, is able to provide metric-based load balancing of requests to available OC4J instances. See the discussion of the `<metric-collector>` element in "[Elements in the server.xml File](#)" on page B-6 for instructions on implementing this feature.

Replicating Application State

The OC4J processes involved in the cluster can replicate application state to all OC4J processes. Once you configure replication, OC4J handles the propagation of the application state for you.

If one OC4J process fails, then another OC4J process—which has had the application state replicated to it—takes over the application request. When an OC4J process fails during a stateful request, the OHS forwards the request in the following order:

1. If another OC4J process is active within the same application server instance, OHS forwards the request to this process.
2. Otherwise, OHS forwards the state request to an OC4J process in another application server instance in the cluster.

There are two types of failure that you want to protect against: software failure and hardware failure.

Failure Type	Avoidance Technique
Software failure occurs when the OC4J process fails.	Multiple OC4J processes in the same OC4J instance. When one OC4J process fails, the OHS forwards the request to another OC4J process in the same OC4J instance.
Hardware failure occurs when the host goes down.	OC4J processes in the cluster configured on separate hosts. When the first host dies, the OC4J process on another host can take over the request. This requires that you have installed an application server instance on another host, which is a part of the cluster, and the OC4J instance has at least one OC4J process.

Islands

An island is a logical grouping of OC4J processes that allows you to determine which OC4J processes will replicate state.

In each OC4J instance, you can have more than one OC4J process. If we consider state replication in a situation where all OC4J processes tried to replicate state, then the CPU load can significantly increase. To avoid a performance degradation, the OC4J instance enables you to subgroup your OC4J processes. The subgroup is called an island.

To ensure that the CPU load is partitioned among the processes, the OC4J processes of an OC4J instance can be partitioned into islands. The state for application requests is replicated only to OC4J processes that are grouped within the same island. All applications are still deployed to all OC4J processes in the OC4J instance. The only

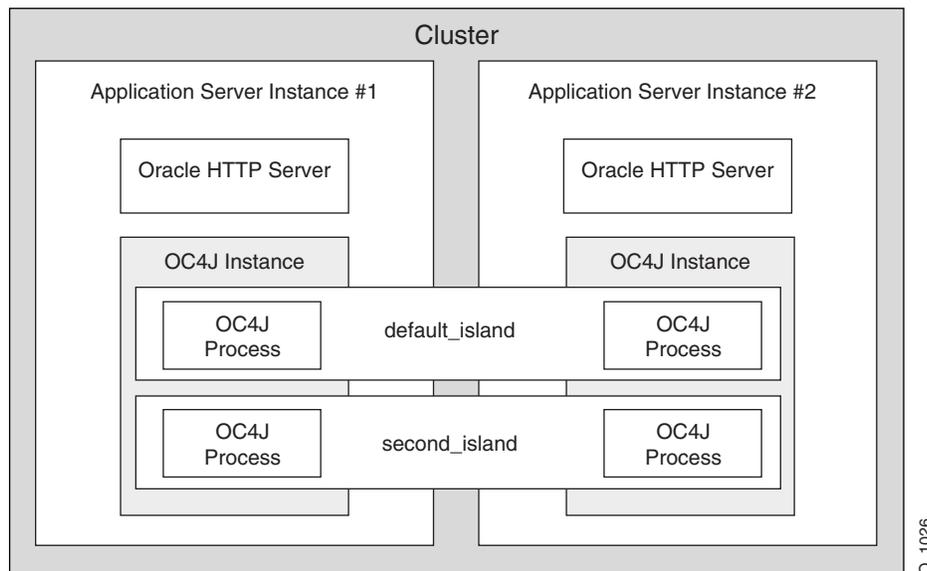
difference is that the state for these applications is confined to only a subset of these OC4J processes.

The island configuration is instance-specific. The name of the island must be identical in each OC4J instance, where you want the island to exist. When you configure the number of OC4J processes on each application server instance, you can also subgroup them into separate islands. The OC4J processes are grouped across application server instances by the name of the island. Thus, the application state is replicated to all OC4J processes within the island of the same name spanning application server instances.

The grouping of OC4J processes for the state replication is different for EJB applications than for Web applications. Web applications replicate state within the island sub-grouping. EJB applications replicate state between all OC4J processes in the OC4J instance and do not use the island sub-grouping.

Figure 4–2 demonstrates OC4J processes in islands within the cluster. Two islands are configured in the home instance: `default_island` and `second_island`. One OC4J process is configured in each island on each application server instance. The OC4J islands, designated within the shaded area, span application server instances.

Figure 4–2 Island Description



J2EE Applications Involved in a Cluster

J2EE applications are deployed in all cases to the OC4J instance—whether the application server instance is included in a cluster or not. However, when the application is deployed to an OC4J instance that is in a cluster, certain configuration details must be accomplished:

- Multicast host and port—The state of the applications is replicated from one OC4J process to another over a multicast address. In the case of an EJB application, you must also specify a username and password. You can either accept the defaults for the multicast address or configure it through the Enterprise Manager.
- State replication request—You request state replication for all applications through the Enterprise Manager.

- XML deployment descriptor elements—Both Web and EJB applications require an additional configuration in their respective XML deployment descriptors.
- Island definition—Web applications use the island sub-grouping for its state replication. EJB applications ignore the island sub-grouping and use all OC4J processes for its state replication.

Instance-Specific Parameters

The following parameters are not replicated across the cluster; thus, each must be set at the OC4J instance level on each application server.

- Islands and number of OC4J processes—While you want to keep the names of the islands consistent across the application server instances, the definition of the islands and the number of OC4J processes is configured independently. The host on which you install each application server instance has different capabilities. On each host, you can tune the number of OC4J processes to match the host capabilities. Remember that the state is replicated in islands across application boundaries. So the island names must be the same in each OC4J instance.
- Port numbers—The RMI, JMS, and AJP port numbers can be different for each host.
- Command line options—The command line options you use can be different for each host.

OC4J Clustering Examples

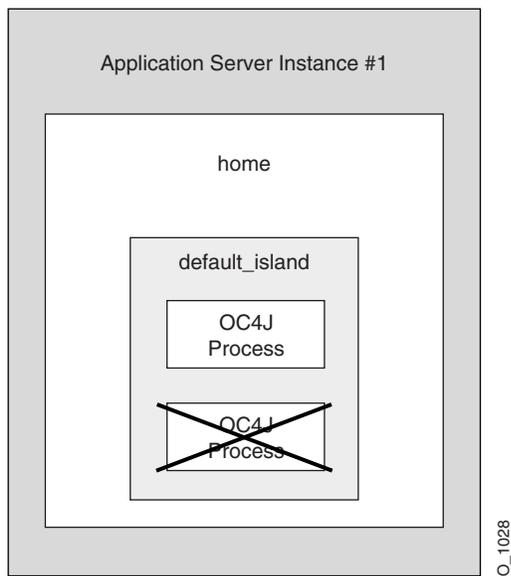
No matter how many application server instances you add within the cluster, the cluster-wide configuration is replicated within the cluster. You control protecting against software and hardware failure with how you configure island and OC4J processes, which are instance-specific parameters.

Software Failure

Suppose you configure more than one OC4J process within your OC4J instance, then if one of these processes fails, another process can take over the work load of the failed process. [Figure 4-3](#) shows application server instance 1, which is involved in the cluster. Within this application server instance, there are two OC4J processes defined in the `default_island` in the home instance. If the first OC4J process fails, the other can pick up the work load.

Both of these OC4J processes are on the same host; so, if the host goes down, both OC4J processes fail and the client cannot continue processing.

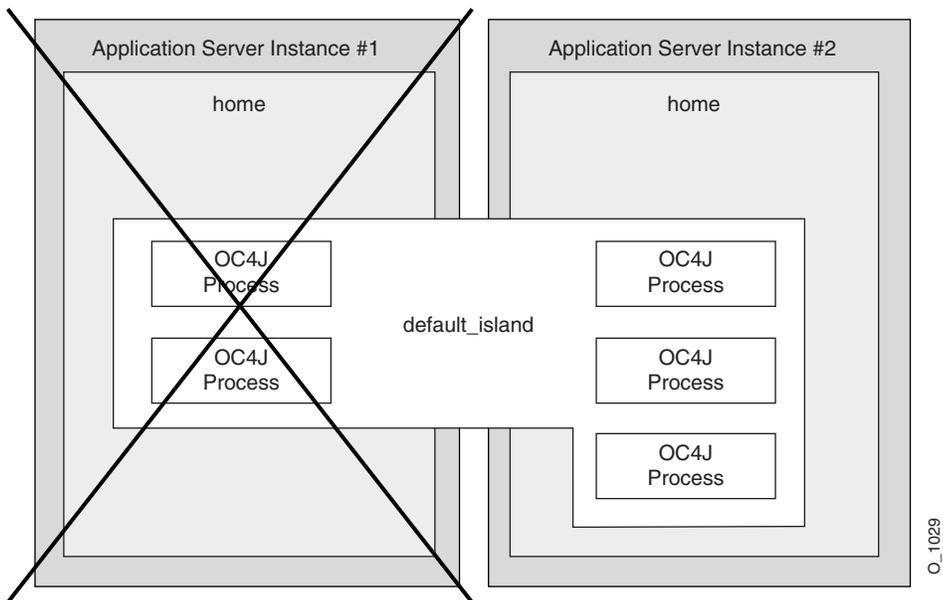
Figure 4-3 Software Failure Demonstration



Hardware Failure

To protect against hardware failure, you must configure OC4J processes in the same OC4J instance across hosts. [Figure 4-4](#) shows home instance in application server instance 1 and 2. Within the default_island, two OC4J processes are configured on application server instance 1 and three are configured in application server instance 2. If a client is interacting with one of the OC4J processes in application server 1, which terminates abnormally, the client is redirected automatically to one of the OC4J processes in the default_island in application server 2. Thus, your client is protected against hardware failure.

Figure 4-4 Hardware Failure Demonstration



State Replication

If the client is a stateful application, then the state is replicated only within the same island. In the previous example, there is only a single island, so the state of the application would be preserved.

To enhance your performance, you want to divide up state replication among islands. However, you must also protect for hardware and software failure within these islands.

The optimal method of protecting against software and hardware failure, while maintaining state with the least number of OC4J processes, is to configure at least one OC4J process on more than one host in the same island. For example, if you have application server instance 1 and 2, within the `home` instance, you configure one OC4J process in the `default_island` on each application server instance. Thus, you are protected against hardware and software failure and your client maintains state if either failure occurs.

- If one of the OC4J processes fails, then the client request is redirected to the other OC4J process in the island. The state is preserved and the client does not notice any irregularity.
- If application server 1 terminates abnormally, then the client is redirected to the OC4J process in the `default_island` on application server 2. The state is preserved and the client does not notice any irregularity.

As demand increases, you will configure more OC4J processes. To guard against a performance slowdown, separate your OC4J processes into separate islands. For example, if fifteen OC4J processes utilize the hardware efficiently on the two hosts and serve the client demand appropriately, then you could divide these processes into at least two islands. The following shows the fifteen OC4J processes grouped into three islands:

Island Names	Application Server 1	Application Server 2
<code>default_island</code>	two	three
<code>second_island</code>	two	three
<code>third_island</code>	three	two

- The host where application server 1 is installed can handle seven OC4J processes; the host where application server 2 is installed can handle eight OC4J processes.
- Each island contains at least one OC4J process in each island across hosts to protect against software and hardware failure.
- The performance is maximized by dividing up the state replication across three islands.

OC4J Cluster Configuration

The following sections describe how to manage OC4J elements in the cluster: EJBs and Servlets. For directions on how to create and modify Oracle Application Server clusters, see the Oracle Application Server High Availability Guide.

- [OC4J Instance Configuration](#)
- [Configuring OC4J Instance-Specific Parameters](#)

Note: As an alternative to using Enterprise Manager, you can create a cluster, add application server instances to the cluster, and manage the cluster using the DCM command-line tool. See the Distributed Configuration Management Administrator's Guide for information on the DCM command-line tool.

OC4J Instance Configuration

The following sections describe how to configure your OC4J Instance for clustering:

- [Configuring Islands and Processes](#)
- [Configuring Web Application State Replication](#)
- [Configuring EJB Application State Replication](#)

Configuring Islands and Processes

To modify the islands and the number of processes each island contains, do the following:

1. Select the Administration page off the OC4J Home Page.
2. Select Server Properties in the Instance Properties column.
3. Scroll down to the Multiple VM Configuration section. This section defines the islands and the number of OC4J processes that should be started on this application server instance in each island.
4. Create any islands for this OC4J instance within the cluster by clicking **Add Another Row**. You can supply a name for each island within the Island ID field. You can designate how many OC4J processes should be started within each island by the number configured in the Number of Processes field.

[Figure 4–5](#) displays the Multiple VM Configuration section.

Figure 4–5 Island and Process Configuration

Multiple VM Configuration

Islands

Island ID	Number of Processes	Related Links
default_island	1	Virtual Machine Metrics
Add Another Row		

Ports

RMI Ports	3101-3200
JMS Ports	3201-3300
AJP Ports	3000-3100

Command Line Options

Java Executable	
OC4J Options	
Java Options	

Configuring Web Application State Replication

Configuring state replication for stateful applications is different for Web applications than for EJB applications. To configure state replication for Web applications, do the following:

1. Select the Administration page off of the OC4J Home Page.
2. Select Replication Properties in the Instance Properties column.
3. Scroll down to the Web Applications section. [Figure 4–6](#) shows this section.
4. Select the Replicate session state checkbox.
5. Optionally, you can provide the multicast host IP address and port number. If you do not provide the host and port for the multicast address, it defaults to host IP address 230.230.0.1 and port number 9127. The host IP address must be between 224.0.0.2 through 239.255.255.255. Do not use the same multicast address for both HTTP and EJB multicast addresses.

Note: When choosing a multicast address, ensure that it does not collide with the addresses listed in <http://www.iana.org/assignments/multicast-addresses>. Also, if the low order 23 bits of an address is the same as the local network control block (224.0.0.0 - 224.0.0.255), then a collision may occur. To avoid this, provide an address that does not have the same bits in the lower 23 bits of the address as the addresses in this range.

6. Add the `<distributable/>` tag to all `web.xml` files in all Web applications. If the Web application is serializable, you must add this tag to the `web.xml` file.

The following shows an example of this tag added to `web.xml`:

```

<web-app>
  <distributable/>
  <servlet>
    ...
  </servlet>
</web-app>

```

Figure 4–6 Web State Replication Configuration

Replication Properties

Refreshed at Tuesday, March 19, 2002 2:29:33 PM EST 

Web Applications

 **TIP** Setting session state replication here will enable session state replication for all web applications. The load-on-startup property will be automatically set to true for all web modules.

Replicate session state

Multicast Host (IP)

Multicast Port

EJB Applications

 **TIP** EJB applications replicate state between all OC4J processes in the OC4J instance.

Replicate State

Multicast Host (IP)

Multicast Port

Username

Password

Revert

Apply

Configuring EJB Application State Replication

To create an EJB cluster, you specify the OC4J instances that are to be involved in the cluster, configure each of them with the same multicast address, username, and password, and deploy the EJB, which is to be clustered, to each of the nodes in the cluster.

Unlike HTTP clustering, EJBs involved in a cluster cannot be sub-grouped in an island. Instead, all EJBs within the cluster are in one group. Also, only session beans are clustered.

The state of all beans is replicated at the end of every method call to all nodes in the cluster using a multicast topic. Each node included in the EJB cluster is configured to use the same multicast address.

The concepts for understanding how EJB object state is replicated within a cluster are described in the *Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide*. To configure EJB replication, you configure the following:

1. Select the Administration page off of the OC4J Home Page.
2. Select Replication Properties in the Instance Properties column.

3. Scroll down to the EJB Applications section. [Figure 4-7](#) shows this section.
4. Select the Replicate session state checkbox.
5. Provide the username and password, which is used to authenticate itself to other hosts in the cluster. If the username and password are different for other hosts in the cluster, they will fail to communicate. You can have multiple username and password combinations within a multicast address. Those with the same username/password combinations will be considered a unique cluster.
6. Optionally, you can provide the multicast host IP address and port number. If you do not provide the host and port for the multicast address, it defaults to host IP address 230.230.0.1 and port number 23791. The host IP address must be between 224.0.0.2 through 239.255.255.255. Do not use the same multicast address for both HTTP and EJB multicast addresses.

Note: When choosing a multicast address, ensure that it does not collide with the addresses listed in <http://www.iana.org/assignments/multicast-addresses>. Also, if the low order 23 bits of an address is the same as the local network control block (224.0.0.0 - 224.0.0.255), then a collision may occur. To avoid this, provide an address that does not have the same bits in the lower 23 bits of the address as the addresses in this range.

7. Provide the host name where the OC4J Instance resides in the RMI Server Host field.
8. Configure the type of EJB replication within the `orion-ejb-jar.xml` file within the JAR file. See ["Stateful Session Bean Replication Configuration in the Application JAR"](#) on page 4-12 for full details. You can configure these within the `orion-ejb-jar.xml` file before deployment or add this through the Enterprise Manager screens after deployment. If you add this after deployment, drill down to the JAR file from the application page.

Figure 4-7 EJB State Replication Configuration

Replication Properties

Refreshed at Tuesday, March 19, 2002 2:29:33 PM EST 

Web Applications

TIP Setting session state replication here will enable session state replication for all web applications. The load-on-startup property will be automatically set to true for all web modules.

Replicate session state

Multicast Host (IP)

Multicast Port

EJB Applications

TIP EJB applications replicate state between all OC4J processes in the OC4J instance.

Replicate State

Multicast Host (IP)

Multicast Port

Username

Password

Stateful Session Bean Replication Configuration in the Application JAR For stateful session beans, you may have you modify the `orion-ejb-jar.xml` file to add the state replication configuration. Since you configure the replication type for the stateful session bean within the bean deployment descriptor, each bean can use a different type of replication.

Stateful session beans require state to be replicated among nodes. In fact, stateful session beans must send all their state between the nodes, which can have a noticeable effect on performance. Thus, the following replication modes are available to you to decide on how to manage the performance cost of replication:

JVM Termination Replication The state of the stateful session bean is replicated to only one other node in the cluster (with the same multicast address) when the JVM is terminating. Since this uses JDK 1.3 shutdown hooks, you must use JVM version 1.3 or later. This is the most performant option, because the state is replicated only once. However, it is not very reliable for the following reasons:

- Your state is not replicated if the power is shut off unexpectedly.
- The state of the bean exists only on a single node at any time; the depth of failure is equal to one node.

Set the replication attribute of the `<session-deployment>` tag in the `orion-ejb-jar.xml` file to "VMTermination". This is shown below:

```
<session-deployment replication="VMTermination" .../>
```

End of Call Replication The state of the stateful session bean is replicated to all nodes in the cluster (with the same multicast address) at the end of each EJB method call. If the

node loses power, then the state has already been replicated. The JVM termination replication mode does not guarantee state replication in the case of lost power.

Set the `replication` attribute of the `<session-deployment>` tag in the `orion-ejb-jar.xml` file to `"endOfCall"`. This is shown below:

```
<session-deployment replication="EndOfCall" .../>
```

Configuring OC4J Instance-Specific Parameters

The manageability feature of the cluster causes the configuration to be replicated across all application server instances in the cluster, which is defined as a cluster-wide configuration. However, there are certain parameters where it is necessary to configure them separately on each OC4J instance. These parameters are referred to as OC4J instance-specific.

The following parameters are OC4J instance-specific parameters, which are not replicated across the cluster. You must modify these parameters on each application server instance.

The following are instance-specific parameters within each OC4J instance:

- islands
- number of OC4J processes
- port numbers
- command-line options

All other parameters are part of the cluster-wide parameters, which are replicated across the cluster.

[Figure 4-8](#) shows the section where these parameters are modified. These sections are located in the Server Properties on the Administration page.

Figure 4–8 Non-Replicated Configuration

Multiple VM Configuration

Islands

Island ID	Number of Processes	Related Links
default_island	1	Virtual Machine Metrics
<input type="button" value="Add Another Row"/>		

Ports

RMI Ports	3101-3200
JMS Ports	3201-3300
AJP Ports	3000-3100

Command Line Options

Java Executable	
OC4J Options	
Java Options	

Troubleshooting OC4J

This appendix describes common problems that you may encounter when using OC4J and explains how to resolve them. It includes the following topics:

- [Problems and Solutions](#)
- [Need More Help?](#)

Problems and Solutions

This section describes common problems and solutions. It contains the following topics:

- [Unable to Start OC4J When Using JDK 1.3](#)
- [Unable to Restart OC4J After Abnormal Termination When OracleAS JMS is Active](#)
- [Stateful Replication Not Consistent Across OC4J instances](#)
- [Using A Non-Certified Version of the JDK for OC4J Only](#)
- [java.lang.OutOfMemory Errors Thrown When Running OC4J](#)
- [Connection Timeouts Through a Stateful Firewall Affect System Performance](#)
- [OPMN-Managed OC4J Unable to Access EJB Resources Via the Default RMI Port](#)
- [Application Performance Impacted by JVM Garbage Collection Pauses](#)
- [Invalid or Unneeded Library Elements Degrade Performance](#)
- [JSP Error: Tag Not Registered](#)
- [JSP Error: Zero-Length Class File](#)
- [JSP Error: Illegal use of <when>-style tag without <choose> as its direct parent](#)

Unable to Start OC4J When Using JDK 1.3

Problem

OC4J fails to start when using JDK 1.3.

Solution

The failure to start is caused by a logging implementation dependency issue. The solution to this problem is to remove or comment out the following entry in `ORACLE_HOME/j2ee/home/config/server.xml`:

```
<j2ee-logging-config path="./j2ee-logging.xml" />
```

Unable to Restart OC4J After Abnormal Termination When OracleAS JMS is Active

Problem

When persistence is enabled in OracleAS JMS, the JMS server creates persistent queues/topics. It also creates lock files (.lock) associated with these queues/topics in the /persistence directory. If the JVM is terminated abnormally, such as with `kill -9`, the lock files are not deleted. This creates a condition in which OC4J cannot be restarted.

Solution

Manually delete all .lock files from the /persistence directory.

Stateful Replication Not Consistent Across OC4J instances

Problem

The common scenario is that failover is seen from OC4J instance A to instance B, but not back again from B to A.

Solution

OC4J does not require stateful replication to be set up globally for all applications, but instead allows each Web module to configure replication through the `<cluster-config>` element in its `orion-web.xml` descriptor file. Ensure that this element is populated correctly in each Web module's descriptor.

Note that OPMN-managed OC4J only supports clustering at the global level, and not at the application/module level.

Using A Non-Certified Version of the JDK for OC4J Only

Problem

In this scenario, you wish to use a later version of the JDK with OC4J than the version certified for use with all Oracle Application Server components. However, using a later version of the JDK globally for all components increases the risk of breaking certification.

Solution

To use the later JDK version with OC4J only, specify its location in the `<java-bin>` element in the `opmn.xml` configuration file. For example:

```
<module-data>
  <category id="start-parameters">
    <data id="java-bin" value="/myjavalocation/jdk/bin/java"/>
  </category>
</module-data>
```

java.lang.OutOfMemory Errors Thrown When Running OC4J

Problem

This error indicates that the heap size of the Java instance is lower than the memory required by applications running within OC4J.

Solution

Increase the heap size by setting `-Xmx` to the desired amount of memory in the `<java-option>` element in `opmn.xml`:

```
<module-data>
  <category id="start-parameters">
    <data id="java-options" value="-Xmx256M" />
  </category>
</module-data>
```

Alternatively, you can set a system property at OC4J startup:

```
java -Xmx256M -jar oc4j.jar
```

If running under Unix/Linux, verify that `ulimit` settings allow the JVM process to allocate this much memory.

Connection Timeouts Through a Stateful Firewall Affect System Performance**Problem**

To improve performance the `mod_oc4j` component in each Oracle HTTP Server process maintains open TCP connections to the AJP port within each OC4J instance it sends requests to.

In situations where a firewall exists between OHS and OC4J, packages sent via AJP are rejected if the connections can be idle for periods in excess of the inactivity timeout of stateful firewalls.

However, the AJP socket is not closed; as long as the socket remains open, the worker thread is tied to it and is never returned to the thread pool. OC4J will continue to create more threads, and will eventually exhaust system resources.

Solution

The OHS TCP connection must be kept "alive" to avoid firewall timeout issues. This can be accomplished using a combination of OC4J configuration parameters and Apache runtime properties.

Set the following parameters in the `httpd.conf` or `mod_oc4j.conf` configuration files. Note that the value of `Oc4jConnTimeout` sets the length of inactivity, in seconds, before the session is considered inactive.

```
Oc4jUserKeepalive on
```

```
Oc4jConnTimeout 12000 (or a similar value)
```

Also set the following AJP property at OC4J startup to enable OC4J to close AJP sockets in the event that a connection between OHS and OC4J is dropped due to a firewall timeout:

```
ajp.keepalive=true
```

For example:

```
java -Dajp.keepalive=true -jar oc4j.jar
```

OPMN-Managed OC4J Unable to Access EJB Resources Via the Default RMI Port

Problem

OC4J cannot access EJB resources via the default RMI port when running as a component of Oracle Application Server.

Solution

The most common cause is that a user more familiar with Standalone OC4J is reading the RMI port from `rmi.xml`, unaware that the value specified in this file is not used in an OPMN-managed Oracle Application Server environment.

OPMN-managed OC4J instances use dynamic RMI port assignments. The port value ranges are specified in the `<port>` element in `opmn.xml` or are determined using dynamic `opmn:ormi` lookup from the application client.

See the *Oracle Process Manager and Notification Server Administrator's Guide* for more information.

Application Performance Impacted by JVM Garbage Collection Pauses

Problem

An application running on OC4J appears unresponsive, with simple requests experiencing noticeable delays. The cause is that the JVM has crossed the low memory threshold and is running a full garbage collection to free up memory.

Solution

Consider using the *incremental low pause collector*, which avoids long major garbage collection pauses by doing portions of the major collection work at each minor collection. This collector (also known as the *train collector*) collects portions of the tenured generation - a memory pool holding objects that are typically collected in a major collection - at each minor collection. The result is shorter pauses spread over many minor collections.

Note that the incremental collector is even slower than the default tenured generation collector when considering overall throughput.

To use the incremental collector, the `-Xincgc` option must be passed in on the Java command line at application startup. Set the initial and maximum size of the young generation (object pool) to the same value using the `XX:NewSize` and `-XX:MaxNewSize` options. Set the initial and the maximum Java heap sizes to the same value using the `-Xms` and `-Xmx` options.

For example, to use this collector with a server with 1GB of physical memory:

```
java -server -Xincgc -XX:NewSize=64m -XX:MaxNewSize=64m -Xms512m -Xmx512m
```

For more information on garbage collection tuning, read *"Tuning Garbage Collection with the 1.4.2 Java™ Virtual Machine"* which is available at <http://java.sun.com/docs/hotspot/gc1.4.2/>

Invalid or Unneeded Library Elements Degrade Performance

Problem

If the OC4J process memory is growing consistently during program execution, then you may have references to invalid symbolic links in your `application.xml` file.

This problem is usually characterized by a growth in the C heap and not a growth in Java object memory, as one would see with a more traditional Java object memory leak. OC4J loads all resources defined in the `application.xml` file. If these links are invalid, then the C heap continues to grow, causing OC4J to run out of memory.

Solution

Ensure that all symbolic links are valid in `application.xml`, and restart OC4J.

In addition, keep the number of JAR files OC4J is configured to load to a minimum. Eliminate all unused JAR files from the configuration and from the directories OC4J is configured to search. OC4J searches all JAR files for classes and resources, thereby causing the file cache to use extra memory and processor time.

You can control the loading more precisely if your `<library>` elements in the `application.xml` file point to the individual JAR and ZIP files that are needed, instead of to the directories where they reside.

JSP Error: Tag Not Registered

Problem

This error occurs when a JSP attempts to call a tag that cannot be found within the OC4J server. The problem typically arises when one or more additional "well-known tag library locations" have been incompletely defined within OC4J.

Solution

Defining a well-known tag library location is a two-step process:

- The directory is defined in the `jsp-taglib-locations` attribute of the `<orion-web-app>` element in the `global-web-application.xml` file; and
- The directory is added to the `path` attribute of the `<library>` element in the `application.xml`.

The error typically indicates that the second step was not completed.

Alternatively, you can copy the JAR file containing the tag library to the default well-known tag library location, which is `ORACLE_HOME/j2ee/home/jsp/lib/taglib/`.

JSP Error: Zero-Length Class File

Problem

This error occurs when OC4J attempts to serve a JSP that failed to compile into a Java class. The cause is that the Java compiler either could not be loaded or ran out of memory, resulting in a 0 byte `.class` file.

Solution

Delete the 0 byte `.class` file. The class will be compiled the next time the JSP is requested.

JSP Error: Illegal use of <when>-style tag without <choose> as its direct parent

Problem

This error occurs when OC4J fails to serve a requested JSP that includes a JSP Standard Tag Library (JSTL) tag—in this case, the `<choose>` tag. The likely cause is that more than one version of the JSTL exists within the OC4J instance.

Solution

Delete the version of the JSTL installed by default with OC4J. This library is packaged as the `standard.jar` file in the `ORACLE_HOME/j2ee/home/jsp/lib/taglib` directory.

Need More Help?

You can search for additional solutions on the following Oracle support-oriented Web sites:

- Oracle Application Server Release Notes, available on the Oracle Technology Network at <http://www.oracle.com/technology/documentation/index.html>
- Oracle MetaLink, available at <http://metalink.oracle.com>

If you still cannot find a solution for the problem you are facing, please log a service request.

Additional Information

This appendix contains complete information about the following topics:

- [Description of XML File Contents](#)
- [Elements in the server.xml File](#)
- [Elements in the application.xml File](#)
- [Elements in the orion-application.xml File](#)
- [Elements in the application-client.xml File](#)
- [Elements in the orion-application-client.xml File](#)
- [Configuration and Deployment Examples](#)
- [OC4J Command-Line Options and System Properties](#)

Most of these sections discuss how to modify your XML files. Modify all XML files only through Enterprise Manager. Do not modify XML files on a single node.

Description of XML File Contents

OC4J uses configuration and deployment XML files. The following sections describe each of these files and their function.

OC4J Configuration XML Files

This section describes the following XML files, which are necessary for OC4J configuration:

- [server.xml](#)
- [default-web-site.xml](#)
- [jaza-data.xml](#)
- [principals.xml](#)
- [data-sources.xml](#)
- [jms.xml](#)
- [rmi.xml](#)

server.xml

This file contains the configuration for the application server. The `server.xml` file is the root configuration file—it contains references to other configuration files. In this file, specify the following:

- Library path, which is located in the application deployment descriptor
- Global application, global Web application, and default Web site served
- Maximum number of HTTP connections the server allows
- Logging settings
- Java compiler settings
- Cluster ID
- Transaction time-out
- SMTP host
- Location of the `data-sources.xml` configuration
- Location of the configuration for JMS and RMI
- Location of the default and additional Web sites

Specify these locations by adding entries that list the location of the Web site configuration files. You can have multiple Web sites. The `default-web-site.xml` file defines a default Web site; therefore, there is only one of these XML files. All other Web sites are defined in `web-site.xml` configuration files. Register each Web site within the `server.xml` file, as follows:

```
<web-site path="./default-web-site.xml" />
<web-site path="./another-web-site.xml" />
```

Note: The path that is designated is relative to the `config/` directory.

- Pointers to all applications for the container to deploy and execute
- Specify the applications that run on the container in the `server.xml` file. You can have as many application directories as you want, and they do not have to be located under the OC4J installation directory.

default-web-site.xml

This file contains the configuration for a Web site. In the `default-web-site.xml` file, specify the following:

- Host name or IP address, virtual host settings for this site, listener ports, and security using SSL
- Default Web application for this site
- Additional Web applications for this site
- Access-log format
- Settings for user Web applications (for `/~user/ sites`)
- SSL configuration
- Restrict access to the site from one or more hosts

jazn-data.xml

This file contains security information for the OC4J server. It defines the user and group configuration for employing the default `JAZNUserManager`. In the `jazn-data.xml` file, specify the following:

- Username and password for the client-admin console
- Name and description of users/groups, and real name and password for users

principals.xml

This file contains security information for the OC4J server. It defines the user and group configuration for employing the `XMLUserManager`, which is no longer the default security manager. In the `principals.xml` file, specify the following:

- Username and password for the client-admin console
- Name and description of users/groups, and real name and password for users
- Optional X.509 certificates for users

data-sources.xml

This file contains configuration for the data sources that are used. In addition, it contains information on how to retrieve JDBC connections. In the `data-sources.xml` file, specify the following:

- JDBC driver
- JDBC URL
- JNDI paths to which to bind the data source
- User/password for the data source
- Database schema to use
- Inactivity time-out
- Thread policy
- Garbage collection granularity
- Maximum number of connections allowed to the database

Note: Database schemas are used to make auto-generated SQL work with different database systems. OC4J contains an XML file format for specifying properties, such as type-mappings and reserved words. OC4J comes with database schemas for MS SQL Server/MS Access, Oracle, and Sybase. You can edit these or make new schemas for your DBMS.

jms.xml

This file contains the configuration for the OC4J Java Message Service (JMS) implementation. In the `jms.xml` file, specify the following:

- Host name or IP address, and port number to which the JMS server binds
- Settings for queues and topics to be bound in the JNDI tree
- Log settings

rmi.xml

This file contains configuration for the Remote Method Invocation (RMI) system. It contains the setting for the RMI listener, which provides remote access for EJBs. In the `rmi.xml` file, specify the following:

- Host name or IP address, and port number to which the RMI server binds
- Remote servers to which to communicate
- Clustering settings
- Log settings

J2EE Deployment XML Files

The OC4J-specific deployment XML files contain deployment information for different components. If you do not create the OC4J-specific files, they are automatically generated when the application is deployed. You can edit OC4J-specific deployment XML files manually. OC4J uses these files to map environment entries, resources references, and security-roles to actual deployment-specific values.

This section describes the following XML files necessary for J2EE application deployment:

- [The J2EE application.xml File](#)
- [The OC4J-Specific orion-application.xml File](#)
- [The J2EE ejb-jar.xml File](#)
- [The OC4J-Specific orion-ejb-jar.xml File](#)
- [The J2EE web.xml File](#)
- [The OC4J-Specific orion-web.xml File](#)
- [The J2EE application-client.xml File](#)
- [The OC4J-Specific orion-application-client.xml File](#)

The J2EE application.xml File

This file identifies the Web or EJB applications that are contained within the J2EE application. See ["Elements in the application.xml File"](#) on page B-14 for a list of the elements.

The OC4J-Specific orion-application.xml File

This file configures the global application. In the `orion-application.xml` file, specify the following:

- Add files to the library path
- Whether to auto-create and auto-delete tables for CMP beans
- Which default data source to use with CMP beans
- Security role mappings
- Which user manager is the default for security
- JNDI namespace-access rules (authorization)

See ["Elements in the orion-application.xml File"](#) on page B-16 for a list of the elements.

The J2EE ejb-jar.xml File

This file defines the deployment parameters for the EJBs in this JAR file. See the Sun Microsystems EJB specification for a description of these elements.

The OC4J-Specific orion-ejb-jar.xml File

This file is the OC4J-specific deployment descriptor for EJBs. In the `orion-ejb-jar.xml` file, specify the following:

- Time-out settings
- Transaction retry settings
- Session persistence settings
- Transaction isolation settings
- CMP mappings
- OR mappings
- Finder method specifications
- JNDI mappings
- Minimum and maximum instance pool settings
- resource reference mappings

See the appendix in the *Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide* for description of the elements.

The J2EE web.xml File

This file contains deployment information about the servlets and JSPs in this application. See the Sun Microsystems specifications for a description of these elements.

The OC4J-Specific orion-web.xml File

This is the OC4J-specific deployment descriptor for mapping Web settings. This XML file contains the following:

- Auto-reloading (including modification-check time-interval)
- Buffering
- Charsets
- Development mode
- Directory browsing
- Document root
- Locales
- Web timeouts
- Virtual directories
- Clustering
- Session tracking
- JNDI mappings
- Classloading priority for Web applications

See the appendix in the *Oracle Application Server Containers for J2EE Servlet Developer's Guide* for description of the elements.

The J2EE application-client.xml File

This file contains JNDI information for accessing the server application and other client information. See ["Elements in the application-client.xml File"](#) on page B-22 for a list of the elements.

The OC4J-Specific orion-application-client.xml File

This OC4J-specific deployment file is for the client application. It contains JNDI mappings and entries for the client. See ["Elements in the orion-application-client.xml File"](#) on page B-24 for a list of the elements.

Elements in the server.xml File

The `server.xml` file is where you perform the following tasks:

- Configure OC4J
- Reference other configuration files
- Specify your J2EE application(s)

Configure OC4J

Configuring the OC4J server includes defining the following elements in the `server.xml` file:

- Library path
- Global application, global Web application, and default Web site
- Maximum number of HTTP connections the server allows
- Logging settings
- Java compiler settings
- Cluster ID
- Transaction time-out
- SMTP host

Reference Other Configuration Files

Referencing other configuration files in the `server.xml` file includes specifying the following:

- `data-sources.xml` location
- `jazn-data.xml` location
- `jms.xml` and `rmi.xml` locations

Several XML files and directories are defined in the `server.xml` file. The path to these files or directories can be relative or absolute. If relative, the path should be relative to the location of the `server.xml` file.

<application-server> Element Description

The top level element of the `server.xml` file is the `<application-server>` element.

`<application-server>`

This element contains the configuration for an application server.

Attributes:

- `application-auto-deploy-directory`=".../applications/auto"
—Specifies the directory from which EAR files are automatically detected and deployed by the running OC4J server. In addition, it performs the Web application binding for the default Web site.
- `auto-start-applications`="true | false"—If set to `true`, all applications defined in the `<applications>` elements are automatically started when the OC4J server is started. If set to `false`, the applications are not started unless their `auto-start` attribute is set to `true`. The default for `auto-start-applications` is `true`.
- `application-directory`=".../applications"— Specifies a directory in which to store applications (EAR files). If none is specified (the default), OC4J stores the information in `j2ee/home/applications`.
- `deployment-directory`=".../application-deployments"—Specifies the master location where applications that are contained in EAR files are deployed. The location defaults to `j2ee/home/application-deployments/`.
- `connector-directory`—The location and file name of the `oc4j-connectors.xml` file.
- `check-for-updates`="true | false"—This attribute is only used for standalone OC4J.
- `recovery-procedure`="automatic | prompt | ignore"— Specifies how the EJB container reacts for recovery if an error occurred in the middle of a global transaction (JTA). If a CMP bean is in the middle of a global transaction, the EJB container saves the transactional state to a file. The next time OC4J is started, these attributes specify how to recover the JTA transaction.
 - `automatic` — automatically attempts recovery (the default)
 - `prompt` — prompts the user (system in/out)
You may notice a prompt for recovery even if no CMP beans were executing. This is because the OC4J server asks for permission to see if there was anything to recover.
 - `ignore` — ignores recovery (useful in development environments or if you are never executing a CMP entity bean)
- `taskmanager-granularity`=`milliseconds`. The task manager is a background process that performs cleanup. However, the task manager can be expensive. You can manage when the task manager performs its duties through this attribute, which sets how often the task manager is kicked off for cleanup. Value is in milliseconds. Default is 1000 milliseconds.

Elements Contained Within `<application-server>`

Within the `<application-server>` element, the following elements, which are listed alphabetically and not by DTD ordering, can be configured:

`<application>`

An application is a entity with its own set of users, Web applications, and EJB JAR files.

Attributes:

- `auto-start="true|false"` — Specifies whether the application should be automatically started when the OC4J server starts. The default is `true`. Setting `auto-start` to `false` is useful if you have multiple applications installed and want them to start on demand. This can improve general server startup time and resource usage.
- `deployment-directory=".../application-deployments/myapp"` — Specifies a directory to store application deployment information. If none is specified (the default), OC4J looks in the global `deployment-directory`, and if none exists there, it stores the information inside the EAR file. The path can be relative or absolute. If relative, the path should be relative to the location of the `server.xml` file.
- `name="anApplication"` — Specifies the name used to reference the application.
- `parent="anotherApplication"` — The name of the optional parent application. The default is the global application. Children see the namespace of its parent application. This is used to share services such as EJBs among multiple applications.
- `path=".../applications/myApplication.ear" />` — The path to the EAR file containing the application code. In this example, the EAR file is named `myApplication.ear`.

<compiler>

This element is deprecated for version 9.0.4 and forward. See the `<java-compiler>` element for the alternative. For previous releases, it specifies an alternative compiler (such as Jikes) for EJB/JSP compiling.

Attributes:

- `classpath="/my/rt.jar"` — Specifies an alternative/additional classpath when compiling. Some compilers need an additional classpath (such as Jikes, which needs the `rt.jar` file of the Java 2 VM to be included).
- `executable="jikes" />` — The name of the compiler executable to use, such as Jikes or JVC.

<cluster>

Cluster settings for this server.

Attribute:

- `id="123" />` — The unique cluster ID of the server.

<execution-order>

Defines the ordering of how the startup classes are executed. Value is an integer. OC4J loads from 0 on up. If duplicate numbers, OC4J chooses the ordering for those classes.

<global-application>

The default application for this server. This acts as a parent to other applications in terms of object visibility.

Attributes:

- `name="default"` — Specifies the application.
- `path=".../application.xml" />` — Specifies the path to the global `application.xml` file, which contains the settings for the default application. An `application.xml` file exists for each application as the standard J2EE application descriptor file, which is different than this file. This

application.xml may have the same name, but it exists to provide global settings for all J2EE applications.

<global-thread-pool>

You can specify unbounded, one, or two thread pools for an OC4J process through this element. If you do not specify this element, an infinite number of threads can be created. See "[Thread Pool Settings](#)" on page 3-29 for a full description.

Attributes:

- `min` —The minimum number of threads that OC4J can simultaneously execute. By default, a minimum number of threads are preallocated and placed in the thread pool when the container starts. Value is an integer. The default is 20. The minimum value you can set this to is 10.
- `max` —The maximum number of threads that OC4J can simultaneously execute. New threads are spawned if the maximum size is not reached and if there are no idle threads. Idle threads are used first before a new thread is spawned. Value is an integer. The default is 40.
- `queue` —The maximum number of requests that can be kept in the queue. Value is an integer. The default is 80.
- `keepAlive` —The number of milliseconds to keep a thread alive (idle) while waiting for a new request. This timeout designates how long an idle thread remains alive. If the timeout is reached, the thread is destroyed. The minimum time is a minute. Time is set in milliseconds. To never destroy threads, set this timeout to a negative one.

Value is a long. The default is 600000 milliseconds.

- `cx-min` —The minimum number of connection threads that OC4J can simultaneously execute. Value is an integer. The default is 20. The minimum value you can set this to is 10.
- `cx-max` —The maximum number of connection threads that OC4J can simultaneously execute. Value is an integer. The default is 40.
- `cx-queue` —The maximum number of connection requests that can be kept in the queue. Value is an integer. The default is 80.
- `cx-keepAlive` —The number of milliseconds to keep a connection thread alive (idle) while waiting for a new request. This timeout designates how long an idle thread remains alive. If the timeout is reached, the thread is destroyed. The minimum time is a minute. Time is set in milliseconds. To never destroy threads, set this timeout to a negative one.

Value is a long. The default is 600000 milliseconds.

- `debug` —If true, print the application server thread pool information at startup. The default is false.

<global-web-app-config>

Attributes:

- `path`— The path where the web-application.xml file is located.

```
path="../../../web-application.xml" />
```

<init-library>

Attributes:

- `path`—The path in which the startup and shutdown classes are located. The path indicates the directory in which the class resides or the directory and JAR filename of the JAR where the class is archived. If more than one directory or JAR file exists, then supply an `<init-library>` element for each directory and JAR filename.

```
<init-library path="../xxx">
```

<init-param>

Attributes:

- Defines the key-value pairs of the parameters to pass into the startup class.

<javacache-config>

Attributes:

- `path`—Specifies the path to the `javacache.xml` file.

```
<javacache-config path="../../../javacache/admin/javacache.xml" />
```

<java-compiler>

You can specify an alternative compiler—either in or out of process—for your JSP and EJB compilation. The default compiler is an out of process `javac` compiler found in the JDK bin directory.

Attributes:

- `name`—Specify the name of the compiler to use. Valid compiler names are as follows:
 - for in-process compilers—`modern`, `classic`, `javac` or `ojc`
 - for out-of-process compilers (forked)—`modern`, `javac`, `ojc`, or `jikes`

These names are defined as follows:

- * `javac`—the standard compiler name for all JDKs.
- * `classic`—the standard compiler of JDK 1.1/1.2.
- * `modern`—the standard compiler of JDK 1.3/1.4.
- * `jikes`—the Jikes compiler.
- * `ojc`—The Oracle Java compiler.

- `in-process`—If true, the compiler is to run in process. If false, the compiler runs out of the process. Most compilers can execute both in and out of the process. The exceptions are as follows:
 - The `classic` compiler cannot run out of the process; thus, the `in-process` attribute is always true.
 - The `jikes` compiler cannot run in process; thus, the `in-process` attribute is always false.
- `encoding`—Specify the type of character encoding for the source file, such as UTF-8, EUCJIS, or SJIS. Encoding is only supported by the `javac` compiler. The default is determined by the language version of the JVM that is installed.
- `bindir`—Provide the absolute path to the compiler directory. You do not need to specify this attribute for `javac`, `modern`, or `classic` as the JDK bin directory is searched for this compiler.

The syntax is specific to the operating system platform:

- Sun Microsystems Solaris example—If you are using `jikes`, which is in `/usr/local/bin/jikes`, then specify the following:

```
name="jikes"
bindir="/usr/local/bin"
```

- Windows example—To specify `jikes`, which is located in `c:\jdk1.3.1\bin\jikes.exe`, specify the following:

```
name="jikes"
bindir="c:\\jdk1.3.1\\bin"
```

- `extdirs`—Specifies extension directories that the compilation uses to compile against. The default is your JDK extension directories. Multiple directories can be specified, each separated by a colon. Each JAR archive in the specified directories are searched for class files. You can specify certain directories to be searched by modifying the `-Djava.ext.dirs` system property. The `jikes` compiler requires that extension directories are specified in either this attribute or in the `-Djava.ext.dirs` system property.

The following are four examples of how to define alternate compilers in this element:

```
<java-compiler name="jikes" bindir="C:\java\jikes\bin"
in-process="false" />
<java-compiler name="ojc" bindir="C:\java\jdev\jdev\bin"
in-process="false"/>
<java-compiler name="classic" in-process="true" />
<java-compiler name="modern" in-process="true" />
```

<jms-config>

Attribute:

- `path`— Specifies the path to the `jms.xml` file.

```
path=".../jms.xml"
```

<log>

<file>

Attribute:

- `path=".../log/server.log"` — Specifies a relative or absolute path to a file where log events are stored.

<mail>

An e-mail address where log events are forwarded. You must also specify a valid mail-session if you use this option.

Attribute:

- `address="my@mail.address"` — Specifies the mail address.

<odl>

The ODL log entries are each written out in XML format in its respective log file. The log files have a maximum limit. When the limit is reached, the log files are overwritten.

When you enable ODL logging, each message goes into its respective log file, named `logN.xml`, where `N` is a number starting at one. The first log message starts the log file, `log1.xml`. When the log file size maximum is reached, the second log file is opened to continue the logging, `log2.xml`. When the last logfile is full, the first log

file, `log1.xml` is erased and a new one is opened for the new messages. Thus, your log files are constantly rolling over and do not encroach on your disk space.

Attributes:

- `path`: Path and folder name of the log folder for this area. You can use an absolute path or a path relative to where the configuration XML file exists, which is normally in the `j2ee/home/config` directory. This denotes where the log files will reside for the feature that the XML configuration file is concerned with. For example, modifying this element in the `server.xml` file denotes where the server log files are written.
- `max-file-size`: The maximum size in KB of each individual log file.
- `max-directory-size`: The maximum size of the directory in KB. The default directory size is 10 MB.

New files are created within the directory, until the maximum directory size is reached. Each log file is equal to or less than the maximum specified in the attributes.

<max-http-connections>

Used to define the maximum number of concurrent connections any given Web site can accept at a single point in time. If text exists inside the tag, it is used as a redirect-URL when the limit is reached.

Attributes:

- `max-connections-queue-timeout="10"` — When the maximum number of connections are reached, this is the number of seconds that can pass before the connections are dropped and a message is returned to the client stating that the server is either busy or connections will be redirected. The default is 10 seconds.
- `socket-backlog` — The number of connections to queue up before denying connections at the socket level. The default is 30.
- `value` — The maximum number of connections.

<metric-collector>

The `<metric-collector>` element specifies that OC4J sends a metric between 0 and 100, inclusive, to `mod_oc4j` so that `mod_oc4j` can make routing decisions to load-balance incoming requests to a list of available OC4J instances. The metric sent has a relative value only, where 0 means that the OC4J instance is very busy and 100 means that the OC4J instance is available (i.e. not busy). When configured for metric load balancing, `mod_oc4j` routes first to the OC4J instance with the greater value.

The metric sent from OC4J to `mod_oc4j` is used only when metric-based load balancing is specified for `mod_oc4j` and when OC4J runs in an Oracle Application Server environment.

If you specify metric-based load balancing in `mod_oc4j` and do not specify the `<metric-collector>` element in `server.xml`, then `mod_oc4j` expects OC4J to send metrics, but OC4J does not send metrics. In this case, `mod_oc4j` reports the following warning message:

```
No run time metrics for oc4j(opmnid=%s) in notification Oc4jSelectMethod is
configured to use run time metrics, please make sure OC4J side is configured
accordingly. Default to 50.
```

In this case, `mod_oc4j` uses the value 50 for each of the OC4J processes and continues.

Likewise, if you specify the `<metric-collector>` element in `server.xml`, but do not specify metric-based load balancing in `mod_oc4j`, then OC4J sends metrics but

`mod_oc4j` is not configured to receive metrics. In this case, `mod_oc4j` ignores the metrics and uses whatever the configured method is for load balancing. You specify the load balancing method with `Oc4jSelectMethod`. If no `Oc4jSelectMethod` is specified, then `mod_oc4j` uses the default, which is `roundrobin`.

The `<metric-collector>` element takes a single attribute: `classname`. This attribute defines an interface for gathering and calculating a server-wide metric. Use `oracle.oc4j.server.DMSMetricCollector` for the `classname` attribute when using a DMS-noun-based metric collector. A `DMSMetricCollector` instance takes several parameters.

For example:

```
<metric-collector classname="oracle.oc4j.server.DMSMetricCollector">
  <init-param>
    <param-name>
      dms-noun
    </param-name>
    <param-value>
      /oc4j/default/WEBS/processRequest.time
    </param-value>
  </init-param>
  <init-param>
    <param-name>
      history-proportion
    </param-name>
    <param-value>
      0.2
    </param-value>
  </init-param>
  <init-param>
    <param-name>
      debug
    </param-name>
    <param-value>
      false
    </param-value>
  </init-param>
</metric-collector>
```

For details on using metric-based load balancing with `mod_oc4j`, see the *Oracle HTTP Server Administrator's Guide*.

<rmi-config>

Attribute:

- `path`— Specifies the path to the `rmi.xml` file.

```
path="../../../rmi.xml"
```

<sep-config>

The `<sep-config>` element in this file specifies the pathname, normally `internal-settings.xml`, for the server extension provider properties.

Attribute:

- `path`—The path of the server extension provider properties.

<sfsb-config>

Passivation for stateful session beans is automatically done, unless you set the `enable-passivation` attribute for this element to `false`. For more information on

stateful session bean passivation, see the Advanced chapter in the *Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide*.

Attribute

- `enable-passivation`—Default is true, which means that stateful session bean passivation occurs. If you have a situation where your stateful session beans are not in a state to be passivated, set this attribute to false.

<shutdown-classes>

Shutdown classes can be defined by the user, and are executed after undeployment, but before the core services are stopped.

<shutdown-class>

Each startup class is defined within the `<startup-class>` element.

Attributes:

- `classname`—The classname of the user-defined startup class.

<startup-classes>

Startup classes can be defined by the user, and will be executed after the core services (JMS, RMI) are started, but before applications are deployed. The shutdown classes are executed after undeployment, but before the core services are stopped.

<startup-class>

Each startup class is defined within the `<startup-class>` element.

Attributes:

- `classname`—The classname of the user-defined startup class.
- `failure-is-fatal`—If true, if an exception is thrown, then OC4J logs the exception and exit. If false, OC4J logs the exception and then continues. Default is false.

<transaction-config>

Transaction configuration for the server.

Attribute:

- `timeout="30000"` — Specifies the maximum amount of time (in milliseconds) that a transaction can take to finish before it is rolled back due to a timeout. The default value is 30000. This timeout will be a default timeout for all transactions that are started in OC4J. You can change it by using the dynamic API—`UserTransaction.setTimeout(milliseconds)`.

<web-site>

Attribute:

- `path`— The path to a `*web-site.xml` file that defines a Web site. For each Web site, you must specify a separate `*web-site.xml` file. This example shows that a Web site is defined in the `my-web-site.xml` file.

```
path="../../../my-web-site.xml"
```

Elements in the application.xml File

This section provides an overview of the J2EE application deployment descriptor file.

<application> Element Description

The top level element of the `application.xml` file is the `<application>` element.

Elements Contained Within `<application>`

Within the `<application>` element, the following elements, which are listed alphabetically and not by DTD ordering, can be configured:

`<alt-dd>path/to/dd</alt-dd>`

The `alt-dd` element specifies an optional URI to the post-assembly version of the deployment descriptor file for a particular J2EE module. The URI must specify the full pathname of the deployment descriptor file relative to the application's root directory. If `alt-dd` is not specified, the deployer must read the deployment descriptor from the default location and file name required by the respective component specification.

`<connector>context</connector>`

The `connector` element specifies the URI of a resource adapter archive file, relative to the top level of the application package.

`<context-root>thedir/</context-root>`

The `context-root` element specifies the context root of a web application.

`<description>A description.</description>`

The `description` element provides a human readable description of the application. The `description` element should include any information that the application assembler wants to provide the deployer.

`<display-name>The name.</display-name>`

The `display-name` element specifies an application name. The application name is assigned to the application by the application assembler and is used to identify the application to the deployer at deployment time.

`<ejb>pathToEJB.jar</ejb>`

The `ejb` element specifies the URI of a EJB JAR, relative to the top level of the application package.

`<icon>`

The `icon` element contains a `small-icon` and a `large-icon` element which specify the location within the application for a small and large image used to represent the application in a GUI tool.

`<java>pathToClient.jar</java>`

The `java` element specifies the URI of a Java application client module, relative to the top level of the application package.

`<large-icon>path/to/icon.gif</large-icon>`

The `large-icon` element contains the location within the application of a file containing a large (32x32 pixel) icon image. The image must be either GIF or JPEG format and the filename must end with the extension of ".gif" or ".jpg".

`<module>`

The `module` element represents a single J2EE module and contains an EJB, Java, or Web element, which indicates the module type and contains a path to the module file, and an optional `alt-dd` element, which specifies an optional URI to the post-assembly version of the deployment descriptor. The application deployment

descriptor must have one module element for each J2EE module in the application package.

```
<role-name>nameOfRole</role-name>
```

The name of the role.

```
<security-role>
```

The `security-role` element contains the definition of a security role which is global to the application. The definition consists of a description of the security role, and the security role name. The descriptions at this level override those in the component level security role definitions and must be the descriptions tool display to the deployer.

```
<small-icon>path/to/icon.gif</small-icon>
```

The `small-icon` element contains the location within the application of a file containing a small (16x16 pixel) icon image. The image must be either GIF or JPEG format and the filename must end with the extension of ".gif" or ".jpg".

```
<web>
```

The `web` element contains the `web-uri` and `context-root` of a Web application module.

```
<web-uri>pathTo.war</web-uri>
```

The `web-uri` element specifies the URI of a web application file, relative to the top level of the application package.

Elements in the orion-application.xml File

This section describes the OC4J-specific application deployment descriptor file.

<orion-application> Element Description

The top level element of the `orion-application.xml` file is the `<orion-application>` element.

Attributes:

- `autocreate-tables` - Whether or not to automatically create database tables for CMP beans in this application. The default is true.
- `autodelete-tables` - Whether or not to automatically delete old database tables for CMP beans when redeploying in this application. The default is false.
- `default-data-source` - The default data source to use if other than server default. This must point to a valid CMT data source for this application if specified.
- `deployment-version` - The version of OC4J that this JAR was deployed against, if it is not matching the current version then it will be redeployed. This is an internal server value; do not edit.
- `treat-zero-as-null` - Whether or not to treat read zero's as null's when they represent primary keys. The default is false.

Elements Contained Within <orion-application>

Within the `<orion-application>` element, the following elements, which are listed alphabetically and not by DTD ordering, can be configured:

```
<argument value="theValue" />
```

An argument used when invoking the client.

Attribute:

- `value` - The value of the argument.

<arguments>

A list of arguments to used when invoking the application client if starting it in-process (`auto-start="true"`).

```
<client-module auto-start="true|false"
deployment-time="073fc2ab513bc3ce" path="myappclient.jar"
user="theUser">
```

An application client module of the application. An application client is a GUI or console-based standalone client that interacts with the server.

Attributes:

- `auto-start` - Whether or not to auto-start the client (in-process) at server startup. The default is false.
- `deployment-time` - Last deploy time attribute. Internal to OC4J; do not edit.
- `path` - The path (relative to the enterprise archive or absolute) to the application-client.
- `user` - User to run the client as if run in-process (`autostart="true"`). Must be specified if `auto-start` is activated.

<commit-coordinator>

Configure the two-phase commit engine.

```
<commit-class
class="com.evermind.server.OracleTwoPhaseCommitDriver" />
```

Attribute:

- `class` - Configures the `OracleTwoPhaseCommitDriver` class for two-phase commit engines.

```
<connectors path="./oc4j-connectors.xml" />
```

Attribute:

- `path` - The name and path of the `oc4j-connectors.xml` file. If no `<connectors>` element is specified, then the default path is `<oc4j>/j2ee/home/connectors/rarname./oc4j-connectors.xml`.

```
<data-sources path="./data-sources.xml" />
```

Attribute:

- `path` - The path.

```
<description>A short description</description>
```

A short description of this component.

```
<ejb-module path="myEjbs.jar" remote="true|false" />
```

An EJB JAR module of the application.

Attributes:

- `path` - The path (relative to the enterprise archive or absolute) to the `ejb-jar`.

- `remote` - `true/false` value stating whether or not to activate the EJB instances on this node or to look them up remotely from another server (remote or inside a cluster). The default is `false`.

```
<file path="../../log/server.log" />
```

A relative/absolute path to log events to.

Attribute:

- `path` - The path.

```
<group name="theGroup" />
```

A group that this security-role-mapping implies. That is, all members of the specified group are included in this role.

Attribute:

- `name` - The name of the group.

```
<jazn provider="XML" location="./jazn-data.xml" />
```

Configure the JAAS to use the XML-based provider type.

Attributes:

- `provider` - XML
- `location` - Path to file. For example: `./jazn-data.xml` This can be an absolute path, or a path relative to the `jazn.xml` file, where the JAAS Provider first looks for the `jazn-data.xml` in the directory containing the `jazn.xml` file. Optional if `jazn.xml` file configured, otherwise Required
- `persistence` - Values can be `NONE` (Do not persist changes), `ALL` (Persist changes after every modification), `VM_EXIT` - (Default- Persist changes when VM exits)
- `default-realm` - A realm name. For example: `sample_subrealm`. Optional if only one realm is configured.

```
<jazn-web-app auth-method="SSO" runas-mode="false"
doasprivileged-mode="true" />
```

The filter element of `JAZNUserManager`.

Attributes:

- `auth-method` - Set `auth-method` to `SSO` (single sign-on). If you do not set this parameter, it defaults to `null`.
- The `runas-mode` and `doasprivileged-mode` settings are described in [Table B-1](#). See the *Oracle Application Server Containers for J2EE Security Guide* for more information.

Table B-1 *runas-mode and doasprivileged-mode Settings*

If runas-mode is Set To...	If doasprivileged-mode is Set To...	Then...
true	true (default)	Subject.doAsPrivileged in a privilegedExceptionAction block that calls chain.doFilter (myrequest, response)
true	false	Subject.doAs in a privilegedExceptionAction block that calls chain.doFilter (myrequest, response)
false (default)	true	chain.doFilter (myrequest, response)
false	false	chain.doFilter (myrequest, response)

<library path="../../lib/" />

A relative/absolute path/URL to a directory or a JAR/ZIP to add as a library-path for this server. Directories are scanned for JARS/ZIP files to include at startup.

Attribute:

- path - The path.

<log>

Logging settings.

<odl>

The ODL log entries are each written out in XML format in its respective log file. The log files have a maximum limit. When the limit is reached, the log files are overwritten.

When you enable ODL logging, each message goes into its respective log file, named logN.xml, where N is a number starting at one. The first log message starts the log file, log1.xml. When the log file size maximum is reached, the second log file is opened to continue the logging, log2.xml. When the last logfile is full, the first log file, log1.xml is erased and a new one is opened for the new messages. Thus, your log files are constantly rolling over and do not encroach on your disk space.

Attributes:

- path: Path and folder name of the log folder for this area. You can use an absolute path or a path relative to where the configuration XML file exists, which is normally in the j2ee/home/config directory. This denotes where the log files will reside for the feature that the XML configuration file is concerned with. For example, modifying this element in the server.xml file denotes where the server log files are written.
- max-file-size: The maximum size in KB of each individual log file.
- max-directory-size: The maximum size of the directory in KB. The default directory size is 10 MB.

New files are created within the directory, until the maximum directory size is reached. Each log file is equal to or less than the maximum specified in the attributes.

<mail address="my@mail.address" />

An e-mail address to log events to. A valid mail-session also needs to be specified if this option is used.

Attribute:

- `address` - The mail-address.

```
<mail-session location="mail/TheSession"
smtp-host="smtp.server.com">
```

The session SMTP-server host (if using SMTP).

Attributes:

- `location` - The location in the namespace to store the session at.
- `smtp-host` - The session SMTP-server host (if using SMTP).

```
<namespace-access>
```

Namespace (naming context) security policy for RMI clients.

```
<namespace-resource root="the/path">
```

A resource with a specific security setting.

Attribute:

- `root` - The root of the part of the namespace that this rule applies to.

```
<password-manager>
```

Specifies the `UserManager` that is used for the lookup of hidden passwords. If omitted, the current `UserManager` is used for authentication and authorization. For example, you can use a JAZN LDAP `UserManager` for the overall `UserManager`, but use a JAZN XML `UserManager` for checking hidden passwords.

To identify a `UserManager`, provide a `<jazn>` element definition within this element, as follows:

```
<password-manager>
<jazn ...>
</password-manager>
```

```
<persistence path="./persistence" />
```

A relative (to the application root) or absolute path to a directory where application state should be stored across restarts.

Attribute:

- `path` - The path (relative to the enterprise archive or absolute) to the persistence directory.

```
<principals path="principals.xml" />
```

Attribute:

- `path` - The path (relative to the enterprise archive or absolute) to the principals file.

```
<property name="theName" value="theValue" />
```

Contains a name/value pair initialization param.

Attributes:

- `name` - The name of the parameter.
- `value` - The value of the parameter.

```
<read-access>
```

The read-access policy.

<resource-provider>

Define a JMS resource provider. To add a custom `<resource-provider>`, add the following to your `orion-application.xml` file:

```
<resource-provider class="providerClassName" name="JNDI name">
  <description> description </description>
  <property name="name" value="value" />
</resource-provider>
```

In place of the user-replaceable constructs (those in italics) in the preceding code, do the following:

- Replace the value *providerClassName* of the `class` attribute with the name of the resource-provider class.
- Replace the value *JNDI name* of the `name` attribute with a name by which to identify the resource provider. This name will be used in finding the resource provider in the application's JNDI as `"java:comp/resource/name/"`.
- Replace the value *description* of the `description` tag with a description of the specific resource provider.
- Replace the values *name* and *value* of the corresponding attributes with the same name in any property tags that the specific resource provider needs to be given as parameters.

<security-role-mapping impliesAll="true|false" name="theRole">

The runtime mapping (to groups and users) of a role. Maps to a security-role of the same name in the assembly descriptor.

Attributes:

- `impliesAll` - Whether or not this mapping implies all users. The default is false.
- `name` - The name of the role

```
<user name="theUser" />
```

A user that this security-role-mapping implies.

Attribute:

- `name` - The name of the user.

<user-manager class="com.name.of.TheUserManager" display-name="Friendly UserManager name">

Specifies an optional user-manager to use. For example, user-managers are `com.evermind.sql.DataSourceUserManager`, `com.evermind.ejb.EJBUserManager`, and so on. These are used to integrate existing systems and provide custom user-managers for Web applications.

Attributes:

- `class` - The fully qualified classname of the user-manager.
- `display-name` - A descriptive name for this `UserManager` instance.

```
<web-module id="myWebApp" path="myWebApp.war" />
```

A Web application module of the application. Each Web application can be installed on any site and in any context on those sites (for instance `http://www.myserver.com/myapp/`).

Attributes:

- `id` - The name used to reference this web-application when used in web-sites etc.
- `path` - The path (relative to the enterprise archive or absolute) to the web-application.

<write-access>

The write access policy.

Elements in the application-client.xml File

This section describes the J2EE application client deployment descriptor file.

<application-client> Element Description

The top level element of the `application-client.xml` file is the `<application-client>` element.

<application-client>

The `application-client` element is the root element of an application client deployment descriptor. The application client deployment descriptor describes the EJB components and external resources referenced by the application client.

Elements Contained Within <application-client>

Within the `<application-client>` element, the following elements, which are listed alphabetically and not by DTD ordering, can be configured:

<callback-handler>

The `callback-handler` element names a class provided by the application. The class must have a no args constructor and must implement the `javax.security.auth.callback.CallbackHandler` interface. The class will be instantiated by the application client container and used by the container to collect authentication information from the user.

<description>The description</description>

A short description.

<display-name>The name</display-name>

The `display-name` element contains a short name that is intended to be displayed by tools.

<ejb-link>EmployeeRecord</ejb-link>

The `ejb-link` element is used in the `ejb-ref` element to specify that an EJB reference is linked to an enterprise bean in the encompassing J2EE Application package. The value of the `ejb-link` element must be the `ejb-name` of an enterprise bean in the same J2EE Application package.

<ejb-ref>

The `ejb-ref` element is used for the declaration of a reference to an enterprise bean's home. The declaration consists of an optional description; the EJB reference name used in the code of the referencing application client; the expected type of the referenced enterprise bean; the expected home and remote interfaces of the referenced enterprise bean; and an optional `ejb-link` information. The optional `ejb-link` element is used to specify the referenced enterprise bean.

<ejb-ref-name>ejb/Payroll</ejb-ref-name>

The `ejb-ref-name` element contains the name of an EJB reference. The EJB reference is an entry in the enterprise bean's environment. It is recommended that name is prefixed with "ejb/".

<ejb-ref-type>Entity/Session</ejb-ref-type>

The `ejb-ref-type` element contains the expected type of the referenced enterprise bean. The `ejb-ref-type` element must be one of the following: Entity Session

<env-entry>

The `env-entry` element contains the declaration of an Enterprise JavaBean's environment entries. The declaration consists of an optional description, the name of the environment entry, and an optional value.

<env-entry-name>minAmount</env-entry-name>

The `env-entry-name` element contains the name of an Enterprise JavaBean's environment entry.

<env-entry-type>java.lang.String</env-entry-type>

The `env-entry-type` element contains the fully-qualified Java type of the environment entry value that is expected by the enterprise bean's code. The following are the legal values of `env-entry-type`: `java.lang.Boolean`, `java.lang.String`, `java.lang.Integer`, `java.lang.Double`, `java.lang.Byte`, `java.lang.Short`, `java.lang.Long`, and `java.lang.Float`.

<env-entry-value>100.00</env-entry-value>

The `env-entry-value` element contains the value of an Enterprise JavaBean's environment entry.

<home>com.aardvark.payroll.PayrollHome</home>

The `home` element contains the fully-qualified name of the Enterprise JavaBean's home interface.

<icon>

The `icon` element contains a `small-icon` and `large-icon` element which specify the URIs for a small and a large GIF or JPEG icon image used to represent the application client in a GUI tool.

<large-icon>lib/images/employee-service-icon32x32.jpg

</large-icon>

The `large-icon` element contains the name of a file containing a large (32 x 32) icon image. The file name is a relative path within the application client JAR file. The image must be either in the JPEG or GIF format, and the file name must end with the suffix ".jpg" or ".gif" respectively. The icon can be used by tools.

<remote>com.wombat.empl.EmployeeService</remote>

The `remote` element contains the fully-qualified name of the Enterprise JavaBean's remote interface.

<res-auth>Application/Container</res-auth>

The `res-auth` element specifies whether the Enterprise JavaBean code signs on programmatically to the resource manager, or whether the Container will sign on to the resource manager on behalf of the bean. In the latter case, the Container uses

information that is supplied by the Deployer. The value of this element must be one of the two following: `Application` or `Container`

<resource-env-ref>

The `resource-env-ref` element contains a declaration of an application's reference to an administered object associated with a resource in the application's environment. It consists of an optional description, the resource environment reference name, and an indication of the resource environment reference type expected by the application code.

<resource-env-ref-name>

The `resource-env-ref-name` element specifies the name of a resource environment entry name used in the application code.

<resource-env-ref-type>

The `resource-env-ref-type` element specifies the type of a resource environment reference.

<resource-ref>

The `resource-ref` element contains a declaration of Enterprise JavaBean's reference to an external resource. It consists of an optional description, the resource factory reference name, the indication of the resource factory type expected by the enterprise bean code, and the type of authentication (`Bean` or `Container`).

<res-ref-name>name</res-ref-name>

The `res-ref-name` element specifies the name of a resource factory reference.

<res-sharing-scope>Shareable</res-sharing-scope>

The `res-sharing-scope` element specifies whether connections obtained through the given resource manager connection factory reference can be shared. The value of this element, if specified, must be one of the following: `Shareable` or `Unshareable`. The default value is `Shareable`.

<res-type>javax.sql.DataSource</res-type>

The `res-type` element specifies the type of the data source. The type is specified by the Java interface (or class) expected to be implemented by the data source.

<small-icon>lib/images/employee-service-icon16x16.jpg</small-icon>

The `small-icon` element contains the name of a file containing a small (16 x 16) icon image. The file name is a relative path within the application client JAR file. The image must be either in the JPEG or GIF format, and the file name must end with the suffix ".jpg" or ".gif" respectively. The icon can be used by tools.

Elements in the orion-application-client.xml File

This section provides an overview of the OC4J-specific application client deployment descriptor file.

<orion-application-client> Element Description

The top level element of the `orion-application-client.xml` file is the `<orion-application-client>` element.

<orion-application-client>

An `orion-application-client.xml` file contains the deploy time information for a J2EE application client. It complements the application client assembly information found in `application-client.xml`.

Elements Contained Within `<orion-application-client>`

Within the `<orion-application-client>` element, the following elements, which are listed alphabetically and not by DTD ordering, can be configured:

```
<context-attribute name="name" value="value" />
```

An attribute sent to the context. The only mandatory attribute in JNDI is the `'java.naming.factory.initial'`, which is the classname of the context factory implementation.

Attributes:

- `name` - The name of the attribute.
- `value` - The value of the attribute.

```
<ejb-ref-mapping location="ejb/Payroll" name="ejb/Payroll" />
```

The `ejb-ref` element is used for the declaration of a reference to another enterprise bean's home. The `ejb-ref-mapping` element ties this to a JNDI-location when deploying.

Attributes:

- `location` - The JNDI location to look up the EJB home from.
- `name` - The `ejb-ref` name. Matches the name of an `ejb-ref` in `application-client.xml`.

```
<env-entry-mapping  
name="theName">deploymentValue</env-entry-mapping>
```

Overrides the value of an `env-entry` in the assembly descriptor. It is used to keep the EAR (assembly) clean from deployment-specific values. The body is the value.

Attributes:

- `name` - The name of the context parameter.

```
<lookup-context location="foreign/resource/location">
```

The specification of an optional `javax.naming.Context` implementation used for retrieving the resource. This is useful when hooking up with third party modules, such as a third party JMS server for instance. Either use the context implementation supplied by the resource vendor or if none exists write an implementation which in turn negotiates with the vendor software.

Attributes:

- `location` - The name looked for in the foreign context when retrieving the resource.

```
<resource-env-ref-mapping location="jdbc/TheDS" >
```

The `resource-env-ref` element is used for the declaration of a reference to an external resource, such as a data source, JMS queue, mail session, or similar. The `resource-env-ref-mapping` ties that element to a JNDI location during deployment.

Attributes:

- `location` - The JNDI location to bind the resource to.

```
<resource-ref-mapping location="jdbc/TheDS"
name="jdbc/TheDSVar">
```

The `resource-ref` element is used for the declaration of a reference to an external resource such as a data source, JMS queue, mail session or similar. The `resource-ref-mapping` ties this to a JNDI-location when deploying.

Attributes:

- `location` - The JNDI location to look up the resource home from.
- `name` - The `resource-ref` name. Matches the name of an `resource-ref` in `application-client.xml`.

Configuration and Deployment Examples

The following example shows how to configure and deploy a J2EE application within OC4J. See ["Deploying Applications"](#) on page 2-11 to learn how to modify the XML configuration files for the FAQ demo.

In this example, the `myapp` application contains a Java client, an EJB assembled into a JAR file, servlets and JSPs assembled into a WAR file, and an EAR file that contains both the EJB JAR file and the Web application WAR file. The tree structure showing the location of all the XML configuration files, the Java class files, and the JSP files is shown in ["Configuration File Examples"](#) below. Notice that you can separate all the configuration files into logical directories within the application directory.

Configuration File Examples

Below are examples of the various configuration files.

application.xml Example

The `myapp/META-INF/application.xml` file lists the EJB JAR and Web application WAR file that is contained in the EAR file using the `<module>` elements.

```
<?xml version="1.0"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application
1.3//EN" "http://java.sun.com/j2ee/dtds/application_1_3.dtd">
<application>
  <display-name>myapp j2ee application</display-name>
  <description>
    A sample J2EE application that uses a Container Managed
    Entity Bean and JSPs for a client.
  </description>
  <module>
    <ejb>myapp-ejb.jar</ejb>
  </module>
  <module>
    <web>
      <web-uri>myapp-web.war</web-uri>
      <context-root>/myapp</context-root>
    </web>
  </module>
</application>
```

web.xml Example

The `myapp/web/WEB-INF/web.xml` file contains the class definitions for EJBs, servlets, and JSPs that are executed within the Web site. The `myapp` Web module specifies the following in its descriptor:

- The default page to be displayed for the application's root context (`http://<host>:<port>/j2ee/myapp`)
- Where to find the stubs for the EJB home and remote interfaces
- The JNDI name for the EJB
- The included servlets and where to find each servlet class
- How servlets are mapped to a subcontext using the `<servlet-mapping>` element (`/template`) off of the application root context

The Web server looks for the following:

- All servlet classes under `WEB-INF/classes/<package>.<class>`.
- All HTML and JSP from the root of the WAR file that is pointed to by `<web-app name="<warfile.war">` in the `web-site.xml` file, which is packaged in the deployed corresponding application EAR file.
- OC4J compiles each JSP from `.java` into `.class` the first time it is used and caches it for subsequent use.

```
<web-app>
  <display-name>myapp web application</display-name>
  <description>
    Web module that contains an HTML welcome page, and 4 JSP's.
  </description>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <ejb-ref>
    <ejb-ref-name>TemplateBean</ejb-ref-name>
    <ejb-ref-type>Entity</ejb-ref-type>
    <home>TemplateHome</home>
    <remote>Template</remote>
  </ejb-ref>
  <servlet>
    <servlet-name>template</servlet-name>
    <servlet-class>TemplateServlet</servlet-class>
    <init-param>
      <param-name>length</param-name>
      <param-value>1</param-value>
    </init-param>
  </servlet>
</web-app>
```

ejb-jar.xml Example

The `ejb-jar.xml` file contains the definitions for a container-managed persistent EJB. The `myapp` EJB deployment descriptor contains the following:

- The entity bean uses container-managed persistence.
- The primary key is stored in a table. This descriptor defines the type and fields of the primary key.

- The table name is `TemplateBean`, and columns are named according to fields in the `ejb-jar.xml` descriptor and type mappings in `j2ee/home/config/database-schemas/oracle.xml`.
- The bean uses JDBC to access databases, as specified in `data-source.xml`, by `ejb-location` or by `default-data-source` in `orion-application.xml`.

```

<ejb-jar>
  <display-name>myapp</display-name>
  <description>
    An EJB app containing only one Container Managed Persistence
    Entity Bean
  </description>
  <enterprise-beans>
    <entity>
      <description>
        template bean populates a generic template table.
      </description>
      <display-name>TemplateBean</display-name>
      <ejb-name>TemplateBean</ejb-name>
      <home>TemplateHome</home>
      <remote>Template</remote>
      <ejb-class>TemplateBean</ejb-class>
      <persistence-type>Container</persistence-type>
      <prim-key-class>java.lang.Integer</prim-key-class>
      <reentrant>False</reentrant>
      <cmp-field><field-name>empNo</field-name></cmp-field>
      <cmp-field><field-name>empName</field-name></cmp-field>
      <cmp-field><field-name>salary</field-name></cmp-field>
      <primkey-field>empNo</primkey-field>
    </entity>
  </enterprise-beans>
  <assembly-descriptor>
    <container-transaction>
      <method>
        <ejb-name>TemplateBean</ejb-name>
        <method-name>*</method-name>
      </method>
      <trans-attribute>NotSupported</trans-attribute>
    </container-transaction>
    <security-role>
      <description>Users</description>
      <role-name>users</role-name>
    </security-role>
  </assembly-descriptor>
</ejb-jar>

```

server.xml Addition

When you deploy the application using the deployment wizard, this adds the location of the application EAR file to the `server.xml` file. This causes the application to be started every time that OC4J is started. If you do not want the application to be started with OC4J, change the `auto-start` variable to `FALSE`.

Note: If you set `auto-start` to `FALSE`, you can manually start the application through Enterprise Manager or it is automatically started when a client requests the application.

```
<application name="myapp" path="../myapp/myapp.ear"
```

```
auto-start="true" />
```

where

- The `name` variable is the name of the application.
- The `path` indicates the directory and filename for the EAR file.
- The `auto-start` variable indicates if this application should be automatically started each time OC4J is started.

default-web-site.xml Addition

The deployment wizard defines the root context for the Web application and binds the Web context and adds the following to the `default-web-site.xml` file:

```
<web-app application="myapp" name="myapp-web" root="/myapp" />
```

- The `name` variable is the name of the WAR file, without the `.WAR` extension.
- The `root` variable defines the root context for the application off of the Web site. For example, if you defined your Web site as `"http://<host>:7777/j2ee"`, then to initiate the application, you would point your browser at `"http://<host>:7777/j2ee/myapp"`.

Client Example

The application client that accesses the `myapp` application has a descriptor, which describes where to find the EJB stubs (home and remote interface) and its JNDI name.

The client XML configuration is contained in two files: `application-client.xml` and `orion-application-client.xml`.

The `application-client.xml` file contains a reference for an EJB, as follows:

```
<application-client>
<display-name>TemplateBean</display-name>
<ejb-ref>
<ejb-ref-name>TemplateBean</ejb-ref-name>
<ejb-ref-type>Entity</ejb-ref-type>
<home>mTemplateHome</home>
<remote>Template</remote>
</ejb-ref>
</application-client>
```

The `orion-application-client.xml` file maps the EJB reference logical name to the JNDI name for the EJB. For example, this file maps the `<ejb-ref-name>` element, `"TemplateBean"`, defined in the `application-client.xml`, to the JNDI name, `"myapp/myapp-ejb/TemplateBean"`, as follows:

```
<orion-application-client>
<ejb-ref-mapping name="TemplateBean" location="myapp/myapp-ejb/TemplateBean" />
</orion-application-client>
```

JNDI Properties for the Client Set the JNDI properties for a regular client so it finds the initial JNDI context factory in one of the following manners:

- Set the JNDI properties within a `Hashtable`, then pass the properties to `javax.naming.InitialContext`.
- Set the JNDI properties within a `jndi.properties` file.

If you provide the JNDI properties in the `jndi.properties` file, package the properties in `myapp-client.jar` to ensure that it is in the `CLASSPATH`.

```
jndi.properties:
-----
java.naming.factory.initial=com.evermind.server.ApplicationClientInitialContext
Factory
java.naming.provider.url=
opmn:ormi://<opmnhost>:<oc4j_instance>:7777/j2ee/myapp
java.naming.security.principal=admin
java.naming.security.credentials=welcome
```

Client Module—Standalone Java Client Invoking EJBs

Package your client module in a JAR file with the descriptor
META-INF/application-client.xml.

Manifest File for the Client Package the client in a runnable JAR with a manifest that has the main class to run and required CLASSPATH, as shown below. Check that the relative paths in this file are correct. Verify that you point to the relative location of the required OC4J class libraries.

```
manifest.mf
-----
Manifest-Version: 1.0
Main-Class: myapp.myapp-client.TemplateClient
Name: "TemplateClient"
Created-By: 1.2 (Sun Microsystems Inc.)
Implementation-Vendor: "Oracle"
Class-Path: ../../../../j2ee/home/oc4j.jar ../../../../j2ee/home/jndi.jar
../../../../j2ee/home/ejb.jar ../myapp-ejb.jar
```

Executing the Client To execute the client, perform the following:

```
% java -jar myapp-client.jar
TemplateClient.main(): start
Enter integer value for col_1: 1
Enter string value for col_2: BuyME
Enter float value for col_3: 99.9
Record added through bean
```

OC4J Command-Line Options and System Properties

You can set both system properties and command-line options on the OC4J command-line before startup. If OC4J is running, you must restart the instance for these to take effect. All system properties are prefaced with a -D. For example, -Dhttp.session.debug. All command-line options are prefaced with a hyphen (-). For example, -help.

- [Table B-2](#) details OC4J command-line options.
- [Table B-3](#) details general system properties.
- [Table B-4](#) details debugging properties.

As described in "[Configuring Server Properties](#)" on page 3-2 and shown in [Figure B-1](#), "[EM Console to Modify Server Properties for an OC4J Instance](#)", the -D system properties are entered on the Java Options line; the OC4J command-line options are entered on the OC4J Options line.

Figure B-1 EM Console to Modify Server Properties for an OC4J Instance

Multiple VM Configuration

✓ **TIP** If OC4J is running, newly added islands and associated processes will be automatically started.

Islands

Island ID	Number of Processes
default_island	1
<input type="button" value="Add Another Row"/>	

Ports

RMI Ports	3101-3200
JMS Ports	3201-3300
AJP Ports	3001-3100

Command Line Options

Java Executable	
OC4J Options	-properties -out oc4j.out -err oc4j.err
Java Options	-Dhttp.session.debug=true

Table B-2 OC4J Command-Line Options

Command-Line Options	Description
-install	Installs the server, activates the Admin account, and rewrites text files to match the OS linefeed, and so on.
-quiet	Suppress standard output.
-config	Specifies a location for the <code>server.xml</code> file.
-rewriteXML	Rewrites bad XML files (after prompting) as accurately as possible. Warning: If you have corrupt XML files, you may lose data when rewriting if they're badly malformed. Use this command with care.
-out [file]	Specifies a file to route the standard output to. The file contains messages that are printed to <code>System.out</code> , as well as the messages sent to output through the servlet logging interface. If not specified, all output is written to standard out. See Table B-5, "stdout/stderr Archive Management Properties" on page B-35 for additional system properties that can be set to manage <code>stdout</code> files.
-err [file]	Specifies a file to route standard error output to. If not specified, all errors are written to standard error. See Table B-5, "stdout/stderr Archive Management Properties" on page B-35 for additional system properties that can be set to manage <code>stderr</code> files.
-monitorResourceThreads	Enables backup debugging of thread resources. Enable this only if you have problems that relates to threads getting stuck in critical sections of code.

Table B-2 (Cont.) OC4J Command-Line Options

Command-Line Options	Description
-verbosity	Define an integer between 1 and 10 to set the verbosity level of the message output. Example: <code>-verbosity 10</code> . See Example 3-4 for an example of this option.
-version	Prints the version and exits.
-? -help	Prints the help message.

Table B-3 -D General System Properties for OC4J

-D Option	Description
<code>java.home</code>	Sets the <code>JAVA_HOME</code> environment variable, which points to the JDK to use for the OC4J instance.
<code>java.ext.dirs</code>	Sets the external directories to be searched for classes when compiling.
<code>java.io.tmpdir= <new_tmp_dir></code>	Default is <code>/tmp/var</code> . To change the temporary directory for the deployment wizard. The deployment wizard uses 20 MB in swap space of the temp directory for storing information during the deployment process. At completion, the deployment wizard cleans up the temp directory of its additional files. However, if the wizard is interrupted, it may not have the time or opportunity to clean up the temp directory. Thus, you must clean up any additional deployment files from this directory yourself. If you do not, this directory may fill up, which will disable any further deployment. If you receive an Out of Memory error, check for space available in the temp directory.
<code>KeepIIOPCode= true/false</code>	Default is false. If true, keeps the generated IIOP stub/tie code.
<code>oracle.arraylist.deepCopy= true/false</code>	If true, then while cloning an array list, a deep copy is performed. If false, a shallow copy is performed for the array list. Default: true
<code>dedicated.rmicontext= true/false</code>	Default is false. This replaces the deprecated <code>dedicated.connection</code> setting. When two or more clients in the same process retrieve an <code>InitialContext</code> , OC4J returns a cached context. Thus, each client receives the same <code>InitialContext</code> , which is assigned to the process. Server lookup, which results in server load balancing, happens only if the client retrieves its own <code>InitialContext</code> . If you set <code>dedicated.rmicontext=true</code> , then each client receives its own <code>InitialContext</code> instead of a shared context. When each client has its own <code>InitialContext</code> , then the clients can be load balanced. This parameter is for the client. You can also set this in the JNDI properties.
<code>associateUsingThirdTable= true/false</code>	For container-managed relationships in entity beans, you can designate if a third database table is used to manage the relationship. Set to false if you do not want a third association table. Default is true. See the "Entity Relationship Mapping" chapter in the <i>Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide</i> for more information.

Table B-3 (Cont.) -D General System Properties for OC4J

-D Option	Description
DefineColumnType= true/false	<p>DefineColumnType=true/false. The default is false. Set this to true if you are using an Oracle JDBC driver that is prior to 9.2. For these drivers, setting this variable to true avoids a round-trip when executing a select over the Oracle JDBC driver. This parameter should be set on the OC4J server.</p> <p>When you change the value of this option and restart OC4J, it is only valid for applications deployed after the change. Any applications deployed before the change are not affected.</p> <p>When true, the DefineColumnType extension saves a round trip to the database that would otherwise be necessary to describe the table. When the Oracle JDBC driver performs a query, it first uses a round trip to a database to determine the types that it should use for the columns of the result set. Then, when JDBC receives data from the query, it converts the data, as necessary, as it populates the result set. When you specify column types for a query with the DefineColumnType extension set to true, you avoid the first round trip to the Oracle database. The server, which is optimized to do so, performs any necessary type conversions.</p>
oracle.mdb.fastUndeploy=<int>	<p>The oracle.mdb.fastUndeploy system property enables you to shut down OC4J cleanly when you are running MDBs in a Windows environment or when the backend database is running on a Windows environment. Normally, when you use an MDB, it is blocked in a receive state waiting for incoming messages. However, if you shut down OC4J while the MDB is in a wait state in a Windows environment, then the OC4J instance cannot be stopped and the applications are not undeployed since the MDB is blocked. However, you can modify the behavior of the MDB in this environment by setting the oracle.mdb.fastUndeploy system property. If you set this property to an integer, then when the MDB is not processing incoming messages and in a wait state, the OC4J container goes out to the database (requiring a database round-trip) and polls to see if the session is shut down. The integer denotes the number of seconds the system waits to poll the database. This can be expensive for performance. If you set this property to 60 seconds, then every 60 seconds, OC4J is checking the database. If you do not set this property and you try to shut down OC4J using CTRL-C, the OC4J process will hang for at least 2.5 hours.</p>
oracle.dms.sensors=[none, normal, heavy, all].	<p>You can set the value for Oracle9iAS built-in performance metrics to the following: none (off), normal (medium amount of metrics), heavy (high number of metrics), or all (every possible metric). The default is normal. This parameter should be set on the OC4J server. The previous method for turning on these performance metrics, oracle.dms.gate=true/false, is replaced by the oracle.dms.sensors variable. However, if you still use oracle.dms.gate, then setting this variable to false is equivalent to setting oracle.dms.sensors=none.</p>

Table B-4 -D System Properties for Debugging

-D Debug System Properties	Description
ajp.debug	Default: false. If true, displays the AJP request (headers, mime types, URI etc) and response (status, error messages, and so on).
ajp.io.debug	Default: false. If true, displays the AJP post data, if any, and response data sent to the client.
KeepWrapperCode	Default: false. If true, keeps and debugs the generated wrapper code.

Table B-4 (Cont.) -D System Properties for Debugging

-D Debug System Properties	Description
<code>DBEntityHomeDebug</code>	Default: false. If true, displays entity bean home interface debug messages.
<code>DBEntityObjectDebug</code>	Default: false. If true, displays entity bean object debug messages.
<code>DBEntityWrapperDebug</code>	Default: false. If true, displays entity bean pool debug messages.
<code>iiop.runtime.debug</code>	Default: false. If true, outputs IIOP debug messages.
<code>NativeJDBCDebug</code>	Default: false. Native JDBC debug messages.
<code>http.cluster.debug</code>	Default: false. HTTP clustering debug messages.
<code>http.request.debug</code>	Default: false. If true, provides information about each HTTP request to standard output.
<code>http.redirect.debug</code>	Default: false. If true, provides information about each HTTP redirects to standard output.
<code>http.method.trace.allow</code>	Default: false. If true, turns on the <code>trace</code> HTTP method.
<code>http.session.debug</code>	Default: false. If true, provides information about HTTP session events
<code>http.error.debug</code>	Default: false. If true, prints all HTTP errors
<code>http.virtualdirectory.debug</code>	Default:false. If true, print the enforced virtual directory mappings upon startup.
<code>debug.http.contentLength</code>	Default: false. If true, print explicit content-length calls as well as extra <code>sendError</code> information.
<code>ejb.cluster.debug</code>	Default: false. EJB clustering debug messages.
<code>cluster.debug</code>	Default: false. Clustering debug messages.
<code>jms.debug</code>	Default: false. JMS debug messages.
<code>multicast.debug</code>	Default: false. Multicast debug messages.
<code>rmi.debug</code>	Default: false. RMI debug messages.
<code>transaction.debug</code>	Default: false. If true, prints debug messages for JTA events.
<code>rmi.verbose</code>	Default: false. RMI verbose information.
<code>datasource.verbose</code>	Default: false. If true, provides verbose information on creation of data source and connections using data sources and connections released to the pool, and so on,
<code>jdbc.debug</code>	Default: false. If true, provides very verbose information when JDBC calls are made
<code>ws.debug</code>	Default:false. If true, turns on Web services debugging
<code>javax.net.debug=[ssl all]</code>	If <code>ssl</code> , turns on SSL debugging. If <code>all</code> , turns on SSL debugging with verbose messages.

For more information about debugging properties, see "[OC4J Debugging](#)" on page 3-36.

Table B-5 *stdout/stderr Archive Management Properties*

Debug Property	Description
<code>stdstream.filesize= <max_file_size></code>	The maximum size any file in the archive will be allowed to grow to, in megabytes. Files are rotated when this maximum is reached.
<code>stdstream.filenumber= <max_files></code>	The maximum number of files to keep as archives. The oldest file will be automatically deleted when the limit is exceeded.
<code>stdstream.rotatetime= <HH:mm></code>	The time at which the log file will be rotated each day.

Third Party Licenses

This appendix includes a description of the Third Party Licenses for all the third party products included with Oracle Application Server.

Third-Party Licenses

Topics include:

- [Apache HTTP Server](#)

Apache HTTP Server

Under the terms of the Apache license, Oracle is required to provide the following notices. However, the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Apache software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the Apache software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Apache.

The Apache Software License

```

/* =====
* The Apache Software License, Version 1.1
*
* Copyright (c) 2000 The Apache Software Foundation. All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in
* the documentation and/or other materials provided with the
* distribution.
*
* 3. The end-user documentation included with the redistribution,
* if any, must include the following acknowledgment:
* "This product includes software developed by the
* Apache Software Foundation (http://www.apache.org/)."
* Alternately, this acknowledgment may appear in the software itself,

```

```
*   if and wherever such third-party acknowledgments normally appear.
*
* 4. The names "Apache" and "Apache Software Foundation" must
*   not be used to endorse or promote products derived from this
*   software without prior written permission. For written
*   permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called "Apache",
*   nor may "Apache" appear in their name, without prior written
*   permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED 'AS IS' AND ANY EXPRESSED OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
* USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
* =====
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation.  For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*
* Portions of this software are based upon public domain software
* originally written at the National Center for Supercomputing Applications,
* University of Illinois, Urbana-Champaign.
*/
```

Symbols

- <access-log> element, 3-35
- <alt-dd> element, B-15
- <application> element, B-7, B-15
- <application-client> element, B-22
- <application-server> element, B-6
- <argument> element, B-16
- <arguments> element, B-17
- <callback-handler> element, B-22
- <client-module> element, B-17
- <cluster> element, B-8
- <commit-class> element, B-17
- <commit-coordinator> element, B-17
- <compiler> element, B-8
- <connector> element, B-15
- <connectors> element, B-17
- <context-attribute> element, B-25
- <context-root> element, B-15
- <data-sources> element, B-17
- <description> element, B-17, B-22
- <display-name> element, B-15, B-22
- <ejb> element, B-15
- <ejb-link> element, B-22
- <ejb-module> element, B-17
- <ejb-ref> element, B-22
- <ejb-ref-mapping> element, B-25
- <ejb-ref-name> element, B-23
- <ejb-ref-type> element, B-23
- <env-entry> element, B-23
- <env-entry-mapping> element, B-25
- <env-entry-name> element, B-23
- <env-entry-type> element, B-23
- <env-entry-value> element, B-23
- <execution-order> element, B-8
- <file> element, 3-35, B-11, B-18
- <global-application> element, B-8
- <global-thread-pool> element, 3-29, B-9
- <global-web-app-config> element, B-9
- <group> element, B-18
- <home> element, B-23
- <icon> element, B-15, B-23
- <init-library> element, 3-25, 3-27, B-9
- <init-param> element, B-10
- <java>, B-15
- <javacache-config> element, B-10
- <java-compiler> element, B-8, B-10
- <jazn> element, B-18, B-20
- <jazn-web-app> element, B-18
- <jms-config> element, B-11
- <large-icon> element, B-15, B-23
- <library> element, 3-20, B-19
- <log> element, 3-33, 3-34, 3-35, B-11, B-19
- <lookup-context> element, B-25
- <mail> element, B-11, B-19
- <mail-session> element, B-20
- <max-http-connections> element, B-12
- <module> element, B-15
- <namespace-access> element, B-20
- <namespace-resource> element, B-20
- <odl> element, 3-33, B-11, B-19
- <odl-access-log> element, 3-33
- <orion-application> element, B-16
- <orion-application-client> element, B-24
- <password-manager> element, B-20
- <persistence> element, B-20
- <principals> element, B-20
- <property> element, B-20
- <read-access> element, B-20
- <remote> element, B-23
- <res-auth> element, B-23
- <resource-env-ref> element, B-24
- <resource-env-ref-mapping> element, B-25
- <resource-env-ref-name> element, B-24
- <resource-env-ref-type> element, B-24
- <resource-provider> element, B-21
- <resource-ref> element, B-24
- <resource-ref-mapping> element, B-26
- <res-ref-name> element, B-24
- <res-sharing-scope> element, B-24
- <res-type> element, B-24
- <rmi-config> element, B-13
- <role-name> element, B-16
- <security-role> element, B-16
- <security-role-mapping> element, B-21
- <sep-config> element, B-13
- <servlet> element, 3-6
- <sfsb-config> element, B-13
- <shutdown-class> element, 3-27, B-14
- <shutdown-classes> element, 3-27, B-14
- <small-icon> element, B-16, B-24
- <startup-class> element, 3-25, B-14

- <startup-classes> element, 3-25, B-14
- <transaction-config> element, B-14
- <user> element, B-21
- <user-manager> element, B-21
- <web> element, B-16
- <web-module> element, B-21
- <web-site> element, B-14
- <web-uri> element, B-16
- <write-access> element, B-22

A

- access logging
 - disabling, 3-21
- administration, 2-3
- admin.jar tool
 - undeployment, 2-21
- AJP
 - overview, 1-7
- ajp.debug property, B-33
- ajp.io.debug property, B-33
- ANT, 2-11
- Apache
 - Oracle HTTP Server, 1-4
- Apache JServ protocol, see AJP
- application
 - deployment, 2-11
 - example, 2-6
 - undeployment, 2-21
- application-client.xml file
 - element description, B-22
 - example, B-29
- application.xml file, 2-9
 - element description, B-15
 - example, B-26
- associateUsingThirdTable property, B-32
- authentication, 3-10

C

- CLASSPATH, 1-7
- cluster.debug property, B-34
- clustering, 4-1 to ??
 - configuration, 4-7
 - configure replication, 4-9
 - configuring islands, 4-8
 - configuring OC4J instance, 4-1
 - configuring OC4J processes, 4-8
 - EJB applications, 4-4, 4-10
 - OC4J instance, 4-2
 - replicating application state, 4-3
 - tuning parameters, 4-5, 4-13
 - Web applications, 4-4, 4-9
- command-line options, 2-21, 3-4, B-30
 - performance settings, 3-28
- compiler
 - specifying, B-10
- configuration
 - application.xml file, 2-9
 - default, 1-4, 2-4

- server.xml file, 2-13
- Web context, 3-22

D

- datasource.verbose property, B-34
- DBEntityHomeDebug property, B-34
- DBEntityObjectDebug property, B-34
- DBEntityWrapperDebug property, B-34
- DCM
 - overview, 1-6
- dcmctl
 - DCM utility, 1-6
- debugging, 3-36 to 3-39
- debug.http.contentLength property, B-34
- dedicated.connection setting, 3-28, B-32
- dedicated.rmicontext property, B-32
- dedicated.rmicontext setting, 3-28
- default-web-site.xml file, 3-20
 - example, B-29
- DefineColumnType property, 3-28, B-33
- deployment
 - applications, 2-11
 - error recovery, 2-21
- development
 - recommendations, 2-5
- Distributed Configuration Management, see DCM

E

- EAR file
 - creation, 2-12
 - structure, 2-11
 - used in deployment, 2-11
- EJB
 - deployment, 2-11
 - replication, 4-12
- ejb.cluster.debug property, B-34
- ejb-jar.xml file
 - example, B-27
- enable-passivation attribute, B-13
- Enterprise JavaBeans, see EJB
- environment
 - modifications, 2-10
- environment variables, 3-4

F

- front-end listener
 - Oracle HTTP Server, 1-4

H

- hashtable, B-29
- hot deployment, 2-19
- HTTP method
 - trace, 3-37, B-34
- http.cluster.debug property, B-34
- http.error.debug property, B-34
- http.method.trace.allow property, 3-37, B-34
- http.redirect.debug property, B-34

http.request.debug property, 3-37, B-34, B-35
http.session.debug property, B-34
http.virtualdirectory.debug property, B-34

I

iiop.runtime.debug property, B-34
InitialContext, 3-28, B-32
installation
 requirements, 1-7

J

J2EE
 definition, 1-1
Java
 command-line options, 3-4
Java Platform Debugging Architecture, see JPDA
JAVA_HOME variable, 2-10
java.ext.dirs property, B-11, B-32
java.home property, B-32
java.io.tmpdir property, B-32
javax.net.debug property, B-34
jdbc.debug property, B-34
JDK, 1-1
JDK 1.4 considerations, 1-2
Jikes, B-8
JMS, B-3
jms.debug property, B-34
JPDA, 3-39
JSP pages
 deployment, 2-11
JVM, 1-1

K

KeepIIOPCode property, B-32
KeepWrapperCode property, B-33

L

library
 sharing, 3-19
logging, 3-31 to 3-36
 log files, 3-32, 3-35
 ODL, 3-32, B-11, B-19
 rollover logging, 3-32, B-11, B-19
 standard error, 3-35
 standard out, 3-35
 text, 3-34
 XML message format, 3-33

M

mod_oc4j module, 1-7
multicast.debug property, B-34

N

NativeJDBCDebug property, B-34

O

OC4J
 application example, 2-6
 clustering role, 4-2
 command-line options, 2-21, 3-4, B-30
 installation requirements, 1-7
 restarting, 2-3
 setup, 1-4
 shutdown class, 3-25
 startup, 2-3
 startup class, 3-25
 stopping, 2-3
 system properties, B-30
 testing, 2-4
OC4J command-line options, B-30
Oc4jMount directive, 3-22
OC4JShutdown interface, 3-27
OC4JStartup interface, 3-25
Oracle Diagnostic Logging, see logging
 ODL
Oracle HTTP Server
 front-end listener, 1-4
Oracle HTTP Server (OHS), 3-22
oracle.dms.gate setting, 3-28, B-33
oracle.dms.sensors setting, 3-28, B-33
oracle.mdb.fastUndeploy property, B-33
orion-application-client.xml file
 element description, B-24
 example, B-29
orion-application.xml file
 element description, B-16
Out of Memory error, 2-21, B-32

P

parent application, 2-12
 setting, 2-12
 XML definition, 2-13
performance
 oracle.dms.sensors setting, 3-28, B-33
performance binding
 command-line options, 3-28
 dedicated.connection, 3-28, B-32
 dedicated.rmicontext, 3-28, B-32
 DefineColumnType, 3-28, B-33
 oracle.dms.gate, 3-28, B-33
 statement caching, 3-31
 task manager granularity, 3-31, B-7
 thread pools, 3-29, B-9
performance settings, 3-28
postDeploy method, 3-25
postUndeploy method, 3-27
preDeploy method, 3-25
preUndeploy method, 3-27

R

RAR, 3-23
requirements
 software, 1-7

Resource Adapter Achieve, see RAR
restart OC4J, 2-3
RMI, B-3
rmi.debug property, B-34
rmi.verbose property, B-34

S

server.xml file, 2-13
 element description, B-6
 example, B-28
servlets
 deployment, 2-11
setStmtCacheSize method, 3-31
sharing libraries, 3-19
shutdown class, 3-27
 postUndeploy method, 3-27
 preUndeploy method, 3-27
standard error
 redirection, 3-35
standard out
 redirection, 3-35
startup class, 3-25 to 3-27
 example, 3-26
 postDeploy method, 3-25
 preDeploy method, 3-25
startup OC4J, 2-3
stateful session bean
 clustering, 4-12
statement caching
 DataSource
 statement caching, 3-31
stmt-cache-size attribute, 3-31
stop OC4J, 2-3
system properties, B-30

T

task manager granularity, 3-31, B-7
taskmanager-granularity attribute, 3-31, B-7
thread
 pooling, 3-29
transaction.debug property, B-34

U

undeployment, 2-21

W

Web
 application deployment, 2-11
 mount points, 3-22
Web context
 customization, 3-22
web.xml file
 example, B-27
ws.debug property, 3-37, B-34