Oracle® Containers for J2EE

Configuration and Administration Guide

10*g* Release 2 (10.1.3)

Part No. B14432-01

May 2005

Beta Draft

ORACLE®

Oracle Containers for J2EE Configuration and Administration Guide, 10*g* Release 2 (10.1.3) for Windows or UNIX

Part No. B14432-01

Primary Author: Dan Hynes

Contributing Authors: Brian Wright, Sheryl Maring

Contributors: Bryan Atsatt, Ellen Barnes, Julie Basu, Steve Button, Olivier Caudron, Jose Alberto Fernandez, Lars Ewe, Marcelo Goncalves, Sumathi Gopalakrishnan, Ping Guo, Hal Hildebrand, James Kirsch, Sunil Kunisetty, Clement Lai, Mike Lehmann, Phlippe Le Mouel, Angie Long, Sharon Malek, Sheryl Maring, Kuassi Mensah, Jasen Minton, Rama Notowidigdo, John O'Duinn, Debu Panda, Shiva Prasad, Chaya Ramanujam, Vinay Shukla, Sanjay Singh, Gael Stevens, Kenneth Tang, Helen Zhao, Frances Zhao, Serge Zloto

Contents

4 Runtime Configuration

5 Starting and Stopping OC4J

6 Using the admin.jar Command Line Utility

7 Clustering in OC4J

8 Logging in OC4J

9 Task Manager and Thread Pool Configuration

10 Using MBeans in OC4J

11 Managing Web Sites in OC4J

12 Registering DTDs and XSDs with OC4J

A Troubleshooting OC4J

B Configuration Files Used in OC4J

Send Us Your Comments

Oracle Containers for J2EE Configuration and Administration Guide, 10*g* Release 2 (10.1.3) for Windows or UNIX

Part No. B14432-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appserverdocs_us@oracle.com
- FAX: (650) 506-7225 Attn: Java Platform Group, Information Development Manager
- Postal service:

  Oracle Corporation
  Java Platform Group, Information Development Manager
  500 Oracle Parkway, Mailstop 4op9
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This documentation serves as the primary reference on configuring and managing an Oracle Containers for J2EE (OC4J) installation in a standalone environment. It essentially replaces the *Oracle Application Server Containers for J2EE User's Guide* and the *Oracle Application Server Containers for J2EE Standalone User's Guide* released with previous versions of OC4J.

This preface contains the following sections:

Intended Audience

This documentation is intended for the following audiences:

- A systems administrator responsible for configuring and managing an OC4J installation
- A Java application developer using OC4J in a standalone environment

The documentation assumes that readers are already familiar with the following:

- The Java 2 Platform, Enterprise Edition (J2EE) environment
- General server and system administration concepts
- General Web technology
- The Java programming language

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. 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 additional information, visit the Oracle Accessibility Program Web site at

http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, 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, JAWS 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.

Structure

This document contains the following chapters:

Chapter 1, "Introducing OC4J"
This chapter provides a general introduction to Oracle Containers for J2EE, or OC4J.

Chapter 2, "Installing Standalone OC4J"
This chapter describes the prerequisites and process installing for the OC4J standalone distribution.

Chapter 3, "Administering OC4J"
This chapter provides instructions for starting, stopping and restarting an OC4J instance.

Chapter 4, "Runtime Configuration"
This chapter provides details on runtime options and system properties that can be set at OC4J startup.

Chapter 5, "Starting and Stopping OC4J"
This chapter provides instructions for installing OC4J, as well as for starting, stopping and restarting an OC4J instance.

Chapter 6, "Using the admin.jar Command Line Utility"
This chapter provides instructions on using the command-line utility called `admin.jar` that can be used to perform operations on a running OC4J instance.

Chapter 7, "Clustering in OC4J"
This chapter discusses the clustering framework provided in OC4J.

Chapter 8, "Logging in OC4J"
This chapter provides instructions on using the system and application logging features available in OC4J.

Chapter 9, "Task Manager and Thread Pool Configuration"

This chapter provides guidelines for configuring the task manager and thread pool management features for an OC4J instance.

Chapter 10, "Using MBeans in OC4J"

This chapter provides an overview of the MBeans provided with OC4J can be used to manage deployed applications, services and other resources.

Chapter 11, "Managing Web Sites in OC4J"

This chapter explains how additional Web sites can be created in OC4J to provide access to deployed Web applications. It also explains how to configure and enable a secure Web site.

Chapter 12, "Registering DTDs and XSDs with OC4J"

This chapter describes the process for registering new entities - specifically any vendor-specific DTDs and XSDs used to define the format of XML deployment descriptors - within OC4J, which is required if XML file validation will be performed.

Appendix A, "Troubleshooting OC4J"

This appendix describes common problems that you may encounter when using OC4J and explains how to resolve them.

Appendix B, "Configuration Files Used in OC4J"

This chapter provides detailed documentation on the XML files used to store configuration data for the OC4J server and J2EE applications and modules deployed into it.

Appendix C, "Overview of the Session State Tables"

This appendix documents the schema for the database tables uses by the OC4J database persistence mechanism to store session state.

Related Documents

For more information, see these Oracle resources available from the Oracle Java Platform Group:

- ***Oracle Containers for J2EE Deployment Guide***

  This book provides instructions on deploying, undeploying and redeploying J2EE-compliant applications and standalone modules to an OC4J instance. Extensive instructions on using the various deployment mechanisms provided with OC4J are provided, including the new JSR 88-compliant "deployment plan editor", incremental EJB deployment and deployment-specific Ant tasks.

- *Oracle Containers for J2EE Services Guide*

  This book provides information about standards-based Java services supplied with OC4J, such as JTA, JNDI, JMS, JAAS, and the Oracle Application Server Java Object Cache.

- *Oracle Containers for J2EE Security Guide*

  This document (not to be confused with the *Oracle Application Server 0g Security Guide*), describes security features and implementations particular to OC4J. This includes information about using JAAS, the Java Authentication and Authorization Service, as well as other Java security technologies.

Also available from the Oracle Java Platform group:

- *Oracle9i Java Developer's Guide*
- *Oracle9i Java Stored Procedures Developer's Guide*
- *Oracle9i JDBC Developer's Guide and Reference*
- *Oracle9i JPublisher User's Guide*

Available from the Oracle Application Server group:

- *Oracle Application Server Administrator's Guide*
- *Oracle Application Server Security Guide*
- *Oracle Application Server Performance Guide*
- *Oracle Enterprise Manager Concepts*
- *Oracle HTTP Server Administrator's Guide*
- *Oracle Application Server Globalization Guide*
- *Oracle Application Server Web Cache Administrator's Guide*
- Oracle Containers for J2EE Web Services Developer's Guide
- *Oracle Application Server Upgrade and Compatibility Guide*

Available from the Oracle JDeveloper group:

- Oracle JDeveloper online help
- Oracle JDeveloper documentation on the Oracle Technology Network:

  `http://otn.oracle.com/products/jdev/content.html`

Available from the Oracle Server Technologies group:

- *Oracle XML Developer's Kit Programmer's Guide*
- *Oracle XML API Reference*
- *Oracle9i Application Developer's Guide - Fundamentals*
- *Oracle9i Supplied PL/SQL Packages and Types Reference*
- *PL/SQL User's Guide and Reference*
- *Oracle9i SQL Reference*
- *Oracle9i Net Services Administrator's Guide*
- *Oracle Advanced Security Administrator's Guide*
- *Oracle9i Database Reference*

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/membership/`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/documentation`

The following OTN Web site for Java servlets and JavaServer Pages is also available:

`http://otn.oracle.com/tech/java/servlets/`

The following resources are available from Sun Microsystems.

- Web site for JavaServer Pages, including the latest specifications:

  `http://java.sun.com/products/jsp/index.html`

- Web site for Java Servlet technology, including the latest specifications:

  `http://java.sun.com/products/servlet/index.html`

- `jsp-interest` discussion group for JavaServer Pages

  To subscribe, send an e-mail to `listserv@java.sun.com` with the following line in the body of the message:

  `subscribe jsp-interest` *`yourlastname yourfirstname`*

  It is recommended, however, that you request only the daily digest of the posted e-mails. To do this add the following line to the message body as well:

  `set jsp-interest digest`

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
Italics	Italic typeface indicates book titles or emphasis, or terms that are defined in the text.
`Monospace (fixed-width) font`	Monospace typeface within text indicates items such as executables, file names, directory names, Java class names, Java method names, variable names, other programmatic elements (such as JSP tags or attributes, or XML elements or attributes), or database SQL commands or elements (such as schema names, table names, or column names).
`Italic monospace (fixed-width) font`	Italic monospace font represents placeholders or variables.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
|	A vertical bar represents a choice of two or more options. Enter one of the options. Do not enter the vertical bar.

1

Introducing OC4J

This chapter provides a general introduction to Oracle Containers for J2EE, or OC4J. It includes the following sections:

What Is OC4J?

Oracle Containers for J2EE 10*g* Release 2 (10.1.3), or OC4J, provides a complete Java 2 Enterprise Edition (J2EE) 1.4-compliant environment. OC4J provides all the containers, APIs, and services mandated by the J2EE specification.

OC4J is distributed in two configurations:

- A **standalone** configuration, in which OC4J is installed as a single, "standalone" instance and is managed, started and stopped directly as a self-contained component.

  See "What Is OC4J in a Standalone Configuration?" on page 1-3 for details on this configuration.

- A **managed** configuration, in which OC4J is installed and managed as a compenent of Oracle Application Server. At a minimum, a managed OC4J installation will include Oracle Process Manager and Notification Server (OPMN) which manages the various Oracle Application Server components, including OC4J. An installation will also include at least one Oracle HTTP Server (OHS) instance, which provides Web communication and load balancing functionality.

  See "What Is OC4J in an Oracle Application Server Configuration?" on page 1-4 for details.

OC4J is based on technology licensed from Ironflare Corporation, which develops the Orion server, one of the leading J2EE containers. As such, the product still contains some reference to the Orion server, including proprietary deployment descriptors that refer to "orion".

OC4J is written entirely in Java and executes on the Java Virtual Machine (JVM) of the standard Java Development Kit (JDK). You can run OC4J on the standard JDK that exists on your operating system.

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.

What's New in OC4J

Oracle Containers for J2EE Release 2 (10.1.3) includes a number of new features and enhancements, as described below.

Support for Web Services

OC4J provides full support for Web services in accordance with the J2EE 1.4 standard, including JAX-RPC 1.1. Web services interoperability is also supported.

- EJB 2.1 Web services end point model
- JSR 109 client and server deployment model
- CORBA Web services: Support for wrapping existing basic CORBA Servants as Web services and auto-generating WSDL from IDL
- Support for source code annotations to customize Web services behavior such as invocation and ending styles (RPC/literal, RPC/encoded, Doc/literal); customizing the Java to XML mapping; enforcing security.
- Database and JMS Web services

Support for New J2EE 1.4 Application Management and Deployment Specifications

OC4J supports the following specifications defining new standards for deploying and managing applications in a J2EE environment.

- The *Java Management Extensions (JMX) 1.2* specification, which allows standard interfaces to be created for managing resources, such as services and applications, in a J2EE environment. The OC4J implementation of JMX provides a JMX client that can be used to completely manage an OC4J server and applications running within it.
- The *J2EE Management Specification (JSR-77)*, a specification that allows standard components to be created for managing applications in a J2EE environment.
- The *J2EE Application Deployment API (JSR-88)*, which defines a standard API for configuring and deploying J2EE applications and modules into a J2EE-compatible environment. The OC4J implementation includes the ability to create and/or edit a deployment plan containing the OC4J-specific configuration data needed to deploy a component into OC4J.

Support for Enterprise JavaBeans 3.0

OC4J provides support for the Enterprise JavaBeans 3.0 specification, including the new program annotation functionality. Note that OC4J must use JDK 5.0 to enable EJB 3.0 support.

Support for Oracle Application Server TopLink

Oracle Application Server TopLink is an advanced, object persistence framework for use with a wide range of Java 2 Enterprise Edition (J2EE) and Java application architectures. OracleAS TopLink includes support for the OC4J Container Managed Persistence (CMP) container and base classes that simplify Bean Managed Persistence (BMP) development.

OracleAS Job Scheduler

The OracleAS Job Scheduler provides asynchronous scheduling services for J2EE applications. Its key features include capabilities for submitting, controlling and monitoring *jobs*, defined as a unit of work that executes when the work is performed.

New Two-Phase Commit Transaction Coordinator Functionality

The new Distributed Transaction Manager in OC4J can coordinate two-phase transactions between any type of XA resource, including databases from Oracle as well as other vendors and JMS providers such as IBM WebsphereMQ. Automatic transaction recovery in the event of a failure is also supported.

Generic JMS Resource Adapter Enhancements

The Generic JMS Resource Adapter can now be used as an OC4J plug-in for OracleAS JMS that ships with the current version of OC4J as well as for IBM WebSphere MQ JMS version 5.3.

Support for lazy transaction enlistment has been added so that JMS connections can be cached and still be able to correctly participate in global transactions.

Finally, the Generic JMS Resource Adapter now has better error handling. Endpoints now automatically retry after provider or system failures, and `onMessage()` errors are handled correctly.

Features of OC4J

The following features are provided with the latest release of OC4J.

J2EE Support

OC4J supports and is certified for the standard J2EE APIs, as listed in Table 1–1

Table 1–1 OC4J J2EE Support

J2EE Standard APIs	Version Supported By OC4J
JavaServer Pages (JSP)	2.0
Servlets	2.4
Enterprise JavaBeans (EJB)	2.1
Java Transaction API (JTA)	1.0
Java Message Service (JMS)	1.1
Java Naming and Directory Interface (JNDI)	1.2
Java Mail	1.2
Java Database Connectivity (JDBC)	2.0 Extension
Oracle Application Server Java Authentication and Authorization Service (JAAS) Provider	1.0
J2EE Connector Architecture	1.5
Java API for XML-Based RPC (JAX-RPC)	1.1
SOAP with Attachments API for Java (SAAJ)	1.2
Java API for XML Processing (JAXP)	1.1
Java API for XML Registries (JAXR)	1.0.5

What Is OC4J in a Standalone Configuration?

The standalone or "unmanaged" OC4J configuration offers a robust J2EE-compliant container that is easy to administer. The standalone configuration is comprised of the following components:

- Oracle Containers for J2EE 10*g* Release 2 (10.1.3)
- Oracle Enterprise Manager 10*g* Application Server Control Console, a Web-based administration application installed by default with OC4J

In this configuration, a single OC4J instance is installed into a single `ORACLE_HOME` - the root directory in which Oracle software is installed. The Application Server Control Console is enabled immediately upon installation.

Installation

The standalone OC4J distribution, which includes Application Server Control Console, is provided as a ZIP archive. See Chapter 2, "Installing Standalone OC4J" for instructions.

Administration

The OC4J instance is administered as a standalone component, using either the Application Server Control Console installed with the instance or the built-in `admin.jar` or `oc4j.cmd/oc4j.sh` command line utilities. See "Tools for Administering OC4J" on page 3-1 for an overview of these tools.

Starting/Stopping

In a standalone configuration, an OC4J instance is started using either the `oc4j.jar` command line or the `oc4j.cmd/oc4j.sh` command line utilities. Startup options and system properties can also be set at startup on the `oc4j.jar` command line.

Backup, Restore and Disaster Recovery Capabilities

The OC4J standalone configuration does not have backup, restore and disaster recovery capabilities.

Web Communication

Web communication in a standalone environment is provided through the built-in OC4J Web server, which supports HTTP and HTTPS communications natively without the use of the Oracle HTTP Server (OHS).

The default Web site is defined in the `http-web-site.xml` file, which specifies the default HTTP listener on port `8888`. Additional Web sites may be defined on different ports using variations of this file. See Chapter 11, "Managing Web Sites in OC4J" for instructions on creating additional Web sites in OC4J.

What Is OC4J in an Oracle Application Server Configuration?

In this configuration, OC4J is installed with other components of Oracle Application Server. At minimum, this configuration includes the following components:

- Oracle Containers for J2EE 10*g* Release 2 (10.1.3)
- Oracle Enterprise Manager 10*g* Application Server Control Console, a Web-based administration application installed by default with OC4J
- Oracle HTTP Server (OHS) 1.3, which provides Web communication and load balancing functionality
- Oracle Process Manager and Notification Server (OPMN), used to start/stop and monitor the other components

Oracle Application Server provides support for HTTP session and stateful session Enterprise JavaBean replication and load balancing across a cluster of OC4J instances. See Chapter 7, "Clustering in OC4J" for details.

OC4J supports a number of managed configurations, including:

- **Single Machine Oracle Application Server**

  In this configuration, all components - OC4J, OHS and OPMN - are installed into a single `ORACLE_HOME` directory.

  Multiple OC4J instances can be created within this `ORACLE_HOME`. Multiple host machines, each hosting one or more OC4J instances, can be included in an Oracle Application Server cluster.

- **Multi-Machine Oracle Application Server**

  In this configuration, the OC4J and OHS components are installed in different `ORACLE_HOME` directories, possibly on different machines.

Installation

Installation of the various components is done using the Oracle Universal Installer. Note that OPMN must be installed in every `ORACLE_HOME` directory to enable monitoring of each installed component.

Administration

All administration tasks are performed using the Web-based Application Server Control Console user interface. In a clustered environment, a single Application Server Control Console can be used to manage all OC4J instances in the cluster.

See "Oracle Enterprise Manager 10g Application Server Control Console" on page 3-1 for details on this application.

Starting/Stopping

In a managed environment, OPMN is used to start and stop all components, including OC4J. See the *Oracle Process Manager and Notification Server Administrator's Guide* for instructions on configuring and using OPMN.

OC4J runtime options and system properties can be manually set in the OPMN configuration file, `opmn.xml`.

Backup, Restore and Disaster Recovery Capabilities

These capabilities are available with the managed Oracle Application Server configuration.

Web Communication

Web communication in Oracle Application Server is provided through Oracle HTTP Server (OHS), which serves a a front-end listener, and the `mod_oc4j` module, which forwards requests to OC4J server instances using the Apache JServ Protocol (AJP) 1.3.

Figure 1–1 illustrates the flow as follows:

1. An incoming HTTP request is received by the OHS listener.
2. OHS passes the request to an OC4J server instance via the `mod_oc4j` module. The connection between the OHS and OC4J uses the Apache JServ Protocol (AJP) on a port number negotiated during OC4J startup.

Figure 1–1 OC4J Web Communication in Oracle Application Server

Mount points are created in `mod_oc4j` for applications deployed into an OC4J instance. Requests that come in for specific mount points are routed to the OC4J instance corresponding to that mount point.

For additional information on configuring and managing OHS and the `mod_oc4j` module, see the *Oracle HTTP Server Administrator's Guide*.

2

Installing Standalone OC4J

This chapter describes the prerequisites and process for installing the OC4J standalone distribution, which is distributed as the as `oc4j_extended.zip` archive.

See the ***Oracle Application Server Installation Guide*** for instructions on installing OC4J in an Oracle Application Server configuration.

The following topics are covered in this chapter:

Standalone OC4J Installation Prerequisites

Ensure the following prerequisites are met before installing OC4J.

Install the JDK

Before installing OC4J, you must first install Java 2 Platform, Standard Edition (J2SE) Development Kit (JDK) release 1.4.2 or 5.0 on the OC4J host machine. You can download the JDK release from `http://java.sun.com/j2se/`.

Set Environment Variables

After installing J2SE, ensure that the following environment variables are set:

Table 2–1 Environment Variable Settings

Environment Variable	Value
`JAVA_HOME`	Set to the location of the JDK. This variable is required to start the OC4J server. Note that the JDK that will be used must be added to the host machine's PATH.
`ORACLE_HOME`	Set to the root directory into which you will install the OC4J distribution. Defining this variable is required if you intend to run the `oc4j.cmd` or `oc4j.sh` executable scripts. For example, if you install OC4J into `C:\oracle`, set the value of the `ORACLE_HOME` variable to this directory.
`J2EE_HOME`	Optionally create and set this variable to `<oc4j_install_dir>/j2ee/home`, the installed location of `oc4j.jar` and `admin.jar`. Setting this variable will allow you to invoke these JAR files from any directory.

Installing the Standalone OC4J Distribution

Install the standalone OC4J distribution by extracting the `oc4j_extended.zip` file into the directory that will serve as the OC4J installed directory - referenced in this documentation as `ORACLE_HOME` - using the archive utility of your choice. specified. The installer automatically creates the required directory structure for you, as shown below.

```
<ORACLE_HOME>
   /bin
   /j2ee
   /javacache
   /javavm
   /jdbc
   /jdk
   /jlib
   /lib
   /rdbms
   /soap
   /sqlj
   /toplink
   /webservices
```

You will be prompted to set a password for the OC4J Administrator account the first time OC4J is started The username for this account is set to `oc4jadmin` by default.

The OC4J standalone distribution is installed with a default configuration that includes a default Web site where applications can be accessed, and a Web site that allows the Application Server Control management interface to be used. These are provided so that you can start using OC4J immediately. See Chapter 11, "Managing Web Sites in OC4J" for additional information.

3

Administering OC4J

This chapter provides an overview of the administrative capablities provided with OC4J. It includes the following sections:

Tools for Administering OC4J

The following tools for managing an OC4J instance in a standalone environment are installed with the OC4J server.

Oracle Enterprise Manager 10*g* Application Server Control Console

The *Oracle Enterprise Manager 10g Application Server Control Console* is a JMX-compliant, Web-based user interface for deploying, configuring and monitoring applications within OC4J, as well as managing the OC4J server instance and the Web services used by your applications.

The Application Server Control Console is installed and configured automatically when you install the OC4J software, and is available via port `1810` by default. To access the console, simply type the following URL in a Web browser:

```
http://<hostname>:1810
```

See the online Help provided with Application Server Control Console for detailed instructions on using this interface.

The Application Server Control Console is organized into several functional areas, described below.

Note: The current release of Application Server Control Console does not provide management support for either OPMN or Oracle HTTP Server. Use the OPMN command-line tool, `opmnctl`, to start/stop and manage instances of these components.

Applications

- Start/stop applications, modules or standalone resource adapters deployed into the OC4J instance
- Deploy, undeploy or redeploy an application or module
- Create or edit a deployment plan as part of deploying an application
- View statistics on HTTP requests and active EJB method calls

Administration

- Manage J2EE services, including JMS and JTA
- View and search for such JNDI names
- Create JDBC data sources and connection pools providing database access
- Set JSP container properties
- Configure security providers and manage users and roles
- Access MBeans through the JMX MBean browser
- Subscribe to event-driven JMX notifications

Performance

- View graphs showing usage of CPU and memory resources by OC4J versus other active applications, as well as OC4J heap usage
- View statistics on database connections and transaction activity, JVM usage, JSP and servlet requests and EJB methods
- Query system for most-requested JSPs, servlets and EJBs

Web Services

- Enable or disable a web service
- View metrics and statistics for Web services running within the instance
- View the WSDL for a web service
- Test a web service
- Configure auditing, logging, reliability and security for a web service

Logs

- View log files for specific applications deployed into the OC4J instance
- View logs for the default (global) application and Application Server Control Console
- Search logs for specific message types and strings
- View XML formatted log files for components using the Oracle Diagnostic Logging (ODL) framework
- Retrieve Web service logs

See Chapter 8, "Logging in OC4J" for more on the logging capabilities provided by OC4J.

The admin.jar Command Line Utility

OC4J provides a command-line utility called `admin.jar` that can be used to perform operations on an active OC4J instance. Among other things, you can use this utility to:

- Shut down and restart a standalone OC4J instance
- Restart a specific application
- Deploy or undeploy applications
- Add or remove a Web site
- Add, remove or test a global or application-specific data source

The utility is installed by default in `ORACLE_HOME/j2ee/home`. Note that OC4J must be started before this utility can be used. Also note that the utility cannot be used to start OC4J. See Chapter 6, "Using the admin.jar Command Line Utility" for instructions on using this tool.

The oc4j/oc4j.cmd Executable Scripts

The OC4J distribution includes executable scripts - a shell script for the Unix/Linux platforms and a batch file for the Windows platform - that can be used in an OC4J standalone configuration to start and stop a local OC4J instance, get the OC4J version, and complete the OC4J installation process.

The `oc4j` executable scripts are located in the `ORACLE_HOME/bin` directory. The scripts are platform-specific:

- Use the `oc4j` shell script on Unix and Linux platforms.
- Use the `oc4j.cmd` batch file on Windows platforms.

Both executables use the same syntax, which is as follows:

```
oc4j [options]

oc4j.cmd [options]
```

The set of options that can be passed to the executables is identical for both, as summarized below.

Table 3–1 Options for oc4j and oc4j.cmd

Option	Description
`-start [-ejb3]`	Starts the OC4J server instance. Optionally include the `-ejb3` switch to enable EJB 3.0 support.
`-shutdown -port <oc4jORMIPort> -password <password>`	Stops the OC4J instance. Note that `-port <oc4jORMIPort>`: You do not need to specify the port if OC4J is running on the default ORMI port, which is `23791`. `-password <password>`: Specify the OC4J Administrator password.
`-version`	Returns the OC4J version number.

Oracle Process Manager and Notification Server (OPMN)

In a managed OC4J environment, OPMN is used to manage as well as start and stop all installed Oracle Application Server components, including all OC4J instances. OPMN also monitors OC4J and associated components, such as OHS. As a result,

OPMN must be installed into each ORACLE_HOME to monitor installed Oracle Application Server components. See the *Oracle Process Manager and Notification Server Administrator's Guide* for instructions on configuring and using OPMN.

A command-line utility, `opmnctl`, is used to invoke OPMN. The utility is installed by default in the `ORACLE_HOME/opmn/bin` directory on any machine hosting Oracle Application Server host components.

> **Note:** The current release of Application Server Control Console does not provide management support for either OPMN or Oracle HTTP Server. Use the OPMN command-line tool, `opmnctl`, to start/stop and manage instances of these components.

OPMN is configured through the `opmn.xml` configuration file, which is located in the `ORACLE_HOME/opmn/conf` directory. All edits to this file must be made by hand, as the current release of Application Server Control Console does not provide a file editing capability.

The following is an abridged example of how OC4J configuration data is structured in the `opmn.xml` configuration file.

- Configuration data for each component is set in an `<ias-component>` element, where the `id` attribute equals the component name.; in this case, `OC4J`.
- Each individual OC4J instance created on the host machine is configured within a `<process-type>` element. The `id` attribute uniquely identifies the instance.
- The `<process-set>` element defines a group of OC4J processes created at startup.

  The value of the `id` attribute identifies the group and is appended to log files generated for processes within the group to aid in management. The value of the `numprocs` attribute specifies the number of OC4J processes, each running within a separate JVM, to spawn for the OC4J instance.

The data applied to each OC4J instance is in turn enclosed in The following is an abridged example of the OC4J configuration data structure in `opmn.xml`:

```
<ias-component id="OC4J">
  <process-type id="home" module-id="OC4J" status="enabled">
    <module-data>
      <category id="start-parameters">
        <data id="java-options" value="-ejb3=true -Djava.awt.headless=true"/>
        <data id="java-bin" value="/jdk/bin"/>
        <data id="oc4j-options" value="-validateXML -verbosity 10"/>
      </category>
      <category id="stop-parameters">
        <data id="java-options" value="-Djava.awt.headless=true"/>
      </category>
    </module-data>
    <start timeout="600" retry="2"/>
    <stop timeout="120"/>
    <restart timeout="720" retry="2"/>
    <port id="default-web-site" protocol="ajp" range="12501-12600"/>
    <port id="rmi" range="12401-12500"/>
    <port id="jms" range="12601-12700"/>
    <process-set id="default" numprocs="5"/>
  </process-type>
</ias-component>
```

4

Runtime Configuration

This chapter provides details on runtime options and system properties that can be set at OC4J startup. It includes the following topics:

- Specifying the JDK Version
- Setting OC4J Runtime Options at Startup
- Setting System Properties at Startup

Specifying the JDK Version

OC4J requires the Java 2 Platform, Standard Edition (J2SE) Development Kit (JDK) release 1.4.2 or 5.0. You can specify the JDK version to use for a standalone OC4J configuration, as well as for each OC4J instance in an Oracle Application Server installation.

Specifying the JDK in a Standalone Configuration

In a standalone OC4J configuration, set the JAVA_HOME environment variable to the location of the JDK you with to use.

Note that the JDK that will be used must be added to the host machine's PATH.

Specifying the JDK in a Managed Configuration

When OC4J is installed as a component of Oracle Application Server, you can specify the JDK to use for each OC4J instance through manual edits to the `opmn.xml` configuration file.

Set Java system properties in the `<data>` element where the `id` attribute is `"java-bin"`. This `<data>` element is enclosed within the `<category id="start-parameters">` subelement of the `<ias-component id="OC4J">` element in the XML structure. For example:

```
<ias-component id="OC4J">
  <process-type id="home" module-id="OC4J" status="enabled">
    <module-data>
      <category id="start-parameters">
        <data id="java-bin" value="/jdk/bin"/>
      </category>
      ...
    </module-data>
  </process-type>
</ias-component>
```

Setting OC4J Runtime Options at Startup

A number of OC4J runtime options can be set on OC4J instances at OC4J startup, most providing options for managing standard output messages. How these options are set differs for standalone OC4J and managed Oracle Application Server configurations.

Setting Runtime Options in a Standalone OC4J Configuration

OC4J runtime options can be set by passing arguments on the `oc4j.jar` command line at OC4J startup. The syntax for `oc4j.jar` is as follows:

```
java [props] -jar oc4j.jar [args]
```

Note that runtime options (`[args]`) are specified after `oc4j.jar` in the syntax. For example:

```
java -jar oc4j.jar -validateXML -verbosity 10
```

Setting Runtime Options in a Managed OC4J Configuration

When OC4J is installed as a component of Oracle Application Server, OC4J runtime options must be manually added to the `opmn.xml` configuration file. Options will be passed to managed OC4J instances at startup.

Set Java system properties in the `<data>` element where the `id` attribute is `"oc4j-options"`. This `<data>` element is enclosed within the `<category id="start-parameters">` subelement of the `<ias-component id="OC4J">` element in the XML structure. Preface all system properties with a `-D`. For example:

```
<ias-component id="OC4J">
  <process-type id="home" module-id="OC4J" status="enabled">
    <module-data>
      <category id="start-parameters">
        <data id="oc4j-options" value="-validateXML -verbosity 10"/>
        ...
      </category>
      ...
    </module-data>
  </process-type>
</ias-component>
```

Overview of OC4J Runtime Options

Table 4–1 below describes the OC4J runtime options

Table 4–1 OC4J Startup Options

Command-Line Argument	Description
`-quiet`	Suppresses standard output to the console.
`-config <path>`	Specify the path to the `server.xml` descriptor file. The default location is the `ORACLE_HOME/j2ee/home/config` directory.
`-validateXML`	Validates XML configuration files at the time they are read.

Table 4–1 (Cont.) OC4J Startup Options

Command-Line Argument	Description
`-rewriteXML`	Prompts you to errors found in XML configuration files and re-rewrites the invalid XML as accurately as possible.
`-out [file]`	Specifies a file to route the standard output to. The file contains messages that are printed to `System.out`, as well as the messages sent to output through the servlet logging interface. If not specified, all output is written to standard out. See "Managing stdout/stderr Log Files" on page 4-7 for additional system properties that can be set to manage `stdout` files
`-err [file]`	Specifies a file to route standard error output to. The file contains messages that are printed to `System.err`. If not specified, all errors are written to standard error. See "Managing stdout/stderr Log Files" on page 4-7 for additional system properties that can be set to manage `stderr` files.
`-verbosity <int>`	Define an integer between 1 and 10 to set the verbosity level of the message output. Example: `-verbosity 10`. A value of `10` will produce the most verbose output.
`-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.
`-userThreads`	Enables context lookup support from user-created threads.
`-version`	Returns the installed version of OC4J installation and exits.
`-? |-help`	Prints the help text for these options to the console.

Setting System Properties at Startup

You can set a number of OC4J-specific system properties on the JVM at OC4J startup.

- Setting System Properties in a Standalone OC4J Configuration
- Setting System Properties in a Managed OC4J Configuration
- Overview of General System Properties
- Overview of Debug Properties
- Managing stdout/stderr Log Files

Setting System Properties in a Standalone OC4J Configuration

You can set system properties on the JVM through the OC4J command-line at startup. If OC4J is running, you must restart the instance for new property settings to take effect.

The syntax is as follows:

```
java [props] -jar oc4j.jar [args]
```

Note that all system properties (`[props]`) are specified before `oc4j.jar` in the syntax. All system properties must be prefaced on the command line with a `-D`. For example:

```
java -Dejb3=true -DGenerateIIOP=true -Dhttp.session.debug=true -jar oc4j.jar
```

Setting System Properties in a Managed OC4J Configuration

When OC4J is installed as a component of Oracle Application Server, OC4J system properties must be manually added to the `opmn.xml` configuration file. Options will be passed to managed OC4J instances at startup.

Set Java system properties in the `<data>` element where the `id` attribute is `"java-options"`. This `<data>` element is enclosed within the `<category id="start-parameters">` subelement of the `<ias-component id="OC4J">` element in the XML structure. Preface all system properties with a `-D`. For example:

```
<ias-component id="OC4J">
  <process-type id="home" module-id="OC4J" status="enabled">
    <module-data>
      <category id="start-parameters">
        <data id="java-options" value="-Dejb3=true -Dhttp.session.debug=true"/>
      </category>
      ...
    </module-data>
  </process-type>
</ias-component>
```

Overview of General System Properties

The following table describes the general system properties that can be set for OC4J.

Table 4–2 -D General System Properties for OC4J

Property	Description
`ejb3=<true|false>`	Set to `true` to enable EJB 3.0 support. Note that the JDK release 5.0 is required to use EJB 3.0.
`java.ext.dirs`	Sets the external directories to be searched for classes when compiling.
`java.io.tmpdir=<new_tmpdir>`	Sets the temporary directory for the deployment wizard. The default is `/tmp/var`. 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. However, if the wizard is interrupted, it may not have the time or opportunity to clean up the temp directory. In this case, you must clean up any additional deployment files from this directory yourself. If not, the 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.
`GenerateIIOP= <true|false>`	Enables IIOP stub generation. The default is `false`.
`KeepIIOPCode= <true|false>`	Set whether the generated IIOP stub/tie code is kept. The default is `false`.
`oracle.arraylist.deepCopy= <true|false>`	If `true`, then while cloning an array list, a deep copy is performed. If `false`, a shallow copy is performed for the array list. The default is `true`.

Table 4–2 (Cont.) -D General System Properties for OC4J

Property	Description
`dedicated.rmicontext=<true|false>`	This property replaces the deprecated `dedicated.connection` setting. The default is `false`. 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. Note that you can also set this in the JNDI properties.
`oracle.mdb.fastUndeploy=<int>`	Sets the interval at which OC4J polls the underlying database to check if an MDB session is shut down, in seconds. This property enables you to shut down OC4J cleanly when you are running MDBs in a Windows environment or when the back-end 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 shutdown OC4J while the MDB is in a wait state in a Windows environment, the OC4J instance cannot be stopped and the applications are not undeployed since the MDB is blocked. Setting this property enables OC4J to poll the database to see if the session is shut down when the MDB is not processing incoming messages and in a wait state. If you do not set this property and you try to shutdown OC4J using CTRL-C, then the OC4J process will hang for at least 2.5 hours. Note that this polling process can be expensive for performance, and should not be set to start too frequently.
`oracle.dms.sensors=<none|normal|heavy|all>`	You can set the value for Oracle built-in performance metrics to the following: ■ `none`: Disables metrics ■ `normal`: Medium number of metrics (default) ■ `heavy`: High number of metrics ■ `all`: Every possible metric 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`.
`associateUsingThirdTable=<true|false>`	For container-managed relationships in entity beans, you can designate if a third database table that is used to manage the relationship. Set to `false` if you do not want a third association table. The default is `false`.

Table 4–2 (Cont.) -D General System Properties for OC4J

Property	Description
`DefineColumnType=<true|false>`	Set this to `true` if you are using a pre-9.2.0 Oracle JDBC driver. 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. The default is `false`. 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.
`oc4j.formauth.redirect=<true|false>`	This property is applicable when form-based authentication is used by a Web application. If set to `true`, OC4J will perform a client side redirect back to the request URL after a user enters valid credentials when accessing a resource. If the user does not have valid credentials, the Web browser will be redirected to the form authentication error page defined for the Web application. If set to `false`, the `/j-security-check` URL will be displayed in the browser after the user enters valid credentials. The default is `false`.
`http.webdir.enable=<true|false>`	This property enables or disables servlet class name invocation for all servlets within the OC4J instance. If set to `false`, the default value, servlets cannot be invoked by class name. If set to `true`, any servlet running in the OC4J instance can be invoked by class name by default. To disable this functionality on a per-Web-application basis, set this property to `true`, then set `<orion-web-app servlet-webdir="" ... />` in the `orion-web.xml` descriptor for each Web application that should not allow servlet class name invocation. Note that the value set for `servlet-webdir` in `orion-web.xml` overrides the default value set for this attribute in `ORACLE_HOME/j2ee/home/config/global-web-application`, which is `servlet-webdir="/servlet"`.

Overview of Debug Properties

Use the following properties to better debug applications running within OC4J. Debug messages are printed to the console. All properties take a Boolean value.

Table 4–3 OC4J Debug Properties

Debug Property	Description
`http.session.debug=<true|false>`	Provides information about HTTP session events to the console.
`http.request.debug=<true|false>`	Provides information about each HTTP request to the console.
`http.cluster.debug=<true|false>`	Provides information about HTTP clustering events to the console.
`http.error.debug=<true|false>`	Prints all HTTP errors to the console.
`http.method.trace.allow=<true|false>`	Enables the `trace` HTTP method.
`datasource.verbose=<true|false>`	Provides verbose information on creation of data source and connections using data sources and connections released to the pool.
`jdbc.debug=<true|false>`	Provides verbose information when JDBC calls are made.
`ejb.cluster.debug=<true|false>`	Enables EJB clustering debug messages.
`rmi.debug=<true|false>`	Prints RMI debug information to the console.
`rmi.verbose=<true|false>`	Provides verbose information on RMI calls.
`jca.connection.debug=<true|false>`	Provides extra diagnostic information for J2CA connections.
`ws.debug=<true|false>`	Enables Web Services debugging.

Managing stdout/stderr Log Files

The following properties are used to manage standard `stderr` and `stdout` log files.

The type of log file(s) the properties pertain to are specified with the `-out` and/or `-err` command line options. You can also set a location to output these log files to on these options.

For example, set the following to rotate `stdout/stderr` files when the file size reaches 2.5MB. Log files will be output to the `D:\logs` directory.

```
java -Dstdstream.filesize=2.5 -jar oc4j.jar -out d:\logs\oc4j.out -err
d:\logs\oc4j.err
```

This example will rotate `stdout` logs at 2:30 p.m. every day and limit the archive to a maximum of 10 files:

```
java -Dstdstream.rotatetime=14:30 -Dstdstream.filenumber=10 -jar oc4j.jar -out
d:\logs\oc4j.out
```

Table 4–4 stdout/stderr Archive Management Properties

Debug Property	Description
`stdstream.filesize=<max_file_size>`	The maximum size any file in the archive will be allowed to grow to, in megabytes. Files are rotated when this maximum is reached.
`stdstream.filenumber=<max_files>`	The maximum number of files to keep as archives. The oldest file will be automatically deleted when the limit is exceeded.
`stdstream.rotatetime=<HH:mm>`	The time at which the log file will be rotated each day.

5

Starting and Stopping OC4J

This chapter provides instructions for installing OC4J, as well as for starting, stopping and restarting an OC4J instance. It includes the following sections:

Starting OC4J in a Standalone Environment

You can start an OC4J server instance in a standalone environment using the default configuration with either the `oc4j.jar` command line or the `oc4j` executable scripts.

Starting OC4J with oc4j.jar

To start OC4J by invoking `oc4j.jar`, issue the following command from the `ORACLE_HOME/j2ee/home/` directory:

```
java -jar oc4j.jar [args]
```

Invoking `oc4j.jar` as shown above starts OC4J using the default `server.xml` configuration files, which you can find in the `j2ee/home/config` directory. To start OC4J using a non-default version of the `server.xml` file, issue the following command. Note that you must supply the path to the modified configuration file.

```
java -jar oc4j.jar -config /yourpath/server.xml [args]
```

Note that you can optionally pass in arguments at startup to set runtime options in OC4J. For an overview of valid arguments, see "Setting OC4J Runtime Options at Startup" on page 4-2.You can also view the console help by issuing the following command from the `ORACLE_HOME/j2ee/home` directory:

```
java -jar oc4j.jar -help
```

You can also set system properties on the JVM through the `oc4j.jar` command-line at OC4J startup. For details on setting system properties, see "Setting System Properties at Startup" on page 4-3.

Starting OC4J with oc4j/oc4j.cmd

To start OC4J using the `oc4j` scripts, issue the following command from the `ORACLE_HOME/bin` directory.

On Unix/Linux:

```
oc4j -start
```

On Windows:

```
oc4j.cmd -start
```

Starting OC4J in an Oracle Application Server Environment

In a managed configuration, all Oracle Application Server components - including OC4J and OHS - should be started using `opmnctl`, the OPMN command line tool. This tool is installed in the `ORACLE_HOME/opmn/bin` directory.

Use the following command to start all OPMN managed processes, including OC4J, on a local Oracle Application Server instance:

```
opmnctl startproc
```

Use the following command to start a specific managed processes - in this case OC4J - on a local Oracle Application Server instance:

```
opmnctl startproc ias-component=OC4J
```

Stopping OC4J in a Standalone Environment

You can stop OC4J using the `admin.jar` command line utility as well as the `oc4j.cmd` or `oc4j` executable scripts. You can also stop OC4J with your operating system commands, such as `Control-C` on Windows or `kill` on Unix/Linux machines.

Stopping OC4J with admin.jar

To stop OC4J using `admin.jar`, issue the following command:

```
java -jar admin.jar ormi://<oc4jHost>:<oc4jOrmiPort> <adminId>
   <adminPassword> -shutdown [ordinary|force] [reason]
```

Note the following options that can be specified:

- `[ordinary | force]`
  The type of shutdown. The default is `ordinary`, which allows each thread to terminate normally. The `force` option terminates all threads immediately.
- `[reason]`
  You can optionally specify a reason for the shutdown as a string that is written to the `ORACLE_HOME/j2ee/home/log/server.log` file. Spaces are not allowed in the string.

The following example forces a shutdown of the OC4J server using `admin.jar`, which terminates all threads immediately. Note the string entered as the `reason` for the shutdown, which is written to the `ORACLE_HOME/j2ee/home/config/server.log` file.

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -shutdown force
  need_to_reboot_host_machine
```

Stopping OC4J with oc4j/oc4j.cmd

To stop OC4J using the `oc4j` scripts, issue the following command from the `ORACLE_HOME/bin` directory. Note that you must supply the ORMI port used by OC4J, which is `23791` by default, as well as the password for the `oc4jadmin` account.

Unix/Linux:

```
oc4j -shutdown -port <oc4jOrmiPort> -password <adminPassword>
```

Windows:

```
oc4j.cmd -shutdown -port <oc4jOrmiPort> -password <adminPassword>
```

For example:

```
oc4j.cmd -shutdown -port 23791 -password adminpwd
```

Stopping OC4J in an Oracle Application Server Environment

In a managed configuration, all Oracle Application Server components - including OC4J and OHS - should be stopped using `opmnctl`, the OPMN command line tool. This tool is installed in the `ORACLE_HOME/opmn/bin` directory.

Use the following command to stop all OPMN managed processes, including OC4J, on a local Oracle Application Server instance:

```
opmnctl stopproc
```

Use the following command to stop a specific managed component - in this case OC4J - on a local Oracle Application Server instance:

```
opmnctl starpproc ias-component=OC4J
```

Restarting an OC4J Instance in a Standalone Environment

You can restart OC4J using the `admin.jar` command line utility. Restart OC4J by executing the following command:

```
java -jar admin.jar ormi://<oc4jHost>:<oc4jOrmiPort> <adminId>
   <adminPassword> -restart [reason]
```

Note that you can optionally enter a string as the value for `[reason]`. The string is written to the `ORACLE_HOME/j2ee/home/config/server.log` file.

6

Using the admin.jar Command Line Utility

OC4J provides a command-line utility called `admin.jar` that can be used to perform operations on an active OC4J instance in a standalone environment. Among other things, you can use this utility to restart and stop OC4J, deploy applications and gather information on current resource usage.

The `admin.jar` utility can be used to perform the following tasks:

- Using the admin.jar Command Line Utility
- Managing OC4J in a Standalone Environment
- Managing Applications
- Managing Web Sites
- Managing Data Sources
- Deploying/Undeploying Connectors

The `admin.jar` is installed by default in `ORACLE_HOME/j2ee/home`. Note that OC4J must be started before this utility can be used. Also note that the utility cannot be used to start OC4J, although it can be used to stop then re-start an instance.

> **Note:** The `admin.jar` utility can only be used to manage a single OC4J instance in a standalone environment. You must use OPMN or Application Server Control Console to manage OC4J instances in a managed environment.

Using the admin.jar Command Line Utility

The `admin.jar` command-line utility enables you to administer any OC4J instance from a client console.

The utility is installed by default in `ORACLE_HOME/j2ee/home`. Note that the OC4J server must be started before this utility can be used.

- Understanding the admin.jar Syntax
- Printing Help to the Console

Understanding the admin.jar Syntax

The `admin.jar` utility uses the following syntax. The variables are described in the table below.

```
java -jar admin.jar ormi://<oc4jHost>:<oc4jOrmiPort> <adminId>
   <adminPassword> <options>
```

For example, the following command will force a shutdown of the OC4J server. The value supplied for <oc4jOrmiPort> is the default, 23791. The username supplied for <adminId> is the username for the default administrator account, oc4jadmin.

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -shutdown
```

Note that some of these commands include an -application switch which takes the name of the application to effect. This value can be one of the following:

- The global application name, installed originally as default, specified in the name attribute of the <global-application> element in the server.xml file.
- A specific application name defined within an <application> element in the server.xml file.

Table 6–1 Setting the Host and Login Information

Variable	Description
<oc4jHost>:<oc4jOrmiPort>	The host name and port of the OC4J server on which you are invoking admin.jar. The admin.jar tool uses the OC4J Remote Method Invocation (ORMI) protocol to communicate with the OC4J server. Therefore, the host name and port identified by these variables are defined in the rmi.xml file for the OC4J server to which you are directing the request. The OC4J default port for the ORMI protocol is 23791. This value can be omitted if not changed. Configure both the host name and port number - if not using the default - in the rmi.xml file in the <rmi-server> element, as follows: <rmi-server port="<oc4jOrmiPort>" host="<oc4jHost>" />
<adminId> <adminPassword>	:The OC4J administration username and password. The username for the default administrator account is oc4jadmin.

Printing Help to the Console

To print the inline help text for the admin.jar commands to the console, simply type -help after <oc4jHost>:<oc4jOrmiPort> <adminId> <adminPassword>. For example:

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -help
```

Managing OC4J in a Standalone Environment

This section outlines the functionality provided by admin.jar for managing an OC4J server. It includes the following sections:

- Stopping and Restarting OC4J in a Standalone Environment
- Forcing OC4J to Check For Modified Files

Stopping and Restarting OC4J in a Standalone Environment

You can use admin.jar to shut down a standalone instance of the OC4J server, then restart it.

The following command forces a shutdown of the OC4J server, which terminates all threads immediately. The string entered as the reason for the shutdown is written to the server log file, ORACLE_HOME/j2ee/home/log/server.log.

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -shutdown force
  need_to_reboot_host_machine
```

Table 6–2 Options for OC4J Server Shutdown/Restart

Option	Description

`-shutdown`	Shuts down the OC4J server.
	`[ordinary|force]:`
	The type of shutdown. The default is `ordinary`, which allows each thread to terminate normally. The `force` option terminates all threads immediately.
	`[reason]:`
	You can optionally specify a reason for the shutdown as a string that is written to the `ORACLE_HOME/j2ee/home/log/server.log` file. Spaces are not allowed in the string.

`-restart`	Restarts the OC4J server. The container must have been started with `oc4j.jar`. `[reason]:` You can optionally specify a reason for the restart as a string that is written to the `ORACLE_HOME/j2ee/home/log/server.log` file. Spaces are not allowed in the string.
`-version`	Prints the installed version of OC4J to the console, then exits.

Forcing OC4J to Check For Modified Files

If you have OC4J polling disabled - as it should be in a production environment - then OC4J does not automatically reload modified files. However, you can force OC4J to check the server directories for modified files and reload any that have changed using the `-updateConfig` option.

Note that the value of the `checkForUpdates` flag must be set to either `all` or `adminClientOnly` to use this feature. See Chapter 11, "Automatic Deployment in OC4J" in the *Oracle Containers for J2EE Deployment Guide* for details on the `checkForUpdates` flag.

Table 6–3 Option for Checking for Updated Files

Option	Description
`-updateConfig`	Forces OC4J to check files for changes and reload any files that have been modified.

Managing Applications

This section outlines the functionality provided by `admin.jar` for deploying and managing applications. It includes the following sections:

- Deploying/Undeploying Applications
- Starting/Stopping/Restarting an Application
- Updating an EJB Module Within an Application

Deploying/Undeploying Applications

Deploying an application is a two step process: You must first deploy the archive into OC4J, then bind the Web module to the Web site that will be used to access the application.

The `-deploy` command is first used to deploy the application:

```
java -jar admin.jar ormi://<oc4jHost>:<oc4jOrmiPort> <adminId>
   <adminPassword> -deploy -file <path/filename>
    -deploymentName <appName> -targetPath <deploy_dir>
```

Once the archive is deployed, the `-bindWebApp` command is used to bind a Web application to the Web site it will be accessed through:

```
java -jar admin.jar ormi://<oc4jHost>:<oc4jOrmiPort> <adminId>
   <adminPassword> -bindWebApp <appName> <webAppName>
   <webSiteName> <contextRoot>
```

For example, the following command deploys the `utility` application into OC4J:

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -deploy -file
utility.ear -deploymentName utility
```

Next, the following example binds the `utility` application and its `utility-web` Web module to the default OC4J Web site:

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -bindwebapp utility
utility-web http-web-site /utility
```

Table 6–4 Options for Application Deployment

Option	Description
`-deploy`	Deploys an application. Supply relevant information using the following subswitches: `-file <filename>`: Required. The path and filename of the EAR file to deploy. `-deploymentName <appName>`: Required. The user-defined application deployment name. This same name is used to identify the application within OC4J. It is also provided when you want to undeploy the application. `-targetPath <path>`: Optional. The path to the OC4J server node to copy the archive to. If `-targetPath` is not specified, the EAR file is copied to the `ORACLE_HOME/j2ee/home/applications` directory. When you redeploythe updated EAR file, the existing EAR will be overwritten. `-parent <appName>`: Optional. The parent application of this application. When deployed, any method within the child application can invoke any method within the parent application. In no parent is specified, the `default` application serves as the default parent. `-deploymentDirectory <path>`: Optional. If not specified, the application is deployed into the `application-deployments/` directory. To change the location, specify a path with this option. If you supply the string"[NONE]", the deployment configurations are always read from the EAR file each time the application is deployed. `-iiopClientJar <path/filename>`: Optional. Include to generate IIOP stubs for the home, remote and local interfaces packaged within each EJB JAR included in the EAR. Specify the path and filename of the file to output the generated stubs to. By default, copies of the stubs will be output to an archive named `_iiopClient.jar` in a new subdirectory with the same name as the deployed EJB JAR in `ORACLE_HOME/j2ee/home/<app_name>/application-deployments/`.
`-bindWebApp`	Binds a Web application to the specified Web site and root. ■ `<appName>`: The application name, which is the same name set as the value for `-deploymentName` on the `-deploy` option. ■ `<webAppName>`: The name of the Web module. This should be the name of the WAR file contained within the EAR file, without the `.WAR` extension. ■ `<webSiteName>`: The name of the `<name>_web-site.xml` file that denotes the Web site that this Web application should be bound to. ■ `<contextRoot>`: The root context for the Web module. This will be appended to the URL used to access the application through a Web browser; for example `http://localhost:8888/utility`. This option creates an entry in the `<name>-web-site.xml` configuration file that was denoted in the *web_site_name* variable.

Table 6–4 (Cont.) Options for Application Deployment

Option	Description
`-undeploy <appName>`	Removes the deployed J2EE application from the OC4J instance. The `<appName>` is the name of the application within OC4J, as defined in an `<application>` element within `ORACLE_HOME/j2ee/home/config/server.xml`. Undeploying an application results in the following: ■ The application is removed from the OC4J runtime and the `server.xml` file. ■ Bindings for all the application's Web modules are removed from all the Web sites to which the Web modules were bound. ■ Application files are removed from both the `applications` and `application-deployments` directories. `-keepFiles`: This optional subswitch is deprecated in OC4J 10*g* (10.1.3).

Starting/Stopping/Restarting an Application

You can use `admin.jar` to start, stop and restart an application that has been stopped in the OC4J instance.

The following example restarts a specific application running on OC4J. If a file within the application has been modified, the application or module will be automatically redeployed.

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -application
   myapplication -restart
```

Table 6–5 Options for Application Restart

Option	Description
`-application <appName> -start`	Starts the specified application and any child applications.
`-application <appName> -stop`	Stops the specified application and any child applications.
`-application <appName> -restart`	Restarts the specified application. If OC4J polling is enabled. and a file within the application has been modified, the application will be redeployed.

Updating an EJB Module Within an Application

The `admin.jar` utility includes an `-updateEJBModule` option that allows incremental or partial redeployment of EJB modules within an application running in an OC4J instance. This option is intended to be used by an application developer to redeploy the JAR file directly from his/her development environment.

The syntax is as follows:

```
java -jar admin.jar ormi://<oc4jHost>:<oc4jOrmiPort> <adminId>
    <adminPassword> -application <appName> -updateEJBModule <relativePath>
    [-file <path/ejbJarName>]
```

For example, the following commands can be used to update the `customerEjb.jar` module of the `petstore` application. Assume the following directory structure on the developer's machine:

```
/work
  /src    - application source code
  /build  - compiled class files
  /dist   - assembled EAR and JAR files
```

If the updated EJB JAR is in the `/dist` directory, in a location matching the relative path defined in the application's `application.xml` J2EE standard deployment descritptor, the following command could be issued from the `/dist` directory:

```
java -jar $OC4J_HOME/admin.jar ormi://myoc4jserver:23791 oc4jadmin welcome
    -application petstore -updateEJBModule customerEjb.jar
```

If the updated file is located within the `/build` directory, the following command specifying the JAR location in the option -file subswitch can be issued from the `/dist` directory:

```
java -jar admin.jar ormi://myoc4jserver:23791 oc4jadmin welcome
    -application petstore -updateEJBModule customerEjb.jar
    -file build/customerEjb.jar
```

Table 6–6 Options for Updating an EJB Module

Option	Description
`-application <appName> -updateEJBModule`	Updates the specified EJB module with new EJBs. ▪ `<relativePath>`: The relative path to the EJB JAR containing the updated beans as defined in the application's `application.xml` J2EE deployment descriptor. ▪ `-file <path>`: The path and file name of the updated EJB JAR if the file's location does not match the relative path specified in the `application.xml` deployment descriptor.

Managing Web Sites

The `-site` option enables you to configure new Web sites, including secure sites, for use by applications deployed into OC4J. You can also retrieve a list of existing sites; test existing sites; as and update or remove existing Web sites.

The syntax of the `-add` option, which configures a new site, is as follows:

```
java -jar admin.jar ormi://<oc4jHost>:<oc4jOrmiPort> <adminId>
   <adminPassword> -site -add [-host <hostName>] -port <port> -display-name <name>
[-virtual-hosts <hostNames>] [-secure [true|false]] [-factory <class>] [-keystore
<path>] [-storepass <password>] [-provider <class>] [-needs-client-auth
[true|false]]
```

For example, the following command structure configures a new Web site on port `8899` with two virtual hosts:

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -site -add -host
www1.acme.com -port 8899 -display-name MyServer -virtual-hosts
MyServer.com,MyServer2.com
```

The next example configures a secure Web site to receive HTTPS requests on port `4443`. See "Configuring a Secure Web Site in OC4J" on page 11-6 for instructions on creating secure Web sites.

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -site -add
-host www1.acme.com -port 8899 -display-name MySecureSite -secure
```

```
true -factory com.evermind.ssl.JSSESSLServerSocketFactorykeystore
-keystore ../../server.keystore -storepass welcome
-provider com.sun.net.ssl.internal.ssl.Provider -needs-client-auth true
```

Table 6–7 Options for Web Site Administration

-site options	Description
`-site -add`	Installs a new Web site. Supply information with the following subswitches: `-host <hostName>`: The hostName or IP address hosting the site. Optional. `-port <port>`: The Web site port. Required. `-display-name <name>`: A user-friendly "display-able" name of the Web site. `-virtual-hosts <hostNames>`: One or more virtual Web sites sharing the same IP address. The value is a comma-delimited list of host names tied to this Web site. Optional. `-secure [true|false]`: The value is `true` if the Web site is secure. The default is `false`. `-factory <className>`: The name of the class extending `SSLServerSocketFactory` if you are not using the Java Secure Socket Extension (JSSE). Optional. `-keystore <path>`: The relative or absolute path to a keystore. Optional. `-storepass <password>`: The keystore password. Optional. `-provider <provider>`: The security provider to use if not using JSSE. If not specified, the Sun Microsystems implementation - `com.sun.net.ssl.internal.ssl.Provider` - is used by default. The JSSE defines a provider interface that other security providers can implement. Optional. `-needs-client-auth [true|false]`: If set to `true`, a client that wants to access the Web site must identify itself with a digital certificate. The default is `false`.
`-site -remove`	Removes an existing Web site. Supply the host and port of this Web site with the following subswitches: `-host <hostName>`: The Web site host to be removed. `-port <port>`: The Web site port to be removed.
`-site -test`	Tests an existing Web site. Supply the host and port of the Web site to be tested with the following subswitches: `-host <hostName>`: The Web site host to be tested. `-port <port>`: The Web site port to be tested.
`-site -list`	Lists all existing Web sites configured within the OC4J instance.

Table 6–7 (Cont.) Options for Web Site Administration

-site options	Description
`-site -update`	Updates an existing Web site. Supply information with the following subswitches: `-oldHost <hostName>`: The hostName or IP address of the current Web site host. `-oldPort <port>`: The current port. `-newHost <hostName>`: The hostName or IP address of the new Web site host. `-newPort <port>`: The new port. `-display-name <name>`: A user-friendly "display-able" name of the Web site. `-virtual-hosts <hostNames>`: One or more virtual Web sites sharing the same IP address. The value is a comma-delimited list of host names tied to this Web site. Optional. `-secure [true|false]`: The value is `true` if the Web site is secure. The default is `false`. `-factory <className>`: The name of the class extending `SSLServerSocketFactory` if you are not using the Java Secure Socket Extension (JSSE). Optional. `-keystore <path>`: The relative or absolute path to a keystore. Optional. `-storepass <password>`: The keystore password. Optional. `-provider <provider>`: The security provider to use if not using JSSE. If not specified, the Sun Microsystems implementation - `com.sun.net.ssl.internal.ssl.Provider` - is used by default. The JSSE defines a provider interface that other security providers can implement. Optional. `-needs-client-auth [true|false]`: If set to `true`, a client that wants to access the Web site must identify itself with a digital certificate. The default is `false`.

Managing Data Sources

Use `admin.jar` to create, remove, list or test data sources for a specific application. You can also convert a pre-Release 3 (10.1.3) `data-sources.xml` file to the new file format.

Creating an Application-Specific Data Source

The syntax of the `-installDataSource` option, which configures a new application-specific data source, is as follows:

```
java -jar admin.jar ormi://<oc4jHost>:<oc4jOrmiPort> <adminId>
<adminPassword> -application <appName> -installDataSource -jar <path>
-url <url> -location <jndiName> -pooledLocation <jndiName>
-username <name> -password <password> -className <className>
```

For example:

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome -application myapp
-installDataSource -jar C:/jdbc/lib/ojdbc14dms.jar
```

```
-url jdbc:oracle:thin:@dev2:1521:main -location jdbc/OracleUddi
-username dbuser -password dbpw -className oracle.jdbc.pool.OracleDataSource
```

Table 6–8 Options For Data Source Management

Option	Description
`-application <appName> -installDataSource`	Installs a new data source for the specified application. Supply data source information within the following subswitches: `-jar <path>`: The path to the JAR file containing the JDBC driver that is to be added to the OC4J server. `-url <url>`: The JDBC database URL. `-location <jndiName>`: The JNDI namespace location for the raw source. For example, `"jdbc/DefaultPooledDS"`. Required. `-pooledLocation <jndiName>`: The JNDI namespace location for the pooled data source. For example, `"jdbc/DefaultPooledDS"`. `-xaLocation <jndiName>`: The namespace location for the XA source. For example, `"jdbc/xa/DefaultXADS"`. Required if `-ejbLocation` is specified. `-ejbLocation <jndiName>`: The namespace location for the container-managed transactional data source. This is the only data source that can perform global JTA transactions. For example, `"jdbc/DefaultDS"`. `-username <name>`: The username to log in with. `-password <password>`: The password to log in with. `-connectionDriver <className>`: The JDBC database driver class. `-className <className>`: The data source class name, such as `com.evermind.sql.DriverManagerDataSource`. Required. `-sourceLocation <jndiName>`: The underlying data source of this specialized data source. `-xaSourceLocation <jndiName>`: The underlying XA data source of this specialized data source.

Listing/Testing/Removing Existing Data Sources

You can use `admin.jar` to list, test or even delete data sources tied to a specific application.

Table 6–9 Options For Application And Data Source Management

Option	Description
`-application <appName> -listDataSource`	Retrieves the statically configured information about each installed data source object.

Table 6–9 (Cont.) Options For Application And Data Source Management

Option	Description
`-application <appName> -testDataSource`	Tests an existing data source. Supply information with the following subswitches: `-location <jndiName>`: The namespace location for the `DataSource`. For example, `jdbc/DefaultDS`. Required. `-username <name>`: The username you use to login along with a password. Optional. `-password <password>`: The password to log in with. Optional.
`-application <appName> -removeDataSource`	Removes an existing `DataSource`. Supply information with the following subswitches: `-location <jndiName>`: The namespace location for the `DataSource`. For example, `jdbc/DefaultDS`. Required.

Converting Existing Data Sources to the New Configuration

The `-convertDataSourceConfiguration` option converts a pre-Release 3 (10.1.3) `data-sources.xml` file to the new file format.

The syntax is as follows:

```
java -jar admin.jar ormi://<oc4jHost>:<oc4jOrmiPort> <adminId>
   <adminPassword> -convertDataSourceConfiguration <legacyFileName>
<convertedFileName>
```

For example, the following command converts an existing configuration and writes it to a new file:

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin welcome
-convertDataSourceConfiguration C:\oc4j\j2ee\home\config\data-sources.xml
C:\new\data-sources.xml
```

Ideally, you should rename the "old" `data-sources.xml` after the conversion, rather than delete it, as it contains information that may be needed for reference. After the "new" file has been generated, copy it into the directory containing the legacy file.

Note that the generated `data-sources.xml` file may include JNDI entries that are not needed, but are transferred from the legacy file. Edit the file to remove these entries.

Table 6–10 Options for Data Source File Conversion

Arguments	Description
`<legacyFileName>`	The fully qualified path to the "old" `data-sources.xml` file you want to convert.
`<convertedFileName>`	The fully qualified path to the "new" `data-sources.xml` file containing the converted configuration.

Deploying/Undeploying Connectors

You can use the following commands to deploy or undeploy a Java Connector Architecture-compliant resource adapter packaged in a RAR archive file.

Table 6–11 Options for Application Deployment

Option	Description
`-deployconnector`	Deploys a connector. Supply application information in the following subswitches: `-file <path>`: Required. The path and filename of the RAR file to deploy. `-name <name>`: The name of the resource adapter. `-nativeLibPath <path>`: The path to the directory containing native libraries (such as DLLs) within the RAR file. `-grantAllPermissions`: Specify to grant all runtime permissions requested by the resource adapter, if required.
`-undeployconnector`	Undeploys the specified connector. `name <name>`: The name of the connector to undeploy.

7

Clustering in OC4J

This chapter discusses the clustering framework provided in OC4J 10*g* (10.1.3). It includes the following topics:

- What is Clustering in OC4J?
- Configuring Clustering

What is Clustering in OC4J?

OC4J provides a flexible framework for creating a clustered environment for development and production purposes. In this context, a *cluster* is defined as two or more OC4J server nodes hosting the same set of applications. The OC4J clustering framework supports:

- Replication of objects and values contained in an HTTP session or a stateful session Enterprise JavaBean instance.
- In-memory replication using multicast or peer-to-peer communication, or persistence of state data to a database.
- Load balancing of incoming requests across OC4J instances.
- Transparent failover across applications within the cluster. For HTTP requests, Oracle Application Server Web Cache is able to detect server failures and re-route incoming HTTP requests; EJB clients are re-routed directly by the clustering framework.

Clustering can be configured within an OC4J instance at either the global or the application level. All configuration and management is performed through manual edits to application configuration files - `application.xml` at the global level, or `orion-application.xml` for individual applications.

> **Note:** At the application level, clustering can also be configured at the time the application is deployed using Oracle Enterprise Manager 10*g* Application Server Control Console, through either the deployment tasks or the deployment plan editor. See the *Oracle Application Server Forms Services Deployment Guide* for details.

A new `<cluster>` element, which contains a number of new sub-elements, has been added to the XML schema definition for these files to provide a single mechanism for clustering management. See "Overview of the <cluster> Element" on page 7-9 for descriptions of this element and its sub-elements.

How Does Clustering Differ From Previous OC4J Releases?

The following are no longer included in the clustering framework in OC4J 10*g* (10.1.3).

"Islands" No Longer Used

The notion of *islands*, part of the clustering framework in previous OC4J releases, is no longer supported in OC4J.

In previous releases, an island was essentially a group of OC4J instances within a cluster across which HTTP session data was replicated. Although islands reduced overhead by not replicating data across the entire cluster, they increased configuration and management overhead. In addition, islands were only applicable to Web applications; EJB applications could not utilize the island configuration.

In OC4J 10*g* (10.1.3), you can effectively limit the number of instances to replicate data to using the `write-quota` attribute of the `<cluster>` element. This makes it possible to control the scope of state replication while eliminating the need to configure and manage islands.

See "Overview of the <cluster> Element" on page 7-9 for details on the `write-quota` attribute.

loadbalancer.jar No Longer Used

The `loadbalancer.jar`, which provided load balancing functionality in previous OC4J releases, was deprecated in the previous release of OC4J hand has been removed from the current release.

Deprecated Clustering-Specific XML Elements

The following XML elements are deprecated in OC4J 10*g* (10.1.3) and should no longer be used to configure clustering. The new `<cluster>` element is now used for all cluster management.

- The `<cluster-config>` element in `server.xml`, the OC4J configuration file
- The `cluster-island` attribute of the `<web-site>` element in a `*-web-site.xml` Web site configuration file

Configuring Clustering

Clustering is enabled by adding the `<cluster>` element to the `orion-application.xml`file of each application to be clustered within an OC4J instance. See "Overview of the <cluster> Element" on page 7-9 for descriptions of this element and its sub-elements.

This section includes the following topics:

Clustering can be configured globally for all applications running within an OC4J instance, as well as on a per-application basis.

- Global clustering is configured through the `ORACLE_HOME/j2ee/home/config/application.xml`, the configuration file for the global `default` application. All other applications deployed into the OC4J instance inherit default properties from this application, including clustering configuration.
- Application-level clustering is defined in the application-specific `ORACLE_HOME/j2ee/home/application-deployments/<app_name>/orion-application.xml`. Settings in this file override the global configuration, as well as the configuration inherited from a parent application.

Managing a clustered environment is an entirely manual process, as no distributed management tools are provided. This means that any changes made to a particular application's `orion-application.xml` file on one OC4J server must be manually replicated to the corresponding XML file on all servers.

At the application level, clustering can be configured at the time the application is deployed into an OC4J instance using the deployment plan editor, which sets values in the application's `orion-application.xml` file. See the *Oracle Containers for J2EE Deployment Guide* details on using the deployment plan editor. Note that the deployment plan editor does not support configuration of the global `default` application, and configuration at the global level is completely manual.

Important: The `<distributable/>` tag must be added to the `web.xml` file for all Web modules that are part of an application configured to use clustering. Post-deployment, this J2EE standard Web module descriptor is found in the `ORACLE_HOME/j2ee/home/applications/<app_name>/<web_module>/WEB-INF` directory within OC4J.

Setting Replication Policies

A replication policy defines when replication of `HttpSession` or stateful session bean state occurs, and whether all attributes or only changed values are replicated. Replication can be an expensive process; frequent replication can affect server performance. On the other hand, replicating data too infrequently can result in lost data in the event of server failure.

The default replication policy applied to all Web modules and EJB components within an application is specified in the `<replication-policy>` element within the application's `orion-application.xml` configuration file. The syntax of this element is as follows:

```
<replication-policy trigger="onSetAttribute|onRequestEnd|onShutdown"
 scope="modifiedAttributes|allAttributes" />
```

- The `trigger` attribute specifies when replication occurs. By default, the `onRequestEnd` policy is applied, as it provides frequent replication of data while ensuring that data is not lost if the JVM terminates unexpectedly.

  See Table 7–1 for an overview of `trigger` attribute values.
- The `scope` attribute defines what data is replicated: Either all attributes, or changed attributes only. By default, only modified HTTP session attributes are replicated; for stateful session beans, all attributes are replicated.

See Table 7–2 for an overview of `scope` attribute values.

Table 7–1 <replication-policy> trigger Attribute Values

trigger Value	HttpSession	Stateful Session Bean
`onSetAttribute`	Replicate each change made to an HTTP session attribute at the time the value is modified. From a programmatic standpoint, replication occurs each time `setAttribute()` is called on the `HttpSession` object. This option can be resource intensive in cases where the session is being extensively modified.	Not applicable.
`onRequestEnd` (default)	Queue all changes made to HTTP session attributes, then replicate all changes just before the HTTP response is sent.	Replicate the current state of the bean after each EJB method call. The state is replicated frequently, but offers higher reliance.
`onShutdown`	Replicate the current state of the HTTP session whenever the JVM is terminated gracefully, such as with Ctrl-C. State is not replicated if the host is terminated unexpectedly, as in the case of a system crash. Because session state was not previously replicated, all session data is sent across the network at once upon JVM termination, which can impact network performance. This option can also significantly increase the amount of time needed for the JVM to shut down.	Replicate the current state of the bean whenever the JVM is terminated gracefully. State is not replicated if the host is terminated unexpectedly, as in the case of a system crash. Because bean state was not previously replicated, all state data is sent across the network at once upon JVM termination, which can impact network performance. This option may also significantly increase the amount of time needed for the JVM to shut down.

Table 7–2 <replication-policy> scope Attribute Values

scope Value	HttpSession	Stateful Session Bean
`modifedAttributes` (default)	Replicate only modified HTTP session attributes.	Not applicable.
`allAttributes`	Replicate all attribute values set on the HTTP session.	Replicate all attribute values set on the stateful session bean.

Note that the `<replication-policy>` element in `orion-application.xml` does not allow you to distinguish between Web and EJB components within an application. However, you can specify a different replication policy for an EJB component in the `replication` attribute of the `<session-deployment>` element within the component-specific `orion-ejb-jar.xml` configuration file.

See Table 7–3 for valid values for the `replication` attribute. For example:

```
<session-deployment name="MyStatefulVM" replication="onShutdown" />
```

```
<session-deployment name="MyEntity2" replication="onCall" />
```

The values in this file overrides the corresponding settings in `orion-application.xml`, effectively allowing you to set the replication policy for an EJB component in `orion-ejb-jar.xml` and the policy for Web components in `orion-application.xml`.

Table 7–3 Stateful Session EJB Replication Policy Configuration

replication Value	Description
`onRequestEnd`	Replicate the current state of the bean after each EJB method call. The state is replicated more frequently, but offers higher reliability in the event of host failure. This is the default value.
`onShutdown`	Replicate the current state of the bean whenever the JVM is terminated gracefully. State is not replicated if the host is terminated unexpectedly, as in the case of a system crash or a "`kill -9`" invocation in Unix/Linux.
`none`	Do not replicate data.

Managing the Number of Nodes to Replicate To

You can effectively limit the number of group members that state data is replicated to using the `write-quota` attribute of the `<cluster>` element. This functionality is similar to the "islands" concept used in previous OC4J releases, and makes it possible to reduce overhead by controlling the scope of state replication.

The default value for `write-quota` is `1`, indicating that state will be replicated to one other OC4J node within the cluster. This configuration should be sufficient for most scenarios.

To replicate state to all member nodes within the cluster, you must specify the total number of nodes within the cluster as the value of `write-quota`.

Configuring Multicast Replication

Multicast IP replication is the default replication protocol used in OC4J. In this mode, OC4J uses multicast packages to send and receive HTTP session and stateful session bean state changes. These packages are sent over the network where they are picked up by other OC4J processes using the same multicast address and port. Lost messages are identified and retransmitted, providing a reliable transmission service.

All OC4J servers must use same multicast address and port. The defaults used by OC4J multicast are `230.230.0.1` and `45566` for the port. These values can be changed in the configuration file, if necessary.

Multicast replication can be enabled between multiple application instances simply by adding the `<cluster>` element to each application's `orion-application.xml` file on each OC4J instance:

```
<orion-application ...>
  ...
  <cluster/>
</orion-application>
```

The next example specifies a new multicast address and port using the `ip` and `port` attributes. Note that the `bind_addr` attribute can also be used to specify which Network Interface Card (NIC) to bind to. This is useful if you have OC4J host machines with multiple network cards, each with a specific IP address, and you wish to define which NIC is used to send and receive the multicast messages.

```
<orion-application ...>
  ...
  <cluster allow-colocation="false">
    <replication-policy trigger="onShutdown" scope="allAttributes" />
    <protocol>
      <multicast ip="130.130.0.0" port="45577" bind_addr="140.83.24.10">
    </protocol>
   </cluster>
</orion-application>
```

Using an Existing JavaGroups Configuration for Multicast Replication

The multicast and peer-to-peer -based replication mechanisms provided by OC4J are built on the JavaGroups communication protocol stack. Ideally, you should use one of these OC4J mechanisms to provide in-memory replication of state data, as they utilize OC4J-specific configurations.

However, you do have the option of utilizing your own JavaGroups configuration within the OC4J clustering framework. This feature is enabled by specifying one of the following in the `<javagroups-config>` subelement within the `<cluster>` element:

- A string containing the JavaGroups configuration properties
- A URL to an XML configuration file containing this information

See the description of `<javagroups-config>` in "Overview of the <cluster> Element" on page 7-9 for details.

Configuring Peer-to-Peer Replication

OC4J supports replication in a peer-to-peer (P2P) topology, where cluster members ensure that session data is maintained in a redundant manner. The peer-to-peer clustering model uses TCP to establish connections with nodes within the cluster. The state data held in each application instance is unicast to each node.

Two peer-to-peer configurations are supported:

- Dynamic peer-to-peer, in which Oracle Process Manager and Notification Server (OPMN) is used to enable peer nodes to dynamically discover and communicate with one another. This configuration is only valid in an Oracle Application Server environment where OPMN is used to manage the various components, including OC4J

  See "Configuring Dynamic OMPN-Managed Peer-to-Peer Replication" for details.

- Static peer-to-peer, in which each node in the cluster is explicitly configured to recognize at least one other peer node. This configuration is primarily useful in a standalone OC4J environment, with a relatively small number of standalone OC4J instances clustered together.

  See "Configuring Static Peer-to-Peer Replication" for details.

Configuring Dynamic OMPN-Managed Peer-to-Peer Replication

In an Oracle Application Server environment, Oracle Process Manager and Notification Server (OPMN) is used to provide "dynamic" peer-to-peer replication. In this replication model, each OC4J node registers itself with OPMN. Each node then queries OPMN for the list, enabling it to dynamically discover and communicate with other available nodes in the cluster.

OPMN serves as a gossip server. Each OC4J node registers with OPMN via gossip client, also queries OPMN for other available nodes within its group. Nodes can then communicate with one another via JGroup stack. Nodes begin exchanging data for new sessions. In event one node is lost, another node can service its requests.

Each OC4J process sends periodic ONS (heartbeat) messages to OPMN to inform OPMN of current status, enabling OPMN to maintain a real-time list of available peer nodes, and to notify nodes when one has failed.

```
<orion-application ...>
  ...
  <cluster>
    <protocol>
      <peer>
        <opmn-discovery/>
      </peer>
    </protocol>
  </cluster>
</orion-application>
```

Configuring Static Peer-to-Peer Replication

In this configuration, the host address and port used for peer-to-peer communication of at least one other peer node in a peer-to-peer topology. As a node becomes aware of each of its peers, it also becomes aware each peer's peer(s) - with the end result that all of the nodes in the cluster become aware of one another.

The key challenge in this configuration is in ensuring that host and port definitions are kept up to date, which may present a significant management effort.

Note the following definitions in the example configurations, If the specified port is not available, OC4J will increment this value as specified in the range attribute until it finds a port to bind to. The start-port value is also specified

- The `start-port` attribute of the `<peer>` element specifies the initial port on the host that the local OC4J process will try to bind to for peer communcation. If this port is not available, OC4J will continue to increment this port until an available port is found.
- The `<node>` element specifies a peer node. The `host` and `port` attributes of the define the name of the node host and the port on that node that will be used for peer communication.
- The `range` attribute of the `<peer>` element applies to the ports specified in each `<node>` element - not to the value of the start-port attribute. It defines the number of times to increment the `port` value if the specified port is not available on a node.

The following example illustrates static peer-to-peer configurations as specified in `orion-application.xml` deployed with the `sample` application to three cluster nodes.

In this configuration, each node specifies one another node as its peer. The result is that all of the nodes within the cluster are able to establish connections with one another. Note that this scenario will only work if each node is started in succession; that is, `www1.company.com` must be started before `www2.company.com`. Otherwise, `www2.company.com` will not be able to "see" `www1.company.com`.

- First, `www1.company.com` specifies `www2.company.com` as its peer:

```
<orion-application ...>
  ...
```

```
  <cluster>
    <protocol>
      <peer start-port="7900" range="10" timeout="6000">
        <node host="www2.company.com" port="7900" />
      </peer>
    </protocol>
  </cluster>
</orion-application>
```

- Next, `www2.company.com` specifies `www3.company.com` as its peer:

```
<orion-application ...>
  ...
  <cluster>
    <protocol>
      <peer start-port="7900" range="10" timeout="6000">
        <node host="www3.company.com" port="7900" />
      </peer>
    </protocol>
  </cluster>
</orion-application>
```

- Finally, `www3.company.com` specifies `www1.company.com` as its peer:

```
<orion-application ...>
  ...
  <cluster>
    <protocol>
      <peer start-port="7900" range="10" timeout="6000">
        <node host="www1.company.com" port="7900" />
      </peer>
    </protocol>
  </cluster>
</orion-application>
```

An alternative configuration could have all of the nodes specifying the same node as a peer. For example, you could have the `www1.company.com` and `www3.company.com` nodes both specify `www3.company.com` as a peer. In this configuration, `www2.company.com` would have to be the first node started.

The downside of this configuration is that if this node fails for some reason, the remaining nodes will no longer be able to communicate with one another.

Configuring Database Replication

The new clustering framework provides the ability to replicate application state to a database. Data is persisted outside of the clustered OC4J framework, enabling the entire session to be recovered in the even of a catastrophic failure of all of the OC4J instances within the cluster.

> **Note:** Database replication can be very resource intensive, as it adds to network traffic and requires database resources.

The database connection data is supplied using a data source, specified using the `data-source` attribute of the `<database>` subelement. The `data-source` attribute takes the data source's `jndi-name` as specified in `data-sources.xml` as its value.

The following example will uses the "MyOracleDS" data source, identified by its JNDI name. Note that a data source must already have been created within OC4J to be used.

```
<orion-application ...>
  ...
  <cluster>
    <protocol>
      <database data-source="jdbc/MyOracleDS"/>
    </protocol>
  </cluster>
</orion-application>
```

Session data is persisted to the following tables in the database:

- `OC4J_HTTP_SESSION`, which stores metadata for an HTTP session
- `OC4J_HTTP_SESSION_VALUE`, which stores the values set by the application user on the HTTP session
- `OC4J_EJB_SESSION`, which stores the current state of a stateful session bean

The tables are created by OC4J the first time database replication is invoked. See Appendix C, "Overview of the Session State Tables" for details on the table schema.

The length of time session data is stored in the database is based on the session's time-to-live (TTL). A session is considered expired when the difference between the current database time and the time the session was last accessed is greater than the session timeout value. The actual equation for determining a session's TTL is:

```
(Current Database Time - Last Accessed Time) > Max Inactive Time
```

Expired sessions are removed from the database on the next execution of the OC4J task manager. See "Setting Task Manager Granularity" on page 9-1 for instructions on setting the task manager interval.

In the event that the OC4J server terminates without proper session termination, orphan records will be created in the database. These records will also be deleted the next time the task manager runs.

Disabling Clustering

Clustering can be disabled globally or for a specific application using the Boolean `enabled` attribute of the `<cluster>` element. Note that setting this attribute to `false` in an application's `orion-application.xml` file effectively removes the application from the cluster.

Overview of the <cluster> Element

The `<cluster>` element serves as the single mechanism for clustering configuration. It is used exclusively in the `ORACLE_HOME/j2ee/home/config/application.xml` file to configure clustering at the global level, and in application-specific `orion-application.xml` files for application-level clustering configuration.

<cluster>

Contains the clustering configuration for an enterprise application running within an OC4J instance.

Subelements of `<cluster>`:

```
<javagroups-config>
<replication-policy>
<protocol>
```

Attributes:

- `enabled`: Whether clustering is enabled for the application. The default is `true`. Note that setting this value at the application level overrides the global value, meaning that clustering can be disabled for a specific application.
- `group-name`: The name to use when establishing the replication group channels. If not supplied, the application name as defined in `server.xml`, the OC4J server configuration file, is used by default, and new group channels are created for each enterprise application.

  If a value is specified, the application and all child applications will use the channels associated with this group name.

  Note that this attribute is ignored if the `<database>` tag is included.
- `allow-colocation`: Whether to allow application state to be replicated to a node residing on the same host machine.

  The default is `true`. However, this attribute should be set to `false` if multiple hosts are available.

  If multiple OC4J instances are instantiated on the same machine, different listener ports must be specified for each instance in the `http-web-site`, `jms.xml` and `rmi.xml` configuration files.
- `write-quota`: The number of other group members the application state should be replicated to. This attribute makes it possible to reduce overhead by limiting the number of nodes state is written to, similar to the "islands" concept used in previous OC4J releases.

  The default is `1` node. This is the recommended setting for most clustering scenarios.

  Note that this attribute is ignored if the `<database>` tag is included.
- `cache-miss-delay`: The length of time, in milliseconds, to wait in-process for another group member to respond with a session if the session cannot be found locally. If the session cannot be found, the request will pause for the entire length of time specified. The default is `1000` milliseconds.

  Note that this attribute is ignored if the `<database>` tag is included.

<javagroups-config>

Contains data required to use the JavaGroups group communication protocol to replicate session state across nodes in the cluster.

Attributes:

- `url`: A link to a JavaGroups XML configuration file.
- `property-string`: A string containing the properties that define how the JavaGroups JChannel should be created.

<replication-policy>

The replication policy to apply, which defines when replication of data occurs and what data is replicated.

Attributes:

- `trigger`: The frequency at which replication occurs. See Table 7–1 on page 7-4 for the values for this attribute.

- `scope`: What data is replicated. See Table 7–2 on page 7-4 for the values for this attribute.

<protocol>

Defines the mechanism to use for data replication. Note that only one can be specified.

Subelements:

```
<multicast>
<peer>
<database>
```

<multicast>

Contains the configuration required to use multicast communication for replication. This is the default protocol used.

Attributes:

- `ip`: The multicast address to use. The OC4J default is `230.230.0.1`.
- `port`: The multicast port to use. The OC4J default is port `45566`.
- `bind_addr`: The Network Interface Card (NIC) to bind to. This is useful if you have OC4J host machines with multiple network cards, each with a specific IP address.

<peer>

Contains the configuration required to use peer-to-peer (P2P) communication for replication.

Subelements:

```
<opmn-discovery>
<node>
```

Attributes:

- `start-port`: The initial port on the node hose to attempt to allocate for peer communication. OC4J will continue to increment this value until an available port is found. The default is port `7800`.
- `range`: The number of times to increment the port value specified in each <node> sub-element while looking for a potential peer node. The default is `5` increments.
- `timeout`: The length of time, in milliseconds, to wait for a response from a peer while looking for a potential peer node. The default is `5000` milliseconds.
- `bind_addr`: The Network Interface Card (NIC) to bind to. This is useful if you have OC4J host machines with multiple network cards, each with a specific IP address.

<opmn-discovery>

Configures OC4J to use "dynamic" peer-to-peer replication in an Oracle Application Server environment.

<node>

Contains the hostname and port of a node to poll if using static peer-to-peer communication. One or more instances of this element can be supplied within a `<peer>` element.

Attributes:

- `host`: The hostname of the peer node as a URL.
- `port`: The port on the node to use for peer-to-peer communication. The default is port `7800`.

<database>

Contains the connection information required to persist state data to a database.

Attributes:

- `data-source`: The name of a data source containing the database connection information. This must be the value of the data source's `jndi-name` as specified in `data-sources.xml`.

8

Logging in OC4J

This chapter provides instructions on using the system and application logging features available in OC4J. It covers the following:

- Log Files Generated by OC4J
- Using Plain Text File Logging
- Using Oracle Diagnostic Logging (ODL)

Log Files Generated by OC4J

Each OC4J process generates a number of log files to aid in troubleshooting. If there are multiple processes running for an OC4J instance, multiple sets of log files are generated.

Log files are generated in different locations, depending on the component or application data is being recorded for. The logging configuration for each component or application is defined in component-specific XML configuration files.

Table 8–1 lists the name and location of the various log files generated, as well as the XML configuration file containing the logging configuration for each component.

Table 8–1 List of Log Files Generated for OC4J

Component	Default Log File Name and Location	Description	Configuration File
OC4J components using Java logging	`/j2ee/home/log/oc4j/log.xml`	All events from OC4J components that are using the `java.util.logging` framework. Messages are generated in XML format as part of the Oracle Diagnostic Logging (ODL) framework.	`/j2ee/home/config/j2ee-logging.xml`
Applications deployed into OC4J	`/j2ee/home/application-deployments/<app_name>/application.log`	All events, errors, and exceptions for a deployed application.	`/j2ee/home/application-deployments/<app_name>/orion-application.xml`
Global (default) application	`/j2ee/home/log/global-application.log`	All common events, errors, and exceptions related to applications.	`/j2ee/home/conffigapplication.xml`
Default Web site access logging	`/j2ee/home/log/http-web-access.log`	Logs all requests to the Web site. See "Configuring Access Logging" on page 11-10 for details on configuring Web site access logging.	`http-web-site.xml`

Table 8–1 (Cont.) List of Log Files Generated for OC4J

Component	Default Log File Name and Location	Description	Configuration File
JMS	`/j2ee/home/log/jms.log`	All JMS events and errors.	`jms.xml`
RMI	`/j2ee/home/log/rmi.log`	All RMI events and errors.	`rmi.xml`
OC4J server	`/j2ee/home/log/server.log`	All events not associated with a particular sub-system or application, including server startup/shutdown and internal server errors.	`server.xml`
Application Server Control Console	`/j2ee/home/application-deployments/ascontr/ascontrol-application.log`	All Application Server Control Console events.	`default-web-site.xml`

There are two types of log files that can be generated by OC4J:

- **Plain text log files**

  Plain text logs are the default log files used for OC4J components, and are ideal for use in a developer environment. The messages logged in these text files can be read with any editor, including the Oracle Enterprise Manager 10*g* Application Server Control Console.

- **Oracle Diagnostic Logging (ODL) Log Files**

  The messages logged in these files use an XML format that is viewable through Application Server Control Console. The key benefit of ODL logging is that it supports log file rotation.

Using Plain Text File Logging

Plain text logging is the default format used in OC4J.

This mechanism separates messages 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 log file.

- Enabling/Disabling Text File Logging
- Managing Text Log Files
- Viewing Text Log Files

Enabling/Disabling Text File Logging

Text logging is enabled or disabled through elements in the XML configuration files listed in Table 8–1, except for `http-web-site.xml` file. (See "Configuring Access Logging" on page 11-10 for details on configuring Web site access logging.)

Logging is enabled via the `<file>` subelement of the `<log>` element of the XML configuration file for each component. The element contains a `path` attribute which specifies the name and optionally the location of the log file generated.

To turn off text logging for a component, remove or comment out the `<file>` element from the appropriate configuration file. If you do not remove this line and enable ODL logging, both logging options will be enabled.

For example, to disable text logging for the Application A, comment out the following element in the application's `orion-application.xml` file:

```
<!-- <log>
<file path="application.log" />
</log> -->
```

Note that although both the ODL and text logging can be enabled simultaneously, one of these options should be disabled to save disk space.

Managing Text Log Files

It is important to monitor your log files, as text logging does not have any imposed size limits or log rotation capability. If left unchecked, log files will continue to grow and can overrun the disk.

The only way to manage these files is to stop OC4J, remove the file, then restart OC4J to start the log files over.

Viewing Text Log Files

All text log files are generated by default in the locations listed in Table 8–1, " List of Log Files Generated for OC4J" on page 8-1. However, you can specify a different location and filename using the `path` attribute of the `<log>`.

Paths can be absolute or relative to the configuration file. For example, the following sets a new locations the server log file in the `server.xml` configuration file:

```
<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>
```

Using Oracle Diagnostic Logging (ODL)

The *Oracle Diagnostic Logging* framework, or *ODL*, provides plug-in components that complement the standard Java framework to automatically integrate log data with Oracle log analysis tools.

In the ODL framework, log files are formatted as XML documents, enabling logs to be parsed and reused by other Oracle Application Server and custom developed components, including Application Server Control Console. Another key benefit of ODL logging is that unlike text-based logging, log file rotation is supported.

- Enabling/Disabling ODL Logging
- Managing ODL Log Files
- Viewing ODL Log Files

Enabling/Disabling ODL Logging

ODL logging is enabled by either un-commenting an existing ODL element or adding a new element to the appropriate XML configuration file.

- Add or un-comment the `<odl>` element within the `<log>` element in any of the XML files listed in Table 8–1, except for `http-web-site.xml` file. (See

"Configuring Access Logging" on page 11-10 for details on configuring Web site access logging.)

> **Note:** You can enable ODL logging for an application at the time the application is deployed by setting values for `odls` in the `log` property through the deployment plan editor. See the *Oracle Containers for J2EE Deployment Guide* for details on configuring an application using the deployment plan editor.

The `<odl>` element has the following attributes, all of which are required:

- `path`: The path to the directory where the `log.xml` files for this component will be generated. The path can be absolute or relative to the XML configuration file containing the entry.
- `max-file-size`: The maximum size, in kilobytes, that an individual log file is allowed to grow to. When this limit is reached, a new log file is generated.
- `max-directory-size`: Sets the maximum size, in kilobytes, allowed for the log file directory. When this limit is exceeded, log files are purged, beginning with the oldest files.

Files named `log.xml` are generated within the directory specified in the `path` attribute until the maximum directory size is reached. As the maximum file size (specified by the `max-file-size` attribute) is reached, subsequent log files are named `log2.xml`, `log3.xml`, and so on.

For example, the following entry in the `petstore` application's `orion-application.xml` file will cause `log.xml` files to be generated for this application in a `/petstore-xml` directory within `ORACLE_HOME/j2ee/home/log`. It will also set log files to a maximum of 1000KB and the directory maximum to 10,000KB.

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

Note that although both ODL and text file logging can be enabled, you should ideally disable one of these options to save disk space.

> **Note:** The `path`, `max-file-size` and `max-directory-size` attributes of the `<odl>` element are all required.

Managing ODL Log Files

A key benefit of the ODL framework is that it provides support for managing log files, including log file rotation. The maximum log file size and the maximum size of log directories can also be defined.

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 `log`*N*`.xml`, where N is a number starting at one. When the last log file is full, the following occurs:

1. The oldest log file is erased to provide space in the directory.

2. The `log.xml` file is written to the latest `log`*N*`.xml` file, where N increments by one over the most recent log file.

Viewing ODL Log Files

ODL-formatted log files can be viewed by clicking the **Logs** link in the Web-based Application Server Control Console, allowing administrators to aggregate and view the logging output generated by all components and applications running within OC4J from one centralized location. ODL log files are identified in the Log Files page by the `.xml` extension.

9

Task Manager and Thread Pool Configuration

This chapter provides guidelines for configuring the task manager and thread pool management features for an OC4J instance. It contains the following sections:

- Setting Task Manager Granularity
- Using Thread Pools

Setting Task Manager Granularity

The *task manager* is a background process that executes all pending tasks, such as timing out HTTP sessions and checking for changed configuration files. By default, it is started every second (1000 milliseconds).

The task manager can be an expensive process. For a system under significant load, the default value is probably not appropriate as it will trigger the cleanup operation every second, taking away valuable CPU cycles from applications. As such, you may want to consider configuring the process to run less frequently to boost application performance.

The interval at which the task manager executes is specified in milliseconds in the `taskmanager-granularity` attribute of the `<application-server>` element in the `server.xml` configuration file. This is an OC4J container-level parameter. The default is 1000 milliseconds.

For example, the following entry in `server.xml` configures the task manager to start every minute (60000 milliseconds):

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

Note that you must restart OC4J after making modifications to `server.xml`.

Note: You can also set this parameter through the `granularity` attribute on the `TaskManager` MBean, which is accessible through the JMX Browser in the Application Server Control Console.

See Chapter 10, "Using MBeans in OC4J" for details on accessing and using MBeans to manage OC4J processes.

Using Thread Pools

Thread pools create and store threads for use and re-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.

This section covers the following topics:

- Using the Default Thread Pool Configuration
- Creating Optional Thread Pool Configurations

Using the Default Thread Pool Configuration

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 remain idle until it is needed by a new request.

There is no limit on the number of threads that can be created within the pool in this configuration. Idle threads in the pool are re-used before a new thread is spawned, unless the number of requests exceeds the number of available threads. After 10 minutes of inactivity, idle threads are automatically destroyed.

This default configuration should by sufficient for most OC4J usage scenarios.

Creating Optional Thread Pool Configurations

Note: Configuring thread pools or modifying the default configuration should be considered expert-mode tasks. It is strongly recommended that the default single thread pool configuration be used.

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 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 9–1 on page 9-3 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 9–1 below describes the attributes of the `<global-thread-pool>` element. Note that this element is not included in `server.xml` by default.

Table 9–1 Attributes of <global-thread-pool>

Attribute	Description
`min`	The minimum number of threads to create in the pool. By default, a minimum number of threads are pre-allocated and placed in the thread pool when the container starts. If you add the `<global-thread-element>` element to `server.xml`, the default value is set to `20`. The minimum value that can be specified is `1`.
`max`	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. The default is `40`.
`queue`	The maximum number of requests that can be kept in the queue. The default is `80`.
`keepAlive`	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. To never destroy threads, set to `-1`. The default is `600000` milliseconds (10 minutes), which is also the minimum value allowed if not `-1`.
`cx-min`	The minimum number of threads to create in the connection thread pool. The minimum value that can be specified is `1`.
`cx-max`	The maximum number of threads that can be created in the connection pool. The default is `40`.
`cx-queue`	The maximum number of threads that can be kept in the queue in the connection pool. The default is `80`.
`cx-keepAlive`	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. To never destroy threads, set to `-1`. The default is `600000` milliseconds (10 minutes), which is also the minimum value allowed if not `-1`.
`debug`	If `true`, prints the application server thread pool information to the console at startup. The default is `false`.

An additional *work management thread pool* containing worker threads used by resource adapters, such as the JMS connector, can also be created. This pool is configured through the `<work-manager-thread-pool>` element.

The example below initializes a work management thread pool within the OC4J process.

```
<application-server ...>
 ...
 <work-manager-thread-pool min="10" debug="true"/>
```

```
 ...
</application-server>
```

Note that the work management thread pool is completely independent of the other two pools; the worker and connection thread pool attributes do not have to be configured in order to activate this pool.

Table 9–2below describes the attributes of the `<work-manager-thread-pool>` element. Note that this element is not included in `server.xml` by default.

Table 9–2 Attributes of <work-manager-thread-pool>

Attribute	Description
`min`	The minimum number of threads to create in the work management pool. The minimum value allowed is `1`.
`max`	The maximum number of threads that can be created in the work management thread pool. The default is `40`.
`queue`	The maximum number of threads that can be kept in the queue in the work management pool. The default is `0`, which means that no queue is maintained to handle a sudden burst of work requests.
`keepAlive`	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. To never destroy threads, set to `-1`. The default is `600000` milliseconds (10 minutes), which is also the minimum value allowed if not `-1`.
`debug`	If `true`, prints the application server work management thread pool information to the console at startup. The default is `false`.

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.

10

Using MBeans in OC4J

This chapter describes how the system MBeans provided with OC4J can be used to manage deployed applications, services and other resources within an OC4J instance. It includes the following topics:

- MBeans and Java Management Extensions (JMX) Support in OC4J
- Using the System MBean Browser
- Using JMX Notifications

MBeans and Java Management Extensions (JMX) Support in OC4J

OC4J provides support for the *Java Management Extensions (JMX) 1.2* specification, which allows standard interfaces to be created for managing resources, such as services, applications and resources, in a J2EE environment.

The Oracle Enterprise Manager 10*g* Application Server Control Console user interface is built on a JMX-compliant client that can be used to completely manage and monitor an OC4J instance. The JMX functionality provided through Application Server Control Console is enabled through Java components known as *MBeans*, which are discussed in the next section.

JMX manageable resources within OC4J include:

- The OC4J server
- Applications and Web modules running within an OC4J instance
- J2EE services, such as JTA and JMS
- OC4J processes, such as Task Manager
- Data source and security configuration

This section discusses the following topics:

- What Are MBeans?
- Overview of the Top-Level OC4J System MBeans
- When Do Changes Made Via MBeans Take Effect?
- How Are Changes Persisted?

What Are MBeans?

An *MBean*, or *managed bean*, is a Java object that represents a JMX manageable resource. MBeans are defined in the *J2EE Management Specification (JSR-77)*, which is part of the J2EE 1.4 specification as published by Sun Microsystems.

Each manageable resource within OC4J is managed through an instance of the appropriate MBean. For example, two instance of the `J2EESWebSite` MBean are created at OC4J startup: One representing the default Web site configuration, the other the Web site used to access the Application Server Control Console interface. These MBean instances will be named `J2EEWebSite:http-web-site` and `J2EEWebSite:ascontrol-web-site` respectively.

Each system MBean provided with OC4J exposes a management interface that is accessible through the System MBean Browser. An MBean's interface is comprised of:

- *Attributes*, value s of any type that the JMX client can get or set remotely. Attributes are analogous to properties set on a JavaBean. For example, the `state` attribute of `J2EEApplication:petstore` MBean indicates whether or not the application is currently running.
- *Operations*, methods that the JMX client can invoke on the MBean. For example, the `stop` operation can be used to stop the `petstore` application and all of its child applications.
- *Notifications* that can be generated broadcast errors or specific events, such as when a new account is created. For example, a notification can be sent to alert you that the `petstore` application has stopped.

As noted ealier, the Application Server Control Console application is built on top of the system MBeans. When you set a property or perform a task in the user interface, you are actually setting an attribute or invoking an operation on an underlying MBean.

To provide you with greater flexibility, Application Server Control Console also provides direct access to the system MBeans provided with OC4J through the *System MBean Browser* component. See "Using the System MBean Browser" on page 10-5 for details on using this management tool.

Overview of the Top-Level OC4J System MBeans

The following table provides an overview of the top-level OC4J system MBeans exposed through the System MBean Browser interface.

Table 10–1 Top-Level OC4J System MBeans

MBean	Description
`J2EEDomain`	Represents a management domain. This is the top level management object. All other MBeans bound to the domain are visible beneath this node in the System MBean Browser.
`J2EEServer`	Represents a single OC4J instance.
`ClassLoading`	Provides access to all class loading related state in an OC4J server instance. Includes an operation to execute the more than 15 built-in queries provided to aid in troubleshooting class loading issues on a running OC4J instance. This MBean lazily creates instances of the `ClassLoader` MBean, each representing an instantiated classloader.

Table 10–1 (Cont.) Top-Level OC4J System MBeans

MBean	Description
`J2EEApplication`	Represents a J2EE application deployed into the OC4J instance. Additional MBean instances are visible as child nodes representing the various components of the application: ▪ `OC4JWebModule`: Represents the properties set through the OC4J-specific `orion-web.xml` deployment descriptor generated for a Web module deployed as part of the J2EE application. ▪ `WebModule`: Represents the properties set through the J2EE `web.xml` deployment descriptor packaged with a WAR file. Instances of the `JSP` and `Servlet` MBeans are created for active JSPs and servlets within the Web module.
`J2EELogging`	Represents a Java Logger component defined in the `j2ee-logging.xml` file. For an overview of the Java logging framework, including log levels, visit Sun's site at `http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html`.
`J2EEWebSite`	Represents a Web site defined within the OC4J server. See Chapter 11, "Managing Web Sites in OC4J" for details on Web site configuration.
`JDBCDriver`	Represents a specific JDBC driver.
`JMSAdministratorResource`	Represents the OC4J JMS server used by the OC4J instance. Includes operations for managing the OC4J JMS server and JMS connection factories, as well as adding/removing destinations.
`JMSResource`	Represents a JMS server instance. Additional MBean instances are visible as child nodes include: ▪ `JMSPersistenceResource`: Describes a persistent message store used by JMS.
`JMSResource`	Displays statistics on messages (by type), active handlers and active connections from the JMS server. Child MBeans contain statistics on connection, destination and durable subscriber resources.
`JNDINamespace`	Returns an XML document containing all JNDI bindings for all applications deployed into the OC4J instance.
`JNDIResource`	Returns all JNDI bindings for a specific application.
`JSPConfig`	Configures the OC4J JSP container. See the *Oracle Containers for J2EE Support for JavaServer Pages Developer's Guide* for documentation of the various configuration values. Note that any changes made to MBean attributes require an OC4J server restart to take effect.
`JTAResource`	Represents a transaction manager instance. Note that invoking the `configureCoordinator` operation on this MBean requires an OC4J server restart for the new two-phase commit-coordinator configuration to take effect.
`JVM`	Describes a Java Virtual Machine that an OC4J instance is running within. Includes an operation to get/set system properties and force garbage collection to start.
`SecurityProvider`	Used to manage security for a specific application. Note that a restart of the corresponding application or the OC4J server is required for some attributes and operations to take effect.
`TaskManager`	Describes an OC4J task manager instance. This MBean can be used to set task manager granularity.
`ThreadPool`	Represents a single instantiated thread pool. Use to set the maximum and minimum number of threads in the pool.

Table 10–1 (Cont.) Top-Level OC4J System MBeans

MBean	Description
`TimerService`	Represents an instance of the EJB timer. See the *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide* for details.

When Do Changes Made Via MBeans Take Effect?

Changes can be made to a managed component via an MBean while the component is either stopped or running.

In general, changes made to managed component - values set on an attribute or the results of an operation - are available immediately in the OC4J runtime.

In some cases, however, new attribute values or operation results will require a restart - of the OC4J server, the affected application or even of the MBean - before becoming available in the OC4J runtime. In these cases, the MBean and the Application Server Control Console will display the "new" value; however, the "old" value will continue to be used in the OC4J runtime until the required restart is completed.

For example, suppose you change the value of the `timeout` attribute of the `JSPConfig` MBean from `30` to `15`. The new value of `15` will be displayed both in the MBean and in the JSP Container Properties page in Application Server Control Console. However, because all changes to `JSPConfig` attributes require a restart of the OC4J server, the old value of `30` will continue to be used until the server is restarted.

If a restart is required, the System MBean Browser displays a Required Restart property noting the required actions. Table 10–2 below lists the values for this property.

Table 10–2 Required Restart Property Values

Value	Impact
OC4J Restart	Indicates that the OC4J server instance must be restarted.
Application Restart	Indicates that the J2EE application under which the MBean is registered must be restarted. MBeans that belong to this category are displayed under the `J2EEApplication` node in the the navigation pane to the left of the console.
MBean Restart	Indicates that the affected MBean must be restarted.

Change is managed at the individual attribute/operation level, rather than at the MBean level. This means that an MBean might contain attributes that require a restart before a new value is available in the runtime, and other attributes that become available immediately.

How Are Changes Persisted?

Persistent data set via an MBean is written to the appropriate XML configuration file(s). For example, new values set on attributes of the `JSPConfig` MBean are written to the `global-web-application.xml` configuration file.

Whether an MBean persists data is indicated by the Persist Policy property displayed in the System MBean Browser.

Table 10–3 Persist Policy Property Values

Value	Impact
OnUpdate	Any persistent data set on the MBean is written immediately to the appropriate configuration file(s) at the time the attibute change is applied or the operation is invoked.
Never	Data set on the MBean is not persisted, but exists only in runtime memory.

Using the System MBean Browser

The System MBean Browser is a component of the Web-based Oracle Enterprise Manager 10*g* Application Server Control Console user interface. The console itself is relatively simple to use. To use this feature:

1. Launch the Application Server Control Console.
2. Click the **Administration** link.
3. Click **System MBean Browser**.
4. Specific MBean instances are accessed through the navigation pane to the left of the console. Expand a node in the navigation pane and drill down to the MBean you wish to access.
5. Click the **Attributes** tab in the right-hand pane to access the selected MBean's attributes. If you modify any attribute values, click the **Apply Changes** button to apply your changes to the OC4J runtime.

   Note: The **Apply Changes** button will only be visible if the browser page contains at least one attribute with a modifiable value.

6. Click the **Operations** tab to access the MBean's operations. After selecting a specific operation, click the **Invoke** button to call it.

Using JMX Notifications

Many of the system MBeans provided with OC4J include the ability to generate notifications triggered by a state change registered by the MBean. The following section describes how to subscribe to and view MBean-generated notifications.

Note that not all MBeans generate notifications.

Subscribing to Notifications

You can subscribe to notifications either through the System MBean Browser or the Notification Subscriptions page.

To subscribe to one or more of an MBean's notifications through the System MBean Browser:

1. Click the **Administration** link in the Application Server Control Console.
2. Click **System MBean Browser**.

3. Specific MBean instances are accessed through the navigation pane to the left of the console. Expand a node in the navigation pane and drill down to the MBean you wish to access.
4. Click the **Notifications** tab in the right-hand pane to access the selected MBean's notifications. Note that if this tab is not present, the MBean does not generate notifications.
5. Check the **Subscribe** box.
6. Click the **Apply** button.

To subscribe to notifications generated by multiple MBeans through the Notification Subscriptions page.

1. Click the **Administration** link in the Application Server Control Console.
2. Click the **Notification Subscription** icon. All MBeans that generate notifications are displayed.
3. Check the **Subscribe** box for each notification you wish to subscribe to.
4. Click the **Apply** button.

Using Application-Specific MBeans

Vendor-supplied MBeans deployed with a J2EE application into OC4J can be accessed via the application's "home page" in the Application Server Control Console user interface. Through the user interface, you can view and set attributes and invoke operations on application-specific MBeans, just as you can with the OC4J system MBeans.

1. Click the **Applications** link in the Application Server Control Console.
2. Click the name of the application the MBeans belong to. This opens the "home page" for the application.
3. Click the **Application Defined MBeans** link. The MBeans defined by the application are listed on the page displayed.
4. Click the **Attributes** tab in the right-hand pane to access the selected MBean's attributes. If you modify any attribute values, click the **Apply Changes** button to apply your changes to the OC4J runtime.

> **Note:** The **Apply Changes** button will only be visible if the browser page contains at least one attribute with a modifiable value.

5. Click the **Operations** tab to access the MBean's operations. After selecting a specific operation, click the **Invoke** button to execute.

11

Managing Web Sites in OC4J

This chapter explains how additional Web sites can be configured in an OC4J standalone environment to provide access to Web applications deployed into the OC4J instance. It also explains how to configure and enable a secure Web site utilizing Secure Socket Layer (SSL) communication between the client and OC4J using HTTPS.

The following sections are included:

> **Note:** Note that this chapter focuses exclusively on configuring Web sites served by the built-in Web server bundled with OC4J. Web sites created in a managed OC4J environment are served by the Oracle HTTP Server.

What Is a Web Site in OC4J?

Every Web module deployed into an OC4J instance must be bound to a Web site through with it will be accessed. This binding is typically performed as part of the application deployment process.

Multiple Web sites can be configured for each OC4J instance, one to listen on each protocol. (A Web site cannot listen on more than one protocol.) Web sites are defined and configured using XML configuration files in the `ORACLE_HOME/j2ee/home/config` directory. Web site configurations created by default include:

- `http-web-site.xml`

  This file defines a default Web site used to send HTTP requests directly to OC4J. It is included by default in a standalone OC4J configuration.

  This site is configured to listen for HTTP requests on port `8888` and can be used to receive requests by any application deployed into the OC4J instance.

- `default-web-site.xml`

This file configures the Web site used by OC4J to receive requests forwarded from the Oracle HTTP Server (OHS). It is included by default in an Oracle Application Server configuration. By default, it is configured to listen on port is `3301`.

The Oracle HTTP Server (OHS) forwards incoming HTTP requests to the `mod_oc4j` module, which in turn passes the requests via the Apache JServ Protocol (AJP) to the OC4J server.

- `ascontrol-web-site.xml`

  This file defines a Web site configured on port `1810` for use by the Application Server Control Console Web module. It is included by default in both the standalone OC4J and Oracle Application Server configurations.

Note that the location of each Web site configuration file is declared in `server.xml`, the OC4J server configuration file. See "Referencing a Web Site Configuration File in server.xml" on page 11-4 for details.

Managing Web Site Connection Configuration

Key differences exist in how Web site connection data is managed in standalone OC4J versus Oracle Application Server environments.

Standalone OC4J

In a standalone OC4J configuration, the protocol and listener ports used by a Web site must be explicitly defined in the corresponding `*-web-site.xml` configuration file.

Oracle Application Server

In an Oracle Application Server environment, in which Oracle HTTP Server (OHS) is used to manage incoming requests and Oracle Process Manager and Notification Server (OPMN) is used to manage OHS and OC4J instances, OPMN can be used to efficiently manage Web site protocol and port configuration.

In this scenario, the protocol a Web site will use is specified within a `<port>` element defined for the Web site in `opmn.xml`, the OPMN configuration file. A range of listener ports is also specified in this file; OPMN is then able to dynamically assign a port for each OC4J process started within the OC4J instance that uses the Web site. Allowing OPMN to select from a range of ports in this manner avoids potential conflicts among OC4J processes.

The `<port>` element used to configure a Web site connection in `opmn.xml` is defined in the `opmn.xml` configuration file, which is located in the `ORACLE_HOME/opmn/conf` directory. The syntax of the element is as follows:

```
<port id="<webSiteName>" protocol="http|https|ajp|ajps"
 range="<startPort-endPort>"/>
```

Attributes of ths element are:

- `id` defines the name of the Web site, which is the name of the Web site configuration file minus the `.xml` extension
- `protocol` specifies the protocol the Web site will receive requests through. Note that if either `https` or `ajps` is specified, the value of the secure attribute of the root `<web-site>` element in the `*-web-site.xml` file will be overridden.
- `range` specifies the start and end ports for the range of ports available for assignment by OPMN.

The element is added or included with other <port> elements defining connection protocols in the <process-type> element defining the OC4J instance, which is a sub-element of the <ias-component> element where the id attribute equals OC4J.

For example, to configure a secure-web-site to use AJPS, you would add this entry to the OC4J home instance definition the opmn.xml file on the host machine:

```
<ias-component id="OC4J">
  <process-type id="home" module-id="OC4J" status="enabled">
   ...
    <port id="default-web-site" protocol="ajps" range="12501-12600"/>
    <port id="rmi" range="12401-12500">
    <port id="jms" range="12601-12700">
    <process-set id="default" numprocs="5"/>
  </process-type>
</ias-component>
```

The protocol and port values specified in opmn.xml will override any corresponding values set in the Web site configuration file. Note that using OPMN to manage Web site protocol and port settings is not required in an Oracle Application Server environment; you can opt to not set these values in opmn.xml, and instead set the values directly in the appropriate Web site configuration file.

Note the value of the numprocs attribute of the <process-set> element, which indicates that five OC4J processes will be created for the home instance. OPMN will dynamically assign a port to listen for AJPS requests to each of these processes.

Steps for Configuring a Web Site in OC4J

Bringing a new Web site to life is essentially a two-step process:

1. Create the XML configuration file
2. Add a reference to the Web site configuration file in server.xml, the OC4J configuration file

After this last step, the Web site will be available for binding with applications. The following topics provide details on Web site configuration.

- Creating the New Web Site Configuration File
- Referencing a Web Site Configuration File in server.xml
- Sharing Web Applications Between Web Sites
- Specifying the Cookie Domain

Creating the New Web Site Configuration File

> **Note:** This discussion provides instructions on configuring a Web site using XML configuration files. Web sites can also be created using the admin.jar command line utility. See "Managing Web Sites" on page 6-7 for instructions on creating Web sites using admin.jar.

The key functions of a Web site configuration file are the following:

- It binds specified Web modules to the Web site by identifying each Web module to bind, the J2EE application it belongs to and the context path portion of the URL to use to access the site (for example, /ascontrol).

- It defines key settings for the Web site, such as the host name and port number.

Web site configuration files are named according to the following convention:

The most straightforward way to create a new configuration file is to make a copy of the default Web site configuration file, `http-web-site`, which is located in the `ORACLE_HOMEj2ee/home/config` directory. Name the file according to the following convention:

```
<webSiteName>-web-site.xml
```

The typical configuration file includes a root `<web-site>` element containing attributes that specify the following:

- `port`: The Web site listener port.
- `display-name`: The for-display name of the Web site.
- `virtual-hosts`: Any additional domains bound to this Web site.

The `<web-site>` element also typically contains the following sub-elements:

- An `<access-log>` element specifying the log file that requests sent to the site are logged to
- A `<default-web-app>` element identifying the parent Web application that global settings are inherited from
- One or more `<web-app>` sub-elements for each Web module bound to the Web site. These elements are added by OC4J when each application is bound to the Web site; however, they can be added to the file manually if desired. At a minimum, each `<web-app>` element has the following:
  - An `application` attribute to specify the name of the J2EE application to which the Web module belongs (the same as the EAR file name without the `.ear` extension)
  - A `name` attribute to specify the name of the Web module (the same as the WAR file name without the `.war` extension)
  - A `root` attribute to specify the context path on this Web site to which the Web module is to be bound

As an example, assume that you will create a configuration file named `petstore-web-site.xml` which defines a Web site that will be used by the `petstore` application. The root `<web-site>` element within this file will contain all of the required configuration data, as shown below:

```
<web-site xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/
  web-site-10_0.xsd"  port="8887" display-name="The Petstore Web Site">
    <default-web-app application="default" name="defaultWebApp" />
    <web-app application="petstore" name="petstore" load-on-startup="true"
      root="/petstore" />
    <access-log path="../log/petstore-web-access.log" />
</web-site>
```

See the <web-site> element description on page B-13 for details on the structure of this element.

Referencing a Web Site Configuration File in server.xml

The location of every Web site configuration file must be referenced in a `<web-site>` element in the `server.xml`, the OC4J configuration file located in the `J2EE_`

HOME/config directory. Note that applications will not be able to bind to the Web site unless this declaration exists in server.xml.

Each <web-site> element specifies the path and file name for the corresponding Web site XML file, as in the following sample server.xml entries:

```
<application-server ... >
  <web-site path="./http-web-site.xml" />
  <web-site path="./ascontrol-web-site.xml" />
  <web-site path="./petstore-web-site.xml" />
</application-server>
```

In this example, the locations of all of the Web site configuration files, including the new petstore-web-site.xml, are relative to the location of server.xml.

If OC4J polling is disabled, you must restart OC4J for changes to server.xml to take effect.

Sharing Web Applications Between Web Sites

Sharing a Web application implies the sharing of everything that makes up the application, including sessions, servlet instances, and context values.

A typical use for this mode is to share a Web application between an HTTP site and an HTTPS site on the same context path - essentially *binding* the application to the two different Web sites. This results in improved performance because only sensitive information is encrypted as needed, rather than requiring that all information in a request be encrypted.

Another benefit is that the cookie, rather than the SSL certificate, is used to track the session. The SSL certificate uses 50K to store each certificate when tracking it, which sometimes results in an "out of memory" problem for the session before the session times out. This could possibly make the Web application less secure, but might be necessary to work around issues such as SSL session timeouts not being properly supported in some browsers.

You can set an application as shared by setting the shared attribute of the <web-app> element to true in the *-web-site.xml file defining each Web site the application is bound to. This attribute is false by default.

For example, the sample petstore application is shared between both the default OC4J Web site - which listens on port 8888 - and a new secure Web site listening on port 4443 - by adding or modifying the following <web-app> elements in each Web site configuration file. This configuration will enable the application to accept both HTTP and HTTPS connections.

The <web-app> entry in http-web-site.xml:

```
<web-site xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/
  web-site-10_0.xsd"  port="8888" display-name="OC4J 10g (10.1.3) HTTP Web Site">
    <web-app application="petstore" name="petstore" load-on-startup="true"
      root="/petstore" shared="true"/>
    <access-log path="../log/http-web-access.log" />
</web-site>
```

The similar entry in secure-web-site.xml:

```
<web-site xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/
  web-site-10_0.xsd" port="4443" secure ="true" display-name="My Secure Web Site">
  <web-app application="petstore" name="petstore" load-on-startup="true"
```

```
   root="/petstore" shared="true"/>
  <access-log path="../log/secure-web-access.log" />
  <ssl-config factory="com.evermind.ssl.JSSESSLServerSocketFactory"
    keystore="../../server.keystore" keystore-password="welcome"
    provider="com.sun.net.ssl.internal.ssl.Provider" />
</web-site>
```

Specifying the Cookie Domain

Note that you can optionally set the *cookie domain* to a specific value. This causes the domain to be set to the specified value at the time a cookie is created, resulting in a cookie that can be sent by a Web browser to any Web site within the domain.

If the domain is not specified, the browser defaults to the domain of the fully qualified server name, such as `site1.acme.com`. In this case, the browser would not be able to forward the cookie to `site2.acme.com`. However, if the cookie domain is explicitly set to `acme.com`, the cookie could be sent to either server.

Set the `cookie-domain` attribute in the `<session-tracking>` element in the J2EE standard `orion-web.xml` file for the application. The cookie-domain attribute contains the DNS domain with at least two components of the domain name provided. For example:

```
<session-tracking cookie-domain=".oracle.com" />
```

Note: If the domain is set to `acme.com`, the cookie will not actually be sent to `acme.com`. In this case, `acme.com` must be redirected at the Web server level to `www.acme.com` to enable cookies to be shared among all subdomains of `acme.com`.

The domain cookies will be redirected to should have at least two dots for `org/com/edu/mil/net` domains and three for other domains; for example, `www.acme.com`.

Configuring a Secure Web Site in OC4J

OC4J supports Secure Socket Layer (SSL) communication between the client and OC4J using HTTPS. You can configure the default Web site to utilize SSL to create secure connections, or can create an additional site and bind it to one or more Web applications.

For details on SSL keys and certificates, see the *Oracle Containers for J2EE Security Guide*.

This section covers the following topics:

- Creating the Secure Web Site Configuration File

Note: This discussion provides instructions on configuring a secure Web site using XML configuration files. Secure Web sites can also be created using the `admin.jar` command line utility. See "Managing Web Sites" on page 6-7 for instructions on creating Web sites using `admin.jar`.

Creating the Secure Web Site Configuration File

Specify the appropriate SSL settings under the <web-site> element, as illustrated in the example below.

```
<web-site xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/
  web-site-10_0.xsd" port="4443" secure ="true" display-name="My Secure Web Site">
  <access-log path="../log/secure-web-access.log" />
  <ssl-config factory="com.evermind.ssl.JSSESSLServerSocketFactory"
    keystore="../../server.keystore" keystore-password="welcome"
    provider="com.sun.net.ssl.internal.ssl.Provider" />
</web-site>
```

Note the additions to <web-site>, shown in **bold**:

- Add a secure attribute with the value set to true. Setting secure="true" specifies that the HTTP protocol is to use an SSL socket.
- Set the port attribute to an available port. The default for SSL ports is 443; in the example above, the port attribute is set to 4443.
- Add the <ssl-config> element. This element is required whenever the secure flag is set to true. This element takes the following:
  - The optional factory attribute is used to specify the third-party SSLSecureSocketFactory implementation to use if the application is not using JSSE.

    The Oracle implementation - com.evermind.ssl.JSSESSLServerSocketFactory - is used by default. (Note that although the default implementation is shown in the example, it is implicit and does not need to be specified.)

    If the application uses a third-party SSLServerSocketFactory implementation, you can use <property> subelements of <ssl-config> to send parameters to the factory.
  - The keystore and keystore-password attributes specify the directory path and password for the keystore. The specified keystore must contain the certificates of any clients that are authorized to connect to OC4J through HTTPS. The value of keystore can indicate either an absolute or relative directory path and includes the file name.
  - The optional provider attribute can be used to specify a security provider to use.

    By default, the Sun Microsystems implementation - com.sun.net.ssl.internal.ssl.Provider - is used. (Note that although the default implementation is shown in the example, it is implicit and does not need to be specified.)
  - One or more <property> elements containing parameters to pass to the SSLSecureSocketFactory. Each element contains a name attribute and a value attribute, enabling you to specify parameters as name/value pairs.

When the Web site configuration file is ready, add a <web-site> element referencing it to server.xml, the OC4J configuration file located in the J2EE_HOME/config directory. Note that applications will not be able to bind to the Web site unless this notation exists in server.xml. For example:

```
<application-server ... >
  <web-site path="./http-web-site.xml" />
```

```
  <web-site path="./ascontrol-web-site.xml" />
  <web-site path="./secure-web-site.xml" />
</application-server>
```

When configuration is complete, OC4J listens for SSL HTTP requests on one port and non-SSL HTTP requests on another. You can disable either SSL requests or non-SSL requests by commenting out the appropriate `*-web-site.xml` in the `server.xml` configuration file.

```
<!-- <web-site path="./secure-web-site.xml" /> commented out to remove SSL -->
```

For more information about elements and attributes of the `<web-site>`, `<web-app>`, and `<session-tracking>` elements, see the XML Appendix in the *Oracle Containers for J2EE Servlet Developer's Guide.*

Requiring Client Authentication

You can require that clients be authenticated by the server by setting the `needs-client-auth` attribute of the `<ssl-config>` element to `"true"`. For example:

```
<web-site ... secure="true" ... >
 <ssl-config factory="com.evermind.ssl.JSSESSLServerSocketFactory"
   keystore="../../server.keystore" keystore-password="welcome"
   needs-client-auth="true" />
</web-site>
```

This step sets up a mode where OC4J accepts or rejects a client entity for secure communication, depending on its identity. The `needs-client-auth` attribute instructs OC4J to request the client certificate chain upon connection. If the root certificate of the client is recognized, then the client is accepted.

The keystore specified in the `<ssl-config>` element must contain the certificates of any clients that are authorized to connect to OC4J through HTTPS.

Requesting Client Authentication with OC4J

OC4J supports a "client-authentication mode in which the server explicitly requests authentication from the client before the server will communicate with the client. In this case, the client must have its own certificate. The client authenticates itself by sending a certificate and a certificate chain that ends with a root certificate. OC4J can be configured to accept only root certificates from a specified list in establishing a chain of trust back to the client.

A certificate that OC4J trusts is called a *trust point*. This is the first certificate that OC4J encounters in the chain from the client that matches one in its own keystore. There are three ways to configure trust:

- The client certificate is in the keystore.
- One of the intermediate certificate authority certificates in the client's chain is in the keystore.
- The root certificate authority certificate in the client's chain is in the keystore.

OC4J verifies that the entire certificate chain up to and including the trust point is valid to prevent any forged certificates.

If you request client authentication with the `needs-client-auth` attribute, perform the following:

1. Decide which of the certificates in the client's chain is to be your trust point. Ensure that you either have control of the issue of certificates using this trust point or that you trust the certificate authority as an issuer.
2. Import the intermediate or root certificate in the server keystore as a trust point for authentication of the client certificate.
3. If you do not want OC4J to have access to certain trust points, make sure that these trust points are not in the keystore.
4. Execute the preceding steps to create the client certificate, which includes the intermediate or root certificate installed in the server. If you wish to trust another certificate authority, obtain a certificate from that authority.
5. Save the certificate in a file on the client.
6. Provide the certificate on the client initiation of the HTTPS connection.
   a. If the client is a browser, set the certificate in the client browser security area.
   b. If the client is a Java client, you must programmatically present the client certificate and the certificate chain when initiating the HTTPS connection.

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.

Starting/Stopping Web Sites

A Web site is available by default once it has been configured on an OC4J instance. However, Application Server Control Console provides the ability to stop and start inidividual Web sites through the **Administration>J2EE Websites** pages. These pages also display the configuration for each Web site, and provide access to the Web modules bound to each site.

Note: Note that the `ascontrol-web-site`, used by Application Server Control Console, cannot be stopped via the user interface.

---

1. Click the **Administration** link in the Application Server Control Console.
2. Click the **J2EE Websites** icon under **Administration Tasks>Properties**. The Web sites configured on the OC4J server instance are listed on the page displayed.
3. Click the name of the desired Web site.

Configuring Access Logging

OC4J provides the ability to generate an *access log* for each Web site, which records requests submitted by Web applications to the Web site.

Access logs can be generated as either text-based log files or as Oracle Diagnostic Logging (ODL) files, which are generated in XML format that is viewable through Application Server Control Console. Only one type of access logging may be configured for a Web site.

Access logging is configured for a Web site in the Web site configuration file (`*-web-site.xml`) using either the `<access-log>` or `<odl-access-log>` elements. If neither of these elements are included in the configuration file, access logs are not generated for the Web site.

This section covers the following topics:

- Configuring Text-Based Access Logging
- Configuring ODL Access Logging
- Enabling/Disabling Access Logging for a Web Module/Application

Configuring Text-Based Access Logging

Text-based access logging is configured through the `<access-log>` subelement of the root `<web-site>` element in a Web site's configuration file. This element has the following attributes:

- `format`: Specifies one or more of several supported variables that result in information being prepended to log entries. Supported variables are `$time` `$request`, `$ip`, `$host`, `$path`, `$size`, `$method`, `$protocol`, `$user`, `$status`, `$referer`, `$time`, `$agent`, `$cookie`, `$header`, and `$mime`. Between variables, you can type in any separator characters that you want to appear between values in the log message. The default setting is as follows:

  ```
  "$ip - $user - [$time] '$request' $status $size"
  ```

  As an example, this results in log messages such as the following (with the second message wrapping around to a second line):

  ```
  148.87.1.180 - - [17/Nov/2004:10:23:18 -0800] 'GET / HTTP/1.1' 200 2929
  ```

```
148.87.1.180 - - [17/Nov/2004:10:23:53 -0800] 'GET
/webservices/statefulTest HTTP/1.1' 200 301
```

In this example, the user is null, the time is in brackets (as specified in the `format` setting), the request is in single-quotes (as specified), and the status and size in the first message are 200 and 2929, respectively.

- `path`: Specifies the path and name of the access log. This can be an absolute path or a path relative to the `j2ee/home/config` directory. The default setting in `http-web-site.xml` is the following:

  ```
  path="../log/default-web-access.log"
  ```

- `split`: Specifies how often to begin a new access log. Supported values are `"none"` (equivalent to "never", which is the default), `"hour"`, `"day"`, `"week"`, or `"month"`. Note that if `split` is specified, the suffix attribute (documented below) can be used to specify timestamp data to append to the file name.
- `suffix`: Specifies timestamp information to append to the base file name of the logs if the `split` attribute is specified.

  The default `suffix` setting is `"-yyyy-MM-dd"`.

  As an example, assume the following `<access-log>` element with split specified, using the default `suffix` value:

  ```
  <access-log path="../log/mysite-web-access.log" split="day" />
  ```

  The log file generated will be named as follows:

  ```
  mysite-web-access-2004-11-17.log
  ```

  The format used is that of `java.text.SimpleDateFormat`, and symbols used in `suffix` settings are according to the symbology of that class. Characters are case-sensitive, as described in the `SimpleDateFormat` documentation. For information about `SimpleDateFormat` and the format symbols it uses, refer to the current Sun Microsystems Javadoc at the following location:

  `http://java.sun.com/j2se/`

By default, text-based access logs are generated in the `ORACLE_HOME/j2ee/home/log` directory and are named using the format `<webSiteName>-web-access.log`. However, you can specify a different location and filename using the `path` attribute of the `<access-log>` element. Timestamp data can also be appended to the filename using the `suffix` attribute.

It is important to monitor text-based access log files, as this logging format does not support log rotation. If left unchecked, access log files will continue to grow and can overrun the disk.

Configuring ODL Access Logging

In the ODL framework, log files are formatted as XML documents. A key benefit of ODL access logging is that unlike text-based logging, log file rotation is supported.

ODL access logging is configured through the `<odl-access-log>` subelement of the root `<web-site>` element in a Web site's configuration file. This element has the following attributes, all of which are required:

- `path`: The path to the directory where the `log.xml` files for the Web site will be generated. The path can be absolute or relative to the XML configuration file containing the entry.

- `max-file-size`: The maximum size, in kilobytes, that an individual log file is allowed to grow to. When this limit is reached, a new log file is generated.
- `max-directory-size`: Sets the maximum size, in kilobytes, allowed for the log file directory. When this limit is exceeded, log files are purged, beginning with the oldest files.

New files named `log.xml` are generated within the directory specified in the `path` attribute until the maximum directory size is reached. Each log file is equal to or less than the maximum specified in the attributes.

For example, the following entry in the http-web-site.xml file will cause `log.xml` files to be generated in a `/default-web-access` directory within `ORACLE_HOME/j2ee/home/log`. It will also set log files to a maximum of 1000KB and the directory maximum to 10,000KB.

```
<web-site>
  <odl-access-log path="../log/default-web-access/" max-file-size="1000"
    max-directory-size="10000" />
</web-site>
```

Enabling/Disabling Access Logging for a Web Module/Application

If either the `<access-log>` or `<odl-access-log>` elements are defined in a Web site configuration file, access logging is enabled by default for all Web modules within applications bound to the Web site.

However, it is possible to disable access logging for a specific module by setting the `access-log` attribute of the application-specific `<web-app>` element in the configuration file to `false`. This may be desirable in situations where a Web module submits such a massive number of requests that text-based access log files will quickly become bloated.

For example, the following entry in `http-web-site.xml` disables access logging for the `default` application's DMS Web component, but leaves text-based access logging for the `admin_web` module enabled:

```
<web-site ...>
  <web-app application="default" name="dms0" root="/dmsoc4j" access-log="false" />
  <web-app application="default" name="admin_web" root="/adminoc4j" />
  <access-log path="../log/http-web-access.log" />
</web-site>
```

12

Registering DTDs and XSDs with OC4J

This chapter describes the process for registering new entities - specifically any vendor-specific DTDs and XSDs used to define the format of XML deployment descriptors - within OC4J, which is required if XML file validation will be performed. It contains the following topics:

Why Do DTDs/XSDs Have to be Registered?

OC4J provides the ability to validate XML deployment descriptors at the time the files are read. This feature is enabled by passing the `-validateXML` argument on the `oc4j.jar` command line at OC4J startup. See Chapter 4, "Runtime Configuration" for details on command line options.

Validation requires that the DTD or XSD defining an XML document be registered with the OC4J server. If this entity is not registered, XML validation may not occur.

When an XML document is read, the parser passes one or more keys identifying the DTD or XSD declared in the document to an OC4J component known as the *Entity Resolver*. The Entity Resolver resolves the location of the registered entity and returns it to the parser, enabling the XML document to be validated.

Two types of keys are used to reference an entity: A *public identifier* and a *system identifier*, both of which are declared in the XML document. Consider the following declaration in `orion-application.xml`, which references the DTD used to define the file format:

```
<!DOCTYPE orion-application
PUBLIC "-//Evermind//DTD J2EE Application runtime 1.2//EN"
"http://xmlns.oracle.com/ias/dtds/orion-application.dtd">
```

- The *public identifier* is the string `"-//Evermind//DTD J2EE Application runtime 1.2//EN"`
- The *system identifier* is the URL `"http://xmlns.oracle.com/ias/dtds/orion-application.dtd"`

To enable the Entity Resolver to locate the entity, one or both of these identifiers must be registered with OC4J through entries in the `entity-resolver-config.xml` file. The entity's location must also be specified in this file.

Note that by default, `entity-resolver-config.xml` already contains registration entries for the standard J2EE DTDs and XSDs, as well as for all OC4J-specific XSDs. As such, you are only required to add entries for non-J2EE or non-OC4J entities.

Registering a DTD or XSD

To register a DTD or XSD with OC4J, you must add it to the `entity-resolver-config.xml` file, which is located in the `ORACLE_HOME/j2ee/home/config` directory on the OC4J host machine.

Each entity is declared in an `<entity>` element, which includes the following sub-elements:

- `<description>`: Contains an optional description of the entity.
- `<public-id>`: Contains the entity's public identifier.
- `<system-id>`: Contains the entity's system identifier.

  Either `<public-id>` or `<system-id>` must be specified; however, you are not required to specify both.
- `<location>`: Points to the entity's location. The location can be either the fully qualified path to the entity or a URL that can be resolved locally.

The following `<entity>` element will register `acme-web.dtd` with OC4J. Both the public and system identifiers, which are declared in the `<!DOCTYPE>` element within an XML document, are registered.

```
<entity>
  <description>acme-web-2_0.dtd</description>
  <public-id>-//Acme//Acme web Descriptor 2.0//EN</public-id>
  <system-id>http://xmlns.acme.com/dtd/acme-web-2_0.dtd</system-id>
  <location>META-INF/acme-web-2_0.dtd</location>
</entity>
```

The next example will register `acme-application.xsd` with OC4J. The system identifier is declared in either the `xsi:schemaLocation` or the `xsi:noNamespaceSchemaLocation` attributes of the root element within an XML document.

```
<entity>
  <description>acme-application-1_0.xsd</description>
  <public-id />
  <system-id>http://xmlns.acme.com/schema/acme-application-1_0.xsd</system-id>
  <location>META-INF/acme-application-1_0.xsd</location>
</entity>
```

Note: The OC4J server must be re-started after making changes to `entity-resolver-config.xml`.

A

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

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

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 the `-Xmx` system property to the desired amount of memory 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.

Application Performance Impacted by 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 global `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 using the links 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, 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.

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.

B

Configuration Files Used in OC4J

This chapter provides detailed documentation on the XML files used to store configuration data for the OC4J server and J2EE applications and modules deployed into it.

Overview of the XML Files Used By OC4J

The configuration data for an OC4J server instance and the applications and modules deployed into it is persisted in a number of XML files. Figure B–1 provides an overview of these XML files and their respective roles.

Note that schemas defining the Oracle-proprietary XML files used by OC4J can be viewed at the following link:

`http://www.oracle.com/technology/oracleas/schema/index.html`

Figure B–1 XML Files Used By OC4J

Table B–1 describes the role and function for each OC4J server-level XML file as well as the global configuration files displayed in the preceding figure. Unless otherwise indicated, all of these files are installed in the `ORACLE_HOME/j2ee/home/config` directory by default.

Table B–1 Server-Level and Global Configuration Files

XML Configuration File	Features/Components
`server.xml`	The OC4J server configuration file. Configures the server and points to the XML files that add to this file, such as `jms.xml` for JMS support. The listing of other XML files enables the services to be configured in separate files, but the `server.xml` file denotes that they be used for the OC4J configuration.
`data-sources.xml`	Contains the OC4J data source configuration for all databases used by applications within OC4J.
`rmi.xml`	Contains OC4J RMI port configuration and RMI tunneling over HTTP.

Table B–1 (Cont.) Server-Level and Global Configuration Files

XML Configuration File	Features/Components
jms.xml	Contains the OC4J JMS configuration for Destination topics and queues that are used by JMS and MDBs in OC4J.
system-application.xml	Contains the configuration for the system application, which is the parent of all other applications installed in the OC4J instance. This file includes common settings that serve as default settings for other applications.
application.xml	Contains the configuration for the default application. All user-deployed applications and standalone are deployed to this application by default. Not that this file is completely different from the J2EE standard deployment descriptor, also named application.xml.
global-web-application.xml	An Oracle-specific file for configuring the servlet and JSP containers within OC4J.
oc4j-connectors.xml	Contains global OC4J-specific configuration data for connectors installed in the OC4J instance.
*-web-site.xml	An OC4J-specific file that contains configuration data for a Web site created within the OC4J instance. It is typically installed in the ORACLE_HOME/j2ee/home/config directory, but may be installed in a different location. The default Web site configured with OC4J is defined by default-web-site.xml.

Table B–2 describes the role and function for the various application-level XML files displayed in the preceding figure.

Table B–2 Application-Level Configuration Files

XML Configuration File	Features/Components
application.xml	The J2EE application standard J2EE application descriptor file. 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.
orion-application.xml	The OC4J-specific deployment descriptor, which contains configuration data for a specific deployed application.
web.xml	The J2EE Web application deployment descriptor, 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.
orion-web.xml	Extends the standard J2EE descriptor with application-level OC4J-specific configuration data such as whether or not OC4J features like developer mode or auto-reload of JSPs are enabled.

Table B–2 (Cont.) Application-Level Configuration Files

XML Configuration File	Features/Components
`ejb-jar.xml`	The J2EE EJB module deployment descriptor, included in the EJB JAR file. Defines the specific structural characteristics and dependencies of the Enterprise JavaBeans within a JAR, and provides instructions for the EJB container about how the beans expect to interact with the container.
`orion-ejb-jar.xml`	The OC4J-specific deployment descriptor. Defines OC4J-specific configuration data for all EJBs within an archive, including EJB pool settings, time-out and retry settings, JNDI mappings and finder method specifications. Also includes properties for the TopLink persistence manager.
`application-client.xml`	The J2EE application client configuration file. Describes the EJB modules and other resources used by a J2EE application client packaged in an archive.
`orion-application-client.xml`	Contains OC4J deployment data, including JNDI mappings to an EJB's home interface or to external resources such as a data source, JMS queue or mail session.
`ra.xml`	The J2EE standard deployment descriptor. Contains information on implementation code, configuration properties and security settings for a resource adapter packaged within a RAR file.
`oc4j-ra.xml`	Contains OC4J-specific deployment configuration data for a single resource adapter. This data includes EIS connection information, JNDI name to be used, connection pooling parameters, and resource principal mappings.
`webservices.xml`	The J2EE standard Web services deployment descriptor. Describes a Web service, including WSDL information and JAX-RPC mapping data, for a Web Service application packaged within a WAR file.
`oracle-webservices.xml`	Defines properties used by the OC4J Web services container, such as whether to expose the WSDL file. It also defines endpoint addresses and data specific to EJBs implemented as Web services. The file can be packaged in either a WAR or an EJB JAR containing a Web service.

Overview of the OC4J Server Configuration File (server.xml)

The OC4J configuration file, `server.xml`, is located in the `J2EE_HOME/config` directory. It is the starting point for configuration of the OC4J server and all J2EE applications, Web applications and Web sites enabled within the server.

Unless specifically instructed to do so in the OC4J documentation, you should not have to edit `server.xml` manually, as notations are added and updated as needed by OC4J.

The `server.xml` file points to the application descriptor of each application on OC4J, either directly or indirectly. In the case of a typical J2EE application, `server.xml` points to the EAR file (or extracted EAR top-level directory) and, therefore, to the `application.xml` file that the EAR file contains. In the case of the OC4J global

application, the `server.xml` file points directly to the OC4J global application descriptor.

The `server.xml` file also 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>` element denotes the name and location of the `rmi.xml` file.
- The `<jms-config>` element denotes the name and location of the `jms.xml` file.
- The `<global-application>` element denotes the name and location of the global `application.xml` file.
- The `<global-web-app-config>` element denotes the name and location of the `global-web-application.xml` file.
- The `<web-site>` element 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.

Other elements for `server.xml` are described in "server.xml Element Descriptions" on page B-9.

The `server.xml` file format is described by `application-server-10_1.xsd`, which can be viewed at the following link:

`http://www.oracle.com/technology/oracleas/schema/index.html`

Example of a server.xml File

Below is an example `server.xml`, with `<!-- comments -->` to describe the various sections:

```
<application-server  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/
  application-server-10_1.xsd"  application-directory="../applications"
  deployment-directory="../application-deployments"
  connector-directory="../connectors"
  schema-major-version="10" schema-minor-version="0" >
  <!-- Shared library definitions -->
  <shared-library name="global.libraries" version="1.0" library-compatible="true">
    <code-source path="../applib"/>
    <code-source path="../../../sqlj/lib"/>
    <code-source path="../../../lib/dsv2.jar"/>
  </shared-library>
  <shared-library name="global.tag.libraries" version="1.0"
   library-compatible="true">
    <code-source path="../jsp/lib/taglib/standard.jar"/>
  </shared-library>
  <!-- J2EE services -->
  <rmi-config path="./rmi.xml" />
  <sep-config path="./internal-settings.xml" />
  <jms-config path="./jms.xml" />
  <javacache-config path="../../../javacache/admin/javacache.xml" />
  <!-- Logging -->
  <j2ee-logging-config path="./j2ee-logging.xml" />
  <log>
    <file path="../log/server.log" />
  </log>
  <java-compiler name="javac" in-process="false" encoding="ISO8859_1"
   extdirs="c:\sdk\jdk\jre\lib\ext" />
```

```
  <!--  Default application configuration -->
  <global-application name="default" path="application.xml" />
  <!-- Deployed application configuration -->
  <application name="petstore" path="../applications\petstore.ear" start="true" />
  <application name="ascontrol" path="../applications\ascontrol.ear"
   start="true" />
  <!-- Default Web application configuration file -->
  <global-web-app-config path="global-web-application.xml" />
  <!-- Transaction Manager configuration file -->
  <transaction-manager-config path="transaction-manager.xml" />
  <!-- Configuration files for enabled Web sites -->
  <web-site path="./http-web-site.xml" />
  <web-site path="./ascontrol-web-site.xml" />
</application-server>
```

<application-server>

Required? Required; one only

Child elements:

This is the root element of the OC4J configuration file.

Table B–3 <myotherelement> Attributes

Name	Description
`application-directory`	Values: string Default: `n/a` The target directory for deployed EAR files; also where they are unpacked.
`application-auto-deploy-directory`	Values: string Default: `n/a` The location of a directory into which EAR files can be copied, triggering automatic deployment of the archive.
`connector-directory`	Values: string Default: `n/a` The target directory for resource adapters.
`deployment-directory`	Values: string Default: `n/a` The target directory for copied or generated OC4J descriptors and EJB wrapper classes.
`check-for-updates`	Values: `all|adminClientOnly|none` Default: `none` Enables OC4J polling, which automatically checks for changes made to currently deployed applications and modules, and redeploys any components that have been modified. See Chapter 6, "Automatic Deployment in OC4J" in the *Oracle Containers for J2EE Deployment Guide* for an explanation of supported values and the impact of each. The default is `none`, which disables this feature.

Table B–3 (Cont.) <myotherelement> Attributes

Name	Description
`localhostIsAdmin`	Values: Boolean Default: `true` If `true`, allows easier access it the process initiating the administrative operation is a process local to the OC4J host machine.
`taskmanager-granularity`	Values: int Default: `1000` The interval at which the task manager performs its duties, specified in milliseconds. The default is every second (1000 milliseconds).

<application>

Parent element: <application-server>

Required? Optional; multiple allowed

Child elements:

Defines a J2EE application deployed into the OC4J instance. The `<application>` element defining an application is added to `server.xml` by OC4J at the time the application is deployed. As such, there is generally no need to manually modify this element.

Table B–4 <myotherelement> Attributes

Name	Description
`name`	Values: string Default: `n/a` The application name; typically the same as the EAR file name without the `.ear` extension.
`path`	Values: string Default: `n/a` The location of the EAR file or the extracted EAR top-level directory. As such, the path indirectly points to the J2EE standard `application.xml` descriptor packaged with the application.
`start`	Values: Boolean Default: `true` If `true`, the application is started with OC4J and is available to serve requests or for configuration through JMX. If `false`, the application is not started with OC4J, meaning it is not available to serve requests. However, it is available for configuration through JMX.

<code-source>

Parent element: <shared-library>

Child elements:

Required? Required; one only

Specifies the path to a JAR or ZIP file included in a shared library.

Paths may be absolute if outside of the `/shared-lib` directory, or can be relative to the subdirectory containing the JAR files within the `/shared-lib/<library_name>` directory. If relative, only the archive file name needs to be supplied as the value of the `path` attribute.

You can optionally set `path="*"` to force OC4J to consume all of the archives within the shared library subdirectory.

For additional information on configuring and using shared libraries, see Chapter 9, "Using Shared Libraries in OC4J".

<data-sources>

Parent element: XXX

Child elements:

XXX

Table B–5 <myotherelement> Attributes

Name	Description
`name`	Values: string Default: `n/a` The application name; typically the same as the EAR file name without the `.ear` extension.

<execution-order>

Parent element: startup-class, shutdown-class

Required? Optional; one only

Child elements:

Specifies the order of execution for each startup class. Specify an integer that designates in what order the classes are executed.

<global-application>

Parent element: application-server

Required? Required; one only

Child elements:

Specifies the OC4J global application, otherwise known as the `default` application, which, by default, is the parent of all other applications. The `name` attribute defines its name; the `path` attribute specifies what to use as the OC4J global application descriptor.

Table B–6 <myotherelement> Attributes

Name	Description
`name`	Values: string Default: `default` The global application name.
`path`	Values: string Default: `application.xml` The filename and path for the global application descriptor file. The default descriptor is `ORACLE_HOME/j2ee/home/config/application.xml`.

<global-thread-pool>

Parent element: XXX

Required? Required; one only

Child elements:

Contains the configuration for a single thread pool within the OC4J process. See "Using Thread Pools" on page 9-1 for details. Note that the `cx-*` attributes are used to configure the second thread pool if two pools are being created.

Table B–7 <myotherelement> Attributes

Name	Description
`min`	Values: string Default: `n/a` The minimum number of threads that OC4J can simultaneously execute.
`max`	Values: string Default: `n/a` The maximum number of threads that OC4J can simultaneously execute.
`queue`	Values: string Default: `n/a` The maximum number of requests that can be kept in the queue.
`keep-alive`	Values: string Default: `n/a` The length of time, in 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.
`cx-max`	Values: string Default: `n/a` The minimum number of connection threads that OC4J can simultaneously execute in the second pool.

Table B–7 (Cont.) <myotherelement> Attributes

Name	Description
cx-min	Values: string Default: n/a The maximum number of connection threads that OC4J can simultaneously execute in the second pool.
cx-queue	Values: string Default: n/a The maximum number of requests that can be kept in the queue in the second pool.

<global-web-app-config>

Parent element: XXX

Required? Required; one only

Child elements:

Identifies the OC4J global web application, which by default is the parent of all other Web applications. The `name` attribute defines its name; the `path` attribute specifies what to use as the OC4J global application descriptor.

The name and root directory path of the default Web application are specified in the global application descriptor, and the default Web application is bound to a Web site through the `http-web-site.xml` file for OC4J standalone (`default-web-site.xml` in Oracle Application Server). In OC4J standalone, the default context path for the default Web application is "/".

Table B–8 <myotherelement> Attributes

Name	Description
path	Values: string Default: `global-web-application.xml` The filename and path of the global Web application descriptor file. The default descriptor is `ORACLE_HOME/j2ee/home/config/global-web-application.xml`.

<init-param>

Parent element: startup-class, shutdown-class

Required? Optional; multiple allowed

Child elements: param-name, param-value

Specifies initialization parameters within a `<startup-class>` or `<shutdown-class>` element. Contains 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.

Table B–9 <myotherelement> Attributes

Name	Description
`path`	Values: string Default: `global-web-application.xml` The filename and path of the global Web application descriptor file. The default descriptor is `ORACLE_HOME/j2ee/home/config/global-web-application.xml`.

<jms-config>

Parent element: application-server

Required? Optional; only one allowed

Child elements:

Specifies the file to use as the OC4J JMS descriptor.

Table B–10 <myotherelement> Attributes

Name	Description
`path`	Values: string Default: `jms.xml` The filename and path of the OC4J JMS descriptor

<rmi-config>

Parent element: application-server

Required? Optional; only one allowed

Child elements:

Specifies the file to use as the OC4J RMI descriptor.

Table B–11 <myotherelement> Attributes

Name	Description
`path`	Values: string Default: `rmi.xml` The filename and path of the OC4J RMI descriptor.

<shared-library>

Parent element: application-server

Required? Optional; multiple allowed

Child elements: code-source, import-shared-library

Defines a shared library within OC4J. For additional information on configuring and using shared libraries, see Chapter 9, "Using Shared Libraries in OC4J".

Table B–12 <myotherelement> Attributes

Name	Description
`name`	Values: string Default: required The name of the shared library directory created within the `/shared-lib` directory.
`version`	Values: string Default: required The version number that serves as the name of the subdirectory containing the shared library's archive files in the `/shared-lib/<library_name>` directory.
`library-compatible`	Values: Boolean Default: `false` XXX Ask Bryan

<shutdown-class>

Parent element: shutdown-classes

Required? Optional; multiple allowed

Child elements: execution-order, init-param

Defines a shutdown class to execute before OC4J terminates within the `<startup-classes>` element.

Table B–13 <myotherelement> Attributes

Name	Description
`classname`	Values: string Default: required The name of the class that implements the `oracle.j2ee.server.OC4JShutdown` interface.

<startup-class>

Parent element: startup-classes

Required? Optional; multiple allowed

Child elements: execution-order, init-param

Defines a startup class to execute on OC4J initialization within the `<startup-classes>` element.

Table B–14 <myotherelement> Attributes

Name	Description
`classname`	Values: string Default: required The name of the class that implements the `oracle.j2ee.server.OC4JStartup` interface.

Table B–14 (Cont.) <myotherelement> Attributes

Name	Description
`failure-is-fatal`	Values: Boolean Default: `false` If `true`, OC4J logs an exception and exits when an exception is thrown. If `false`, OC4J logs the exception and continues.

<transaction-manager-config>

Parent element: application-server

Required? Optional; only one allowed

Child elements:

Specifies the transaction manager configuration file.

Table B–15 <myotherelement> Attributes

Name	Description
`path`	Values: string Default: `transaction-manager.xml` The filename and path of the transaction manager configuration file. The default file is `ORACLE_HOME/j2ee/home/config/transaction-manager.xml`.

<web-site>

Parent element: application-server

Required? Optional; multiple allowed

Child elements:

References the configuration file for a single Web site defined within OC4J. A <web-site> element must be created for each Web site; otherwise, the site will not be enabled within OC4J. See Chapter 11, "Managing Web Sites in OC4J" for details.

Table B–16 <myotherelement> Attributes

Name	Description
`path`	Values: string Default: The filename and path of the `*-web-site.xml` configuration file defining the Web site.

Overview of the Web Site Configuration Files (*-web-site.xml)

The element descriptions in this section apply to any OC4J Web site configuration file, including `default-web-site.xml` (Oracle Application Server) and `http-web-site.xml` (OC4J standalone default Web site).

<web-site>

Required? Required; one only

Child elements:

<description>
<frontend>
<web-app>
<default-web-app>
<user-web-apps>
<access-log>
<odl-access-log>
<ssl-config>

This is the root element for a Web site configuration file.

Table B–17 <web-site> Attributes

Name	Description
display-name	Values: string Default: n/a Optionally defines a user-friendly or informal Web site name.
host	Values: string Default: Specifies the host for this Web site, as either a DNS host name or an IP address. If a server is a "multi-home" machine (having multiple IP addresses), you can use the "[ALL]" setting to listen to all IP addresses.
log-request-info	Values: Boolean Default: false Specifies whether to write information about the incoming request into the Web site log if an error occurs. The Web site log is enabled through either the <access-log> or <odl-access-log> element, described later in this section. ("OC4J Logging" on page 2-15 provides additional information about enabling logs, including the Web site log.)
max-request-size	Values: string Default: 15000 Sets a maximum size, in bytes, for incoming HTTP requests. If a client sends a request that exceeds this maximum, it will receive a "request entity too large" error. The default maximum is 15000.

Table B–17 (Cont.) <web-site> Attributes

Name	Description
`secure`	Values: Boolean Default: `false` Specifies whether to support Secure Socket Layer (SSL) functionality. For a protocol setting of "`ajp13`" (used in an Oracle Application Server environment), a "`true`" setting results in secure AJP protocol between Oracle HTTP Server and OC4J. For a protocol setting of "`http`" (used in OC4J standalone), a "`true`" setting results in HTTPS protocol between the client and OC4J. Also note that a `secure="true"` setting requires that you use the `<ssl-config>` element (a subelement under the `<web-site>` element) to specify the keystore path and password. This element is documented later in this section. SSL and HTTPS features are also available through Oracle HTTP Server for communication between Oracle HTTP Server and the client. For information, see *Oracle Application Server Security Guide*.
`protocol`	Values: string Default: n/a Specifies the protocol that the Web site is using. Possible values are "`http`" and "`ajp13`" (for AJP, the default). In a production environment with Oracle Application Server, you should use only the "`ajp13`" setting. The AJP protocol is for use with Oracle HTTP Server and `mod_oc4j`. Note that each protocol must have a corresponding port, and each port must have a corresponding protocol. The "`http`" setting is for OC4J standalone. To use either an "`ajp13`" or "`http`" setting in secure mode (SSL), you must set the `secure` flag to "`true`" and use the `<ssl-config>` subelement to specify the keystore path and password. This element is documented later in this section.
`port`	Values: string Default: n/a Specifies the port number for this Web site. Each port must have a corresponding protocol, and each protocol must have a corresponding port. In OC4J standalone, a port setting of 8888 is used by default for direct access to the OC4J listener, but you can change this as desired. In an Oracle Application Server environment, this port setting is overridden by OPMN, the Oracle Process Management and Notification system. Oracle Application Server uses port `7777` by default for access through Oracle HTTP Server with Oracle Application Server Web Cache enabled. In a UNIX environment, port numbers less than `1024` require root privileges for access. Also note that if there is no port specification from the client browser, port `80` is assumed for HTTP protocol and port `443` for HTTPS.

Table B–17 (Cont.) <web-site> Attributes

Name	Description
`use-keep-alive`	Values: Boolean Default: `true` Typical behavior for a servlet container is to close a connection once a request has been completed. With a `use-keep-alive` setting of `"true"`, however, a connection is maintained across requests. For AJP protocol, connections are always maintained and this attribute is ignored. For other protocols, the default is `"true"`; disabling it may cause significant performance loss.
`virtual-hosts`	Values: string Default: n/a This optional attribute is useful for virtual sites sharing the same IP address. The value is a comma-delimited list of host names tied to this Web site.

<description>

You can use the body of this element for a brief description of the Web site.

<frontend>

This element specifies a perceived front-end host and port of this Web site as seen by HTTP clients. When the site is behind a load balancer or firewall, the `<frontend>` specification is necessary to provide appropriate information to Web application code for functionality such as URL rewriting. Using the host and port specified in the `<frontend>` element, the back-end server running the application knows to refer to the front-end, instead of to itself, in any URL rewriting. This way, subsequent requests properly come in through the front-end again, instead of trying to access the back-end directly.

Attributes of `<frontend>`:

- `host`: Specifies the host name of the front-end server, such as `"www.acme.com"`.
- `port`: Specifies the port number of the front-end server, such as `"80"`.

<web-app>

This element binds a particular Web module to this Web site. It specifies the name of a J2EE application archive (EAR file name minus the `.ear` extension) from the `server.xml` file, and the name of a Web module within the J2EE application. The Web module is defined in the J2EE `application.xml` file in the application EAR file (or possibly in the `orion-application.xml` file in the EAR file). The Web module is bound at the location specified by the `<web-app>` element `root` attribute.

> **Note:** It is possible to deploy a WAR file by itself, instead of within an EAR file. In OC4J standalone, such Web applications are added to the OC4J default application. (In OC4J, there must always be a parent application of some sort.) See "OC4J Default Application and Default Web Application" on page 5-25 for more information.
>
> In this scenario, the Web site XML file `<web-app>` element specifies the name of the default application rather than the name of a J2EE application archive. More details are provided in the attribute descriptions and examples that follow.

Mapping to and from Web site XML files, particularly with respect to the `application` and `name` attributes, is shown in examples elsewhere in this document. See "Example: Mappings to and from Web Site Descriptors" on page 5-19 (for a typical scenario of deploying a WAR file within an EAR file) and "Deploying an Independent WAR File to OC4J Standalone" on page 5-32 (for the scenario of deploying a WAR file by itself to the OC4J default application).

Attributes of `<web-app>`:

- `access-log`: Specifies whether OC4J access logging, which logs requests to the Web site, is enabled for the Web module. The default is `true`. If log file management becomes an issue, set to `false` to disable access logging for the module.

  See the descriptions of the `<access-log>` and `<odl-access-log>` elements within this section for more on access log configuration.

  > **Note:** If the Web site configuration file does not contain either an `<access-log>` or `<odl-access-log>` element, access logging is disabled, regardless of the value of this attribute.

- `application`: Specifies the J2EE application archive name, which is the EAR file name without the `.ear` extension, and which corresponds to the `name` attribute of an `<application>` element in the `server.xml` file.

  > **Notes:** If you deploy a WAR file by itself in OC4J standalone, using the OC4J default application as the parent, then the `application` attribute instead reflects the name of the default application, according to the `<global-application>` element in the `server.xml` file.

- `load-on-startup`: This is an optional attribute to specify whether the Web module should be pre-loaded on application startup. Otherwise, it is loaded upon the first request for it. Supported values are `"true"` and `"false"`. The default is `false`; however, this value is explicitly set to `true` when the module/application is deployed through Oracle Enterprise Manager 10*g* Application Server Control Console.

  Pre-loading of individual servlets, through `<load-on-startup>` elements in the application `web.xml` file, is possible only if this `<web-app>` element `load-on-startup` attribute is enabled. See "Servlet Preloading" on page 2-6 for more information.

- `max-inactivity-time`: This is an optional integer attribute to specify the number of minutes of inactivity after which OC4J will shut down the Web module. By default, a Web module is never shut down due to inactivity.
- `name`: Specifies the name of a Web module within the specified J2EE application, and corresponds to the `<web-uri>` value (without the `.war` extension) of a `<web>` subelement of a `<module>` element in the J2EE `application.xml` file. The J2EE `application.xml` file is in the EAR file.

Notes:

- If you deploy a WAR file by itself in OC4J standalone, using the OC4J default application as the parent, then the `name` attribute instead reflects the value of the `id` attribute of a `<web-module>` element in the OC4J global `application.xml` file. This is the `application.xml` file for the OC4J default application, but be aware that it is not a standard J2EE file; it is OC4J-specific. Also note that the `id` attribute, as with the `name` attribute of the `<web-app>` element, does not have the `.war` extension.
- An application can also have an `orion-application.xml` file in the EAR file, with `<web-module>` elements that define additional Web modules, or even override Web modules defined in the J2EE `application.xml` file (although overriding is *not* advised). The `name` attribute can reflect the `id` value of a `<web-module>` element in `orion-application.xml`, instead of reflecting a `<web-uri>` value in the J2EE `application.xml` file.
- The `orion-application.xml` file uses the same DTD as the OC4J global `application.xml` file; namely, `orion-application.dtd`.

- `root`: Specifies the path to which the Web module is to be bound, which defines the context path portion of the URL used to invoke the module. For example, if the Web module `CatalogApp` at Web site `www.example.com` is bound to the `root` setting `"/catalog"`, then it can be invoked as follows:

```
http://www.example.com/catalog
```

Important:

- The `root` attribute overrides the `<context-root>` value of the corresponding `<web>` element in the J2EE `application.xml` file. Even though the `<context-root>` element is mandatory in an `application.xml` file, its value is not used by OC4J.
- Specifying a `root` setting of "/" will override the OC4J default Web application. This setting or a null setting is not allowed by the `admin.jar` utility when binding a Web application to the Web site.

- `shared`: Allows sharing of a published Web module between Web sites, when a Web site is defined by a particular pairing of a protocol and a port. Supported values are `"true"` and `"false"` (default). Sharing implies the sharing of everything that makes up a Web application, including sessions, servlet instances, and context values. An example is to share a Web application in OC4J standalone between an HTTP site and an HTTPS site at the same context path, when SSL is required for some but not all the communications. (Performance is improved by encrypting only sensitive information, rather than all information.)

  If an HTTPS Web application is marked as shared, its session tracking strategy reverts from SSL session tracking to session tracking through cookies or URL rewriting. This could possibly make the Web application less secure, but may be necessary to work around issues such as SSL session timeouts not being properly supported in some browsers.

  Important: Use `shared="true"` only in OC4J standalone.

<default-web-app>

This element creates a reference to the default Web application of this Web site. For users, it is meaningful only in an OC4J standalone environment. See "OC4J Default Application and Default Web Application" on page 5-25 for more information.

In an Oracle Application Server environment, the OC4J default Web application has system-level functionality but is not otherwise meaningful. See "OC4J Default Web Application in Oracle Application Server" on page 5-40.

The `<default-web-app>` element uses the same attributes as the `<web-app>` element described immediately preceding, but note that the default setting of `load-on-startup` is `"true"`.

<user-web-apps>

Use this element to support user directories and applications. Each user has his or her own Web module and associated `web-application.xml` file. User applications are reached at `/username/` from the server root.

Attributes of `<user-web-apps>`:

- `max-inactivity-time`: Optional integer attribute to specify the number of minutes of inactivity after which OC4J will shut down the Web module. By default, a Web module is never shut down due to inactivity.
- `path`: Specifies a path to specify the local directory of the user application, including a wildcard for the user name. The default path setting in UNIX, for example, is `"/home/username"`, where *username* is replaced by the particular user name.

<access-log>

Use this element to enable text-based access logging for this Web site and to specify information about the access log, including the path, file name, and what information is included. The log file is where incoming requests (each access of the Web site) are logged.

Alternatively, use the `<odl-access-log>` element (described next after this element) for ODL logging.

Note: Do not use both <access-log> and <odl-access-log>; you can use only one type of logging. (The last element in the Web site XML file would take precedence, but do not count on this behavior.)

Attributes of <access-log>:

- format: Specifies one or more of several supported variables that result in information being prepended to log entries. Supported variables are $time $request, $ip, $host, $path, $size, $method, $protocol, $user, $status, $referer, $time, $agent, $cookie, $header, and $mime. Between variables, you can type in any separator characters that you want to appear between values in the log message. The default setting is as follows:

  ```
  "$ip - $user - [$time] '$request' $status $size"
  ```

  As an example, this results in log messages such as the following (with the second message wrapping around to a second line):

  ```
  148.87.1.180 - - [06/Nov/2001:10:23:18 -0800] 'GET / HTTP/1.1' 200 2929
  148.87.1.180 - - [06/Nov/2001:10:23:53 -0800] 'GET
  /webservices/statefulTest HTTP/1.1' 200 301
  ```

  In this example, the user is null, the time is in brackets (as specified in the format setting), the request is in single-quotes (as specified), and the status and size in the first message are 200 and 2929, respectively.

- path: Specifies the path and name of the access log. This can be an absolute path or a path relative to the j2ee/home/config directory. The default setting in default-web-site.xml is the following:

  ```
  path="../log/default-web-access.log"
  ```

Note: Note the difference between the path attribute of <access-log>, which specifies a path and file name, and the path attribute of <odl-access-log>, which specifies a path only. (ODL log file names are fixed.)

- split: Specifies how often to begin a new access log. Supported values are "none" (never, which is the default), "hour", "day", "week", or "month". For a value other than "none", logs are named according to the suffix attribute.

- suffix: Specifies timestamp information to append to the base file name of the logs (as specified in the path attribute) if splitting is used, to make a unique name for each file. The format used is that of java.text.SimpleDateFormat, and symbols used in suffix settings are according to the symbology of that class. For information about SimpleDateFormat and the format symbols it uses, refer to the Sun Microsystems Javadoc at the following location:

  http://java.sun.com/products/jdk/1.2/docs/api/index.html

  The default suffix setting is "-yyyy-MM-dd". These characters are case-sensitive, as described in the SimpleDateFormat documentation.

  As an example, assume the following <access-log> element (using the default suffix value):

```
<access-log path="c:\foo\web-site.log" split="day" />
```

Log files are named such as in the following example:

```
c:\foo\web-site-2001-11-17.log
```

<odl-access-log>

Use this element to enable ODL-based access logging for the Web site and to specify information about the access logs, including the path, and maximum values for the size of each file and the total size of all files in the log directory. The log files are where incoming requests (each access of the Web site) are logged.

See "Oracle Diagnostic Logging Versus Text-Based Logging" on page 2-16 for information about ODL.

Note: Do not use both `<access-log>` and `<odl-access-log>`; you can use only one type of logging or the other. (The last element in the Web site XML file would take precedence, but do not count on this behavior.)

Attributes of `<odl-access-log>`:

- `path`: Specifies the path to the access log directory. This can be an absolute path or a path relative to the `j2ee/home/config` directory. For example:

  ```
  path="../log/default-web-access"
  ```

  The initial log file name in this directory is `log1.xml`. As the maximum file size (specified by the `max-file-size` attribute) is reached, subsequent log files are named `log2.xml`, `log3.xml`, and so on.

  Note: Note the difference between the `path` attribute of `<access-log>`, which specifies a path and file name, and the `path` attribute of `<odl-access-log>`, which specifies a path only. ODL log file names are fixed, and cannot be set through the `path` attribute.

- `max-file-size`: Specifies the maximum size of each log file, in kilobytes.
- `max-directory-size`: Specifies the maximum total size, in kilobytes, of all log files in the directory specified in the `path` attribute.

<ssl-config>

This element specifies SSL configuration settings, if applicable. You must use it whenever you set the `secure` attribute of the `<web-site>` element to `"true"`.

See "Servlet Security" on page 2-34 for related information.

Subelement of `<ssl-config>`:

```
<property>
```

Attributes of `<ssl-config>`:

- `keystore`: A relative or absolute path to the keystore database (a binary file) used by this Web site to store certificates and keys for the user base in this installation.

The path value includes the file name. A relative path is relative to the location of the Web site XML file.

A keystore is a `java.security.KeyStore` instance and can be created and maintained using the `keytool` utility, provided with the Sun Microsystems JDK.

- `keystore-password`: The required password to open the keystore.
- `needs-client-auth`: Indicates whether the entity that is a client to OC4J, such as Oracle HTTP Server, must submit a certificate for authorization so it can communicate with OC4J. Supported values are `"true"` for "client authentication" (certificate required), and `"false"`, the default (no certificate required).
- `provider`: You can use this attribute to specify a provider if you are using JSSE (Java Secure Socket Extension). By default, OC4J usually employs the Sun Microsystems implementation of SSL, using an instance of the following for the provider:

  ```
  com.sun.net.ssl.internal.ssl.Provider
  ```

  However, OC4J employs the Oracle SSL implementation in some cases, such as for SOAP and `http_client`.
- `factory`: If you are not using JSSE, use the `factory` attribute to specify an implementation of `SSLServerSocketFactory`. The default setting is:

  ```
  "JSSE: com.evermind.ssl.JSSESSLServerSocketFactory"
  ```

  If you use a third-party `SSLServerSocketFactory` implementation, you can use `<property>` subelements of the `<ssl-config>` element to send parameters to the factory.

<property>

Use `<property>` subelements of the `<ssl-config>` element to pass parameters to a third-party `SSLServerSocketFactory` implementation, if applicable.

Attributes of `<property>`:

- `name`: The name of a parameter to pass to the factory.
- `value`: The value of the specified parameter.

C

Overview of the Session State Tables

This appendix documents the schema for the database tables used by the OC4J database persistence mechanism. See "Configuring Database Replication" on page 7-8 for additional information on this mechanism.

OC4J_HTTP_SESSION

This table stores metadata for a single HTTP session, including identifiers for the application and user setting properties on the session. The `ID` is the primary key.

There is a 1:many relationship between an `OC4J_HTTP_SESSION` table and the `OC4J_HTTP_SESSION_VALUE` tables. Each entry in the `OC4J_HTTP_SESSION` table may have 0 or more entries in the `OC4J_HTTP_SESSION_VALUE` table.

Table C–1 OC4J_HTTP_SESSION Table Description

Name	Null?	Data Type	Description
`ID`	`NOT_NULL`	`VARCHAR2(100)`	The unique session ID.
`APPLICATION_ID`	`NULL`	`VARCHAR2(100)`	The OC4J internal ID assigned to the application the session belongs to.
`IP`	`NULL`	`NUMBER(38)`	The IP address of the machine hosting the application.
`LAST_ACCESSED`	`NULL`	`NUMBER(38)`	The last time the current record was updated.
`USER_NAME`	`NULL`	`VARCHAR2(50)`	The username for the application user setting values on the session.
`MAX_INACTIVE_TIME`	`NULL`	`NUMBER(38)`	The maximum time the session can remain idle before being expired. Session data will not be persisted after this maximum is exceeded.
`CREATION_TIME`	`NULL`	`NUMBER(38)`	The time at which the table was created.

OC4J_HTTP_SESSION_VALUE

This table stores each HTTP session property and the values set on it by the application user. The values are stored as a BLOB (Binary Large OBject). The `ID` and `KEY_FIELD` together comprise the primary key.

Table C–2 OC4J_HTTP_SESSION_VALUETable Description

Name	Null?	Data Type	Description
ID	NOT_NULL	VARCHAR2(100)	The unique session ID.
KEY_FIELD	NOT_NULL	VARCHAR2(100)	The name of a property set by the application user on the session.
VALUE_FIELD	NULL	BLOB	The value of the property set on the session.

OC4J_EJB_SESSION

This table stores the current state of a stateful session bean. The state data is stored as a BLOB (Binary Large OBject). The ID is the primary key.

Table C–3 OC4J_EJB_SESSION Table Description

Name	Null?	Data Type	Description
ID	NOT_NULL	VARCHAR2(100)	The unique session ID.
VALUE_FIELD	NULL	BLOB	The current state data of the session bean.
LOCATION	NULL	NUMBER(38)	The JNDI name that the session bean is bound to.
CHECKSUM	NULL	NUMBER(38)	Used internally to validate that bytes are formatted correctly.
PASSIVATE	NULL	NUMBER(38)	A Boolean value indicating whether the bean has been passivated. If true, the passivated bean will be retrieved from disk.
LAST_ACCESSED	NULL	NUMBER(38)	The last time the current record was updated.
USER_NAME	NULL	VARCHAR2(50)	The username for the application user setting values on the session.
MAX_INACTIVE_TIME	NULL	NUMBER(38)	The maximum time the session can remain idle before being expired. Session data will not be persisted after this maximum is exceeded.
CREATION_TIME	NULL	NUMBER(38)	The time at which the table was created.

Index

Symbols

A

C

D

E

G

H

T

V

W

X