# Oracle® Containers for J2EE

Configuration and Administration Guide

10*g* (10.1.3.5.0)

**E13978-01**

July 2009

Oracle Containers for J2EE Configuration and Administration Guide, 10*g* (10.1.3.5.0)

E13978-01



Primary Author: Joseph Ruzzi

Contributing Authors: Bonnie Vaughan, Dan Hynes, Sheryl Maring, Brian Wright

Contributors: Bryan Atsatt, Ellen Barnes, Angela Barone, Julie Basu, Steve Button, Olivier Caudron, Lars Ewe, Jose Alberto Fernandez, Marcelo Goncalves, Sumathi Gopalakrishnan, Ping Guo, Hal Hildebrand, James Kirsch, Alex Kosowski, Sunil Kunisetty, Clement Lai, Phlippe Le Mouel, Adam Leftik, Mike Lehmann, 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, Frances Zhao, Helen Zhao, Serge Zloto

# Contents

## 4 OC4J Runtime Configuration



## 5 Starting and Stopping OC4J

## 8 Configuring and Managing Clusters and OC4J Groups

## 9 Application Clustering in OC4J



## 10 Task Manager and Thread Pool Configuration



## 11 Logging in OC4J

## Index

# Preface

This book is the primary reference on configuring and managing Oracle Containers for J2EE (OC4J) in both standalone and OPMN-managed (Oracle Application Server) environments. 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
- Documentation Accessibility
- Related Documents
- Conventions

## Intended Audience

This document 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 document is based on the assumption that readers are already familiar with the following topics:

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

### Accessibility of Code Examples in Documentation

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

### Accessibility of Links to External Web Sites in Documentation

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

### Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at `http://www.fcc.gov/cgb/consumerfacts/trs.html`, and a list of phone numbers is available at `http://www.fcc.gov/cgb/dro/trsphonebk.html`.

## Related Documents

For more information, see the following Oracle resources.

Additional OC4J documents:

- *Oracle Containers for J2EE Deployment Guide*

  This document covers information and procedures for deploying an application to an OC4J environment. This includes discussion of the deployment plan editor that comes with Oracle Enterprise Manager 10*g* Application Server Control.

- *Oracle Containers for J2EE Developer's Guide*

  This document discusses items of general interest to developers writing an application to run on OC4J, issues that are not specific to a particular container, such as the servlet, EJB, or JSP container. (An example is class loading.)

- *Oracle Containers for J2EE Servlet Developer's Guide*

  This document provides information for servlet developers regarding use of servlets and the servlet container in OC4J, including basic servlet development and use of JDBC and EJB modules.

- *Oracle Containers for J2EE Support for JavaServer Pages Developer's Guide*

  This document provides information about JavaServer Pages development and the JSP implementation and container in OC4J. This includes discussion of Oracle features such as the command-line translator and OC4J-specific configuration parameters.

- *Oracle Containers for J2EE JSP Tag Libraries and Utilities Reference*

  This document provides conceptual information as well as detailed syntax and usage information for tag libraries, Enterprise JavaBeans (EJB) modules, and other Java utilities provided with OC4J.

- *Oracle Containers for J2EE Services Guide*

This document 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 describes security features and implementations particular to OC4J. It includes information about using JAAS, the Java Authentication and Authorization Service, as well as other Java security technologies.

- *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide*

  This document provides information about the development of Enterprise JavaBeans (EJB) modules and the EJB implementation and container in OC4J.

- *Oracle Containers for J2EE Resource Adapter Administrator's Guide*

  This document provides an overview of J2EE Connector Architecture features and describes how to configure and monitor resource adapters in OC4J.

Oracle Application Server documents:

- *Oracle Application Server Web Services Developer's Guide*

  This document describes development and configuration of Web services in OC4J and Oracle Application Server.

- *Oracle Application Server Advanced Web Services Developer's Guide*

  This document covers topics beyond basic Web service assembly. For example, it describes how to diagnose common interoperability problems, how to enable Web service management features (such as reliability, auditing, and logging), and how to use custom serialization of Java value types.

  This document also describes how to employ the Web Service Invocation Framework (WSIF), the Web Service Provider API, message attachments, and management features (reliability, logging, and auditing). It also describes alternative Web service strategies, such as using JMS as a transport mechanism.

- *Oracle Application Server Web Services Security Guide*

  This document describes Web services security and configuration in OC4J and Oracle Application Server.

## Conventions

The following text conventions are used in this document.

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

# 1

# Introduction to Oracle Containers for J2EE (OC4J)

This chapter provides a general introduction to Oracle Containers for J2EE (OC4J), in the following sections:



## Overview of OC4J

Oracle Containers for J2EE 10*g* (10.1.3.5.0), 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 started, managed, and stopped directly as a self-contained component.

  See "OC4J in a Standalone Configuration" on page 1-6 for details on this configuration.

- A **managed** configuration, in which OC4J is installed as part of a group of OC4J instances and managed as a component of Oracle Application Server.

  A **group** is a synchronized set of OC4J instances that belong to the same **cluster topology**, which comprises two or more loosely connected Oracle Application Server nodes. Configuration, administration, and deployment operations can be performed simultaneously on all OC4J instances in the group.

  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 typically also include at least one Oracle HTTP Server (OHS) instance, which provides Web communication and load balancing functionality.

  See "OC4J in an Oracle Application Server Configuration" on page 1-7 for details.

OC4J is written entirely in Java and executes on the Java Virtual Machine (JVM) of Java Platform, Standard Edition (Java SE) Development Kit (JDK) 6, Java Platform 2, Standard Edition (J2SE) Development Kit (JDK) 5.0 (also known as JDK 1.5), or JDK 1.4.2. For OPMN-managed OC4J, JDK 5.0 is installed with the server binaries and used by default to start the OC4J instance. For standalone OC4J, you must provide the JDK. You can configure an OC4J instance to run on multiple JVMs.

The OC4J documentation is based on the assumption 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.

# J2EE Support in OC4J

OC4J 10*g* (10.1.3.5.0) supports the standard J2EE specifications listed in Table 1–1.

***Table 1–1 Supported J2EE Specifications***

| J2EE Specification | Version Supported By OC4J |
|---|---|
| JavaServer Pages (JSP) | 2.0 |
| Servlets | 2.4 |
| Enterprise JavaBeans (EJB) | 2.1, 3.0 (Complete EJB 3.0 and JPA implementation) |
| Java Management Extensions (JMX) | 1.2 |
| J2EE Management | 1.0 |
| J2EE Application Deployment | 1.1 |
| 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) | 3.0 |
| Java Authentication and Authorization Service (JAAS) Provider | 1.0 |
| J2EE Connector Architecture | 1.5 |
| Enterprise Web Services | 1.1 |
| 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) | 1.2 |
| Java API for XML Registries (JAXR) | 1.0.5 |

# New and Changed Features in OC4J

The following topics outline new features in Oracle Containers for J2EE 10*g* (10.1.3.*x*) as well as functional changes from previous releases.

## New Features in OC4J

Oracle Containers for J2EE 10*g* (10.1.3.*x*) includes a number of new features and enhancements, as described in the following topics:

### 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.

- Support for the Enterprise Web Services 1.1 specification
- EJB 2.1 Web services endpoint 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 *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 or edit a deployment plan containing the OC4J-specific configuration data needed to deploy a component to OC4J.
- 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.

### Support for Enterprise JavaBeans 3.0

OC4J 10*g* (10.1.3.5.0) provides complete support for the Enterprise JavaBeans 3.0 final specification, including support for EJB annotations and dependency injections. The final specification is available at the following Web site:

```
http://java.sun.com/products/ejb/
```

---

**Note:** OC4J must use either JDK 6 or 5.0 to enable EJB 3.0 support. JDK 5.0 is included with the 10*g* (10.1.3.5.0) release, in which OPMN-managed OC4J instances use JDK 5.0 by default.

---

#### 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. Oracle TopLink includes support for the OC4J Container Managed Persistence (CMP) container and base classes that simplify Bean Managed Persistence (BMP) development.

#### Oracle Job Scheduler

The Oracle Job Scheduler provides asynchronous scheduling services for J2EE applications. Its key features include capabilities for submitting, controlling, and monitoring *jobs*, each job 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 types of XA resources, including databases from Oracle as well as other vendors and JMS providers, such as IBM WebSphere MQ. 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 Oracle Enterprise Messaging Service (OEMS), which ships with OC4J 10*g* (10.1.3.5.0), 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.

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.

#### New admin_client.jar Commands and Remote Client

The `admin_client.jar` utility has new commands for managing data sources and the OC4J JMS connection factories and destinations. The admin_client.jar commands are also available in a remote Administrative Client Utility. You can use these commands through the command-line tool or through the relevant JMX MBeans to add, remove, and get information about data sources and JMS connection factories and destinations. For details, see Chapter 6, "Using the admin_client.jar Utility".

### Configuration File Changes from Previous Releases

The following changes have been made to configuration files utilized in standalone OC4J and in OC4J instances installed as components of Oracle Application Server. All of the files noted are installed by default in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config`, in which *`instance`* represents the OC4J instance name.

#### application.xml

- The `<persistence>` element has been moved to the new `system-application.xml` file.
- The `<jazn>` element now points to the new `system-jazn-data.xml` file as the security configuration file for the OC4J instance. For more information about `<jazn>`, see the *Oracle Containers for J2EE Security Guide*.
- The `default-data-source` attribute of the root `<orion-application>` element now specifies `jdbc/OracleDS` as the default data source in both standalone OC4J and Oracle Application Server.
- The `<ejb-module>` element for PortComponentLinkResolver has been removed.
- The `<odl>` element, used to enable ODL logging for the `default` application, has been added but commented out as a subelement of `<log>`.

#### ascontrol-web-site.xml

- This file has been removed from both standalone OC4J and Oracle Application Server. The Application Server Control instance deployed to OC4J is now bound to `default-web-site.xml` by default and is accessible through the `/em` context root.

#### default-web-site.xml

- This file configures the default Web site used in both standalone OC4J and Oracle Application Server. All applications, including the Application Server Control instance deployed to the OC4J instance, are accessed by default through the `default` Web site using the context root specified in this file.

#### global-web-application.xml

- The `<dtd>` element has been removed from the Oracle Application Server version of this file.
- The `<url-pattern>` element in the rmi-tunnel servlet definition specifies `rmiTunnel/*` in both standalone OC4J and Oracle Application Server.

#### http-web-site.xml

- This file has been removed from both standalone OC4J and Oracle Application Server. All applications deployed to the OC4J instance are now bound to `default-web-site.xml` by default.

#### j2ee-logging.xml

- This new file is used to configure Java loggers, including the `oracle` logger.

#### jazn-data.xml

- This file no longer contains the security configuration for the OC4J instance. This configuration is now defined in the new `system-jazn-data.xml` file. The `jazn-data.xml` file can be specified, however, at the application level to define users and roles. For more information about the `jazn-data.xml` and `system-jazn-data.xml` files, see the *Oracle Containers for J2EE Security Guide*.

#### oc4j-connectors.xml

- The `location` attribute of the `<connector>` element is no longer specified for the data sources and Oracle Enterprise Messaging Service (OEMS) connectors.

### server.xml

- The <web-site> elements pointing to `http-web-site.xml` and `ascontrol-web-site.xml` have been removed. A single element now points to `default-web-site.xml`, the configuration file for the `default` Web site.
- Multiple <shared-library> elements have been added, each referencing a shared library installed with OC4J.
- A <thread-pool> element has been added to the `server.xml` for defining thread pools for use by OC4J processes and applications deployed to OC4J instances. This element replaces the <global-thread-pool> and <work-manager-thread-pool> elements, which are deprecated in OC4J 10*g* (10.1.3.5.0)
- A <custom-thread-pool> element has been added to the `server.xml` file for defining separate, custom thread pools for applications.

### system-application.xml

- This is a new file, added to provide configuration for the `system` application. See "The system Application" on page 1-10 for more information on this new internal component.

### system-jazn-data.xml

- This new file contains the security configuration for the OC4J instance. It essentially replaces `jazn-data.xml`. For more information about the `jazn-data.xml` and `system-jazn-data.xml` files, see the *Oracle Containers for J2EE Security Guide*.

## OC4J in a Standalone Configuration

The standalone, or *unmanaged*, OC4J configuration offers robust, J2EE-compliant containers that are easy to administer. In this configuration, a single OC4J instance is installed into a single *ORACLE_HOME* directory, the root directory in which Oracle software is installed. The standalone OC4J configuration includes the following components:

- Oracle Containers for J2EE 10*g* (10.1.3.5.0)
- Oracle Enterprise Manager 10*g* Application Server Control, a Web-based administration application installed by default with OC4J

Application Server Control is enabled immediately upon installation. See "Oracle Enterprise Manager 10g Application Server Control" on page 3-1 for details on using this management interface.

### Installation

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

### Administration

Standalone OC4J is administered as a standalone OC4J instance, using the Application Server Control application installed with the instance, one of the built-in command-line utilities, such as `admin_client.jar`, or OC4J Ant tasks. For more information about these tools, see Chapter 3, "Tools for Administering OC4J."

You can also administer standalone OC4J remotely with the Administrative Client Utility, which includes OC4J Ant task and `admin_client.jar`. For more information about remote administration, see "Downloading and Extracting the Remote Administration Client" on page 6-5.

The `admin.jar` tool provided with OC4J can perform administration tasks only on a standalone OC4J server. For information about using this tool, see Example 7, "Using the admin.jar Utility."

### Starting, Stopping, and Restarting

In a standalone configuration, an OC4J instance is started using an `oc4j` command script or the executable `oc4j.jar` archive. Startup options and system properties are set before startup for the command script or at startup with the `oc4j.jar` direct execution model.

See "Starting OC4J in a Standalone Environment" on page 5-1 for details.

You can stop and restart a standalone OC4J server with the `admin_client.jar` or `admin.jar` command-line utility or an `oc4j` command script. For details, see "Stopping OC4J in a Standalone Environment" on page 5-3, "Restarting an OC4J Instance in a Standalone Environment" on page 5-5, or "Stopping and Restarting OC4J in a Standalone Environment" on page 7-3.

### Backup, Restore, and Disaster Recovery Capabilities

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

### Web Communications

Web communications 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.

The default Web site is defined in the `default-web-site.xml` file, which specifies the default HTTP listener on port `8888`. You can define additional Web sites using variations of this file. See Chapter 13, "Managing Web Sites in OC4J" for instructions on creating additional Web sites in OC4J.

## OC4J in an Oracle Application Server Configuration

In this configuration, OC4J is installed as a component of Oracle Application Server, in a group of one or more OC4J instances within an Oracle Application Server cluster. A typical configuration includes the following components:

- Oracle Containers for J2EE 10*g* (10.1.3.5.0), one or more instances in one or more groups
- Oracle Enterprise Manager 10*g* Application Server Control, a Web-based administration application installed by default with OC4J
- Oracle HTTP Server 1.3, which provides front-end Web communication and load-balancing functionality
- Oracle Process Manager and Notification Server (OPMN), used to start, stop, and monitor the other installed components, including OC4J and Oracle HTTP Server. OMPN includes Oracle Notification Server (ONS), which manages communications between components.

Oracle Application Server provides support for HTTP session and stateful session Enterprise JavaBeans (EJB) replication and load balancing across a group of OC4J instances within a cluster topology. See Chapter 9, "Application Clustering in OC4J" for details.

The connectivity provided within an Oracle Application Server cluster is a function of Oracle Notification Server (ONS), which manages communications between Oracle Application Server components, including OC4J and Oracle HTTP Server. The ONS server is a component of Oracle Process Manager and Notification Server (OPMN), which is installed by default on every Oracle Application Server host.

The Oracle Universal Installer provides a number of installation options:

- **Integrated Web Server, J2EE Server, and Process Management**

  In this configuration, all components are installed into a single *ORACLE_HOME* directory, including OC4J, Oracle HTTP Server, and OPMN.

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

- **J2EE Server and Process Management**

  This installation includes OC4J and OPMN. It can be utilized as a *standalone* OPMN-managed OC4J instance for development or testing purposes, or can be included within an Oracle Application Server cluster.

- **Web Server and Process Management**

  This installation includes only Oracle HTTP Server and OPMN. It can be used as a *standalone* Oracle HTTP Server instance, typically serving as the front-end Web listener for an Oracle Application Server cluster.

### Installation

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

### Administration

Administration tasks can be performed using any of these tools:

- The Web-based Application Server Control user interface
- The `admin_client.jar` command-line tool
- OC4J Ant tasks
- The `admin.jar` command-line tool, only for standalone OC4J servers

For more information about these tools, see Chapter 3, "Tools for Administering OC4J."

In an Oracle Application Server clustered environment, you can use a single Application Server Control instance to manage all OC4J instances in a cluster. For more information about this application, see "Oracle Enterprise Manager 10g Application Server Control" on page 3-1 for details on this application.

OC4J includes a set of Ant tasks for performing administration tasks on a group of OC4J instances within an Oracle Application Server cluster, on an OPMN-managed OC4J instance, or on a standalone OC4J server. For details about the Ant tasks and guidelines for integrating the tasks into your application build process, see "Deploying with the OC4J Ant Tasks" in the *Oracle Containers for J2EE Deployment Guide*.

The `admin_client.jar` tool provided with OC4J can perform administration tasks on a group of OC4J instances within an Oracle Application Server cluster or on an

OC4J instance. Also, the Administrative Client Utility distribution, `oc4j_admin_client_101350.zip`, contains the client-side jars necessary for performing administrative operations from a remote client in three ways:

- Using `admin_client.jar` commands remotely against an OPMN-managed or standalone OC4J instance
- Using OC4J Ant tasks remotely against an OPMN-managed or standalone OC4J instance
- Using a JMX programmatic client to manage OC4J remotely

For more information about remote administration, see "Downloading and Extracting the Remote Administration Client" on page 6-5.

### Starting and Stopping

In a managed environment, you must use OPMN to start and stop all components, including OC4J. See "Starting OC4J in an Oracle Application Server Environment" on page 5-2 for details.

OC4J runtime options and system properties can be manually set in the OPMN configuration file, `opmn.xml`. See Chapter 4, "OC4J Runtime Configuration" for details.

### Backup, Restore, and Disaster Recovery Capabilities

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

### Web Communications

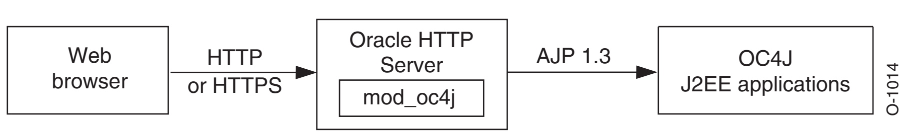
A standalone OPMN-managed OC4J instance (the J2EE Server and Process Manager install type) can use the built-in OC4J Web server to directly receive and respond to HTTP[S] requests.

Web communications with OC4J can also be managed through Oracle HTTP Server, which serves as a front-end listener, and the `mod_oc4j` module, which forwards HTTP requests to OC4J instances using the Apache JServ Protocol (AJP) 1.3.

The request and response flow between Oracle HTTP Server and OC4J is as follows:

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

***Figure 1–1 OC4J Web Communications Through OHS***




Mount points that map request URLs to OC4J instances serving the requesting applications are dynamically created in `mod_oc4j` at the time the applications are deployed. Requests that come in for a specific mount point are routed to the OC4J instance corresponding to that mount point.

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

# Overview of the Application Hierarchy in OC4J

This section provides an overview of the application hierarchy within an OC4J instance.

## The system Application

The `system` application is an internal component of Oracle Containers for J2EE 10*g* (10.1.3.5.0). This application is automatically deployed to an OC4J instance or standalone OC4J the first time it starts.

The application was added primarily to address issues related to deploying or redeploying applications to OC4J. It sits at the root of the application hierarchy, and provides classes and configuration required at OC4J startup. For example, it provides the shared libraries imported by default by all other deployed applications, such as the Oracle JDBC driver and XML parser implementations.

The `system` application is an OC4J internal component only. Applications cannot be deployed to it, nor can it be declared the parent of another application. The `default` application continues to serve as the default parent of all deployed applications.

The configuration for the `system` application is defined in `system-application.xml`, which is installed in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` by default. The default OC4J *`instance`* created at installation is called `home`.

---

**Important:** Because `system` is a key internal component that is critical to OC4J startup, the `system-application.xml` file should *not* be modified except for the `<jazn>` and `<log>` tags.

You can modify the `<jazn>` tag as needed to specify changes to the security provider, the location of the OC4J security configuration file (`system-jazn-data.xml`), or both. For more information about `<jazn>` and the `system-jazn-data.xml` file, see the *Oracle Containers for J2EE Security Guide*.

You can modify the <log> tag to rotate the system log file.

---

## The default Application

The `default` application sits just below `system` in the application hierarchy. It continues to serve as the default parent of all other J2EE applications deployed into the OC4J instance. As such, all configuration parameters defined for the `default` application are inherited by all other applications, unless explicitly overridden at the application level.

Standalone Web modules (WAR files) may also be deployed to the `default` application.

The configuration for the default application is defined in `application.xml`, which is installed in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` by default. The default OC4J *`instance`* created at installation is called `home`.

## The Global Web Application

The global Web application is the Web module component of the `default` application. It provides configuration data applied by default to all Web modules deployed to the OC4J instance. It also contains initialization parameters applied by default to all servlets.

The configuration file for the default Web application is `global-web-application.xml`, which is installed in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` by default. This file contains parameters that apply by default to all Web modules deployed to the OC4J instance, as well as servlet initialization parameters that apply to all servlets. You can override any of these parameter values with corresponding values in a Web module's `orion-web.xml` file.

In a standalone OC4J installation, the root directory of the default Web application is `j2ee/home/default-web-app/`. To deploy to the default Web application, you can place your JSP pages and class files under this directory in the standard directory structure for a Web application.

## J2EE Applications

By default, an application deployed to an OC4J instance inherits configuration parameters from its designated parent application, or from the `default` application if no other parent is specified. However, a parameter value set in an application's `orion-application.xml` descriptor overrides an equivalent parameter inherited from the parent.

A Web module must be contained within a parent J2EE application. A WAR file is typically packaged and deployed with the EAR file that defines the parent J2EE application. However, a WAR file can be deployed to the `default` application as a standalone Web module.

# 2

# Installing Standalone OC4J

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

For instructions on installing OC4J as a component of Oracle Application Server, see the environment-specific *Oracle Application Server Installation Guide*.

The following topics are covered in this chapter:

- Meeting Installation Prerequisites for a Standalone OC4J Server
- Installing the Standalone OC4J Distribution

## Meeting Installation Prerequisites for a Standalone OC4J Server

Ensure that the following prerequisites are met before installing a standalone OC4J server.

### Install JDK 6, 5.0, or 1.4.2

Before installing standalone OC4J, you must install one of the following JDK releases on the OC4J host machine:

- Java Platform, Standard Edition (Java SE) Development Kit (JDK) 6
- Java Platform 2, Standard Edition (J2SE) Development Kit (JDK) 5.0 (also known as JDK 1.5)
- Java Platform 2, Standard Edition (J2SE) Development Kit (JDK) 1.4

You can download the JDK release from `http://java.sun.com/j2se/`.

---

**Note:** For standalone OC4J, you must provide the JDK. For OPMN-managed OC4J, JDK 5.0 is packaged with the server binaries.

---

### Set Environment Variables

After installing J2SE, ensure that the `JAVA_HOME` and `ORACLE_HOME` environment variables are set. You can also set the `J2EE_HOME` environment variable.

***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. For example:<br>JAVA_HOME=/java/j2se15<br>The JDK that will be used must be added to the host machine's PATH environment variable. |
| 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 or oc4j.cmd executable script.<br>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 *ORACLE_HOME*/j2ee/home, which will be the installed location of admin_client.jar, oc4j.jar, and admin.jar. The value of *ORACLE_HOME* is the root directory into which you will install the OC4J distribution.<br>Setting the J2EE_HOME environment variable or the oracle.j2ee.home system property to the J2EE home directory enables you to invoke admin_client.jar, oc4j.jar, or admin.jar from any directory. |

Instead of the environment variables ORACLE_HOME and J2EE_HOME, you can use the system properties oracle.home and oracle.j2ee.home to set the Oracle and J2EE home directories.

If you want to use a locale other than the default locale for the operating system, also set the LC_ALL and LANG environment variables, both to the same value.

# 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 document as *ORACLE_HOME*, using the archive utility of your choice. The installer automatically creates the required directory structure for you, as follows:

```
ORACLE_HOME
   /ant
   /bin
   /diagnostics
   /j2ee
   /javacache
   /javavm
   /jdbc
   /jlib
   /lib
   /opmn
   /rdbms
   /sqlj
   /toplink
   /webservices
   /xqs
```

---

**Note:** The file permissions are not set correctly after the jar utility has been used to unzip standalone OC4J. To fix the file permissions, you can either extract the distribution again using the unzip utility or set the permissions on the executables to allow them to be run.

---

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

You can also activate the oc4jadmin account before starting OC4J, using the jazn.jar tool. This tool is located in the *ORACLE_HOME* /j2ee/home directory. The syntax is as follows:

```
jazn.jar -activateadmin password
```

---

**Note:** The oc4j.jar -install command, previously used to activate the oc4jadmin account as well as set the password for this account, is deprecated in OC4J 10*g* (10.1.3.5.0).

---

The standalone OC4J distribution is installed with a default configuration that includes a default Web site where you can access applications and a Web site that enables you to use Application Server Control. These are provided so that you can start using OC4J immediately. See Chapter 13, "Managing Web Sites in OC4J" for additional information.

# 3

# Tools for Administering OC4J

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

- Oracle Enterprise Manager 10g Application Server Control
- The admin_client.jar Command Line Utility
- The admin.jar Command Line Utility
- The oc4j Executable Scripts
- Oracle Process Manager and Notification Server (OPMN)

## Oracle Enterprise Manager 10g Application Server Control

**Oracle Enterprise Manager 10g Application Server Control** is a JMX-compliant, Web-based user interface for deploying, configuring, and monitoring applications within OC4J, as well as for managing a standalone OC4J server, a group of OC4J instances within an Oracle Application Server cluster, and the Web services used by your applications. This section covers the following topics:

- Accessing Application Server Control in Standalone OC4J
- Accessing Application Server Control in Oracle Application Server
- Functional Overview of the Application Server Control Interface

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

> **Note:** The current release of Application Server Control supports some configuration of OPMN and starting and stopping Oracle HTTP Server, but not the Oracle HTTP Server configuration. For instructions on configuring OPMN and Oracle HTTP Server, see the *Oracle Process Manager and Notification Server Administrator's Guide*.

### Accessing Application Server Control in Standalone OC4J

Application Server Control is installed and configured automatically when you install the OC4J software. It is started by default when OC4J is started.

The console is accessed through the `default` Web site, which is configured to listen for HTTP requests on port `8888`. To access the console, simply type the following URL in a Web browser:

```
http://hostname:8888/em
```

## Accessing Application Server Control in Oracle Application Server

Application Server Control is installed and configured as an embedded component of OC4J. The console is started with all other installed Oracle Application Server components, using the OPMN command-line tool, `opmnctl`, which is installed in the *ORACLE_HOME*/`opmn/bin` directory on each server node.

You can start all installed components by issuing the following command:

```
cd ORACLE_HOME/opmn/bin
opmnctl startall
```

For a cluster topology that includes multiple OC4J instances, if the OPMN configuration file for the cluster, `opmn.xml`, does not include the `sequential` option, you should use the `-sequential` flag in the command:

```
cd ORACLE_HOME/opmn/bin
opmnctl startall -sequential
```

The `sequential` option causes the OC4J instances to start sequentially. If you started the components in parallel, resource contention issues might occur. For information about how to specify the `sequential` option in the `opmn.xml` file, see "Setting Runtime Options in a Managed OC4J Configuration" on page 4-2.

In a typical Oracle Application Server installation, all Web applications, including Application Server Control, are accessed through Oracle HTTP Server. Use the following URL to access the console:

```
http://ohs_host_address:port/em
```

- *ohs_host_address* is the address of the Oracle HTTP Server host machine; for example, `server07.company.com`
- *port* is an HTTP listener port assigned to Oracle HTTP Server by OPMN. Run the following `opmnctl` command on the Oracle HTTP Server host machine to get the list of assigned listener ports from OPMN:

  ```
  opmnctl status -l
  ```

  Supply the port designated as `http1` in the OPMN status output as the value for *port*:

  ```
  HTTP_Server | HTTP_Server | 6412 | Alive | 1970872013 | 1
  6396 | 0:48:01 | https1:4443,http2:722,http1:7779
  ```

## Functional Overview of the Application Server Control Interface

Application Server Control is organized into several functional areas, described in the following text.

### Applications

- Start or stop applications, modules, or standalone resource adapters deployed into an OC4J instance or group of instances within an Oracle Application Server cluster
- 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 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 EJB modules

#### Web Services

- Enable or disable a Web service
- View metrics and statistics for Web services running within an 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 an OC4J instance
- View logs for the default application (which includes the global Web application) and Application Server Control
- 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 11, "Logging in OC4J" for more on the logging capabilities provided by OC4J.

## The admin_client.jar Command Line Utility

OC4J provides a command-line utility— `admin_client.jar`—that can be used to perform operations on active OC4J instances in an Oracle Application Server clustered environment as well as on standalone OC4J servers.

Among the tasks you can perform with this utility:

- Deploy an application (EAR), a standalone Web module (WAR), a standalone EJB module (EJB JAR), or a standalone resource adapter (RAR) to a specific OC4J instance or to a group of instances within an Oracle Application Server cluster
- Undeploy an application, Web module, EJB module, or resource adapter
- Incrementally update a deployed EJB module with modified classes

- Create a new shared library
- Create JDBC and JMS resources
- Stop, start, or restart an OC4J instance
- Stop, start, or restart a specific application, on a specific OC4J instance or on a group of instances cluster wide

See Chapter 6, "Using the admin_client.jar Utility" for instructions on using this tool.

## 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 standalone OC4J instance.

---

**Note:** The `admin.jar` utility can be used only to manage a single OC4J instance in a standalone OC4J installation.

Due to its more advanced capabilities, the `admin_client.jar` utility should be used instead of `admin.jar`. See Chapter 6, "Using the admin_client.jar Utility" for details on using this utility.

---

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 to a standalone OC4J instance
- Add, remove, or test a global or application-specific data source

The utility is installed by default in `ORACLE_HOME/j2ee/instance/`. OC4J must be started before this utility can be used, except when you upgrade data sources. Also, the utility cannot be used to start OC4J. See Chapter 7, "Using the admin.jar Utility" for instructions on using this tool.

## The oc4j Executable Scripts

The OC4J distribution includes executable scripts that can be used in a standalone OC4J configuration to start and stop a local OC4J instance, get the OC4J version, and complete the OC4J installation process.These scripts include a shell script for Linux and UNIX environments and a batch file for Windows environments.

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

- Use the `oc4j` shell script in a Linux or UNIX environment.
- Use the `oc4j.cmd` batch file in a Windows environment.

Before you use one of these scripts, the `ORACLE_HOME` and `JAVA_HOME` environment variables must be set, as described in "Set Environment Variables" on page 2-1.

Both executables use the same syntax, which follows:

```
oc4j [options]
```

The set of options that can be passed to the executables is identical for both, as summarized in Table 3–1.

***Table 3–1 Options for oc4j executables***

| Option | Description |
|---|---|
| `-start` | Starts the OC4J instance. |
| `-shutdown -port` *`ormiport`* `-password` *`password`* | Stops the OC4J instance.<br>`-port` *`ormiport`*:<br>You do not need to specify the port if OC4J is running on the default ORMI port, which is `23791`.<br>`-password` *`password`*:<br>Specify the `oc4jadmin` account password. |
| `-version` | Returns the OC4J version number. |
| `-help` | Displays the syntax and set of options. |

## 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 Oracle HTTP Server. As a result, OPMN must be installed into each *ORACLE_HOME* directory 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 control the OPMN daemon. 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 supports some configuration of OPMN and starting and stopping Oracle HTTP Server, but not the Oracle HTTP Server configuration. For instructions on configuring OPMN and Oracle HTTP Server, see the *Oracle Process Manager and Notification Server Administrator's Guide*

OPMN is configured through the `opmn.xml` configuration file, which is located in the *ORACLE_HOME*`/opmn/conf` directory. Most edits to this file must be made manually because the current release of Application Server Control does not provide a file-editing capability.

The following example shows how OC4J configuration data is structured in the `opmn.xml` configuration file:

- Configuration data for each component is set in an `<ias-component>` element, in which the `id` attribute equals the component name, in this case `default_group`.
- 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 following element is an abridged example of the OC4J configuration data structure in `opmn.xml`:

```
<opmn>
 . . .
 <ias-component id="default_group">
  <process-type id="home" module-id="OC4J" status="enabled">
    <module-data>
      <category id="start-parameters">
        <data id="java-options" value=" -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"/>
    <port id="rmis" range="12701-12800"/>
    <process-set id="default_group" numprocs="1"/>
  </process-type>
 </ias-component>
</opmn>
```

## Changing the oc4jadmin Account Password

The OC4J administrator account is created by default with the user name `oc4jadmin`. This account is required to invoke commands using the various tools provided with OC4J, such as the `admin_client.jar` command-line utility, and can also be used to log in to Application Server Control.

The `oc4jadmin` account is assigned the `oc4j-administrators` role, which an account must have to manage users and roles. An account must also have this role to connect to the MBeanServer server.

The initial password for this account can be set when OC4J is installed; otherwise, you will be prompted to set it the first time OC4J is started. All OC4J instances in a group within an Oracle Application Server cluster need to have the same password for the `oc4jadmin` account so that you can access all of the instances through Application Server Control and perform group operations. Also, all OC4J instances in an Oracle Application Server cluster must have the same password for the `oc4jadmin` account to prevent problems with OPMN.

The password can later be changed, as described in Appendix A of the *Oracle Application Server Administrator's Guide*. The following guidelines apply to changing the `oc4jadmin` password:

- As a best practice, Oracle suggests that you use the `oc4jadmin` account only for the initial login to Application Server Control. After that, you should create a new account (and accounts for your fellow administrators) to use for your everyday work. The `oc4jadmin` account and its password should be used only internally by the `ascontrol` application, which uses that account to log into and manage the other OC4J instances in the Oracle Application Server cluster.

- If you change the `oc4jadmin` password, you must change it for all OC4J instances in the cluster. Changing this password involves quite a few steps, which are documented in Appendix A of the *Oracle Application Server Administrator's Guide*. Specifically, the procedure to change the `oc4jadmin` password for the Administration OC4J instance, which runs the active `ascontrol` application, is different from the procedure to change the `oc4jadmin` password for the remotely managed OC4J instances.

# 4

# OC4J 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
- Enabling Remote Debugging from an Integrated Development Environment

## Specifying the JDK Version

OC4J requires one of the following JDK releases:

- Java Platform, Standard Edition (Java SE) Development Kit (JDK) 6
- Java Platform 2, Standard Edition (J2SE) Development Kit (JDK) 5.0 (also known as JDK 1.5)
- Java Platform 2, Standard Edition (J2SE) Development Kit (JDK) 1.4

You must specify the JDK version to use for a standalone OC4J configuration. You can specify the JDK version for each OC4J instance, which uses JDK 5.0 by default 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 want OC4J to use. The JDK that will be used must be added to the host machine's `PATH` environment variable.

Alternatively, you can specify the JDK to use at OC4J startup. For example:

```
C:\ORACLE_HOME\j2ee\home\C:\jdk\bin\java -jar oc4j.jar
```

### Specifying the JDK in a Managed Configuration

An OPMN-managed OC4J instance installed as a component of Oracle Application Server will use JDK 5.0 by default. JDK 5.0 or JDK 6 is required to utilize EJB 3.0. If applications that will be deployed to OC4J require a JDK 6 or JDK 1.4.2 release, you need to switch to the other version.

Before switching from JDK 5.0 to JDK 6 or JDK 1.4.2, you must remove all compiled application files from the OC4J instance:

1. Stop the OC4J instance.

2. Delete the *ORACLE_HOME*/j2ee/*instance*/application-deployments directory.

   Deleting this directory will cause the application files to be recompiled when OC4J is restarted with JDK 6 or JDK 1.4.2.

You can specify the JDK to use for each OC4J instance through manual edits to the opmn.xml configuration file. If you want to use the javac compiler installed with the JDK defined in the JAVA_HOME environment variable, also remove the <java-compiler> element from the server.xml file and let OC4J rediscover the default settings.

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="default_group"> element in the XML structure. For example:

```
<ias-component id="default_group">
  <process-type id="home" module-id="OC4J" status="enabled">
    <module-data>
      <category id="start-parameters">
        <data id="java-bin" value="c:/myhost/jdk/bin/java"/>
      </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
- Setting Runtime Options in a Managed OC4J Configuration
- Overview of OC4J Runtime Options

## 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]
```

Runtime options ([args]) are specified after oc4j.jar in the syntax. For example:

```
java -jar oc4j.jar -userthreads
```

## 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 OC4J runtime options 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="default_group">` element in the XML structure. For example:

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

## Overview of OC4J Runtime Options

Table 4–1 describes the OC4J runtime options.

***Table 4–1 OC4J Startup Options***

| Command-Line Argument | Description |
|---|---|
| `-quiet` | Suppresses standard output to the console. |
| `-config` *`path`* | Specifies the path to the `server.xml` descriptor file. The default location is the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` directory. |
| `-validateXML` | Validates XML configuration files defined by an XSD at the time they are read.<br><br>If you add the `tx-retry-wait` attribute to the `<entity-deployment>` or `<session-deployment>` element in your orion-ejb-jar.xml file, do not use the `-validateXML` option on the OC4J startup command line. |
| `-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.<br><br>See "Managing the stdout and stderr Log Files" on page 4-6 for additional system properties that can be set to manage `stdout` files.<br><br>In an OPMN-managed configuration, the file will be generated within an *instance*_default_group_1 directory appended to the path specified.<br><br>For example, suppose you specify the following element in `opmn.xml`:<br><br>`<data id="oc4j-options" value="...-out /mypath/mylog.log"/>`<br><br>The `mylog.log` file will actually be generated in this file:<br><br>`/mypath/`*`instance`*`_default_group_1/mylog.log` |
| `-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.<br><br>See "Managing the stdout and stderr Log Files" on page 4-6 for additional system properties that can be set to manage `stderr` files.<br><br>Note that in an OPMN-managed configuration, the file will be generated within an *instance*_default_group_1 directory appended to the path specified. See the `-out` description for details. |

*Table 4–1 (Cont.) OC4J Startup Options*

| Command-Line Argument | Description |
|---|---|
| `-verbosity` *int* | Define an integer between `1` and `10` to set the verbosity level of the message output. A value of `10` will produce the most verbose output. For example: `java -jar oc4j.jar -verbosity 10` |
| `-monitorResourceThreads` | Enables backup debugging of thread resources. Enable only if you have problems with threads getting stuck in critical sections of code. |
| `-userThreads` | Enables context lookup support from user-created threads. |
| `-http.sessionInvalidatingThreads` | Specifies the maximum number of threads to invalidate HTTP sessions. The default value is `3`. |
| `-listProperties` | Outputs a list of all of the OC4J-specific system properties that can be set on the JVM at OC4J startup, then exits. The following example will redirect the output to a text file in the working directory: `java -jar oc4j.jar -listProperties > props.txt` |
| `-sequential` | Starts each OC4J instance within an Oracle Application Server cluster sequentially. |
| `-version` | Returns the installed version of OC4J 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, as the following topics describe:

- Setting System Properties in a Standalone OC4J Configuration
- Setting System Properties in an OPMN-Managed OC4J Configuration
- Allowing Different Dependencies on a Shared Library with manifest.dependencies.warn.only
- Preventing the Use of Symbolic Links
- Managing the stdout and stderr Log Files
- Overview of General System Properties
- Overview of Debug Properties

---

**Note:** You can output a list of all of the OC4J-specific system properties that can be set on the JVM at OC4J startup using the `oc4j.jar -listProperties` option. The following example will redirect the output to a text file in the working directory:

```
java -jar oc4j.jar -listProperties > props.txt
```

---

## 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 follows:

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

All system properties (*props*) are specified before `oc4j.jar` in the syntax. Each system property must be prefaced on the command line with a `-D`. For example:

```
java -Doc4j.formauth.redirect=true -jar oc4j.jar
```

## Setting System Properties in an OPMN-Managed OC4J Configuration

When OC4J is installed as a component of Oracle Application Server, you can add OC4J system properties manually to the `opmn.xml` configuration file. Options will be passed to a managed OC4J instance 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="default_group">` element in the XML structure. Preface all system properties with a `-D`. For example:

```
<ias-component id="default_group">
  <process-type id="home" module-id="OC4J" status="enabled">
    <module-data>
      <category id="start-parameters">
        <data id="java-options" value="-Doc4j.formauth.redirect=true
         -Dhttp.session.debug=true"/>
      </category>
      ...
    </module-data>
  </process-type>
</ias-component>
```

## Allowing Different Dependencies on a Shared Library with manifest.dependencies.warn.only

When OC4J initializes, it validates the dependencies of the shared libraries defined in the container-level configurations by verifying the versions specified in the MANIFEST.MF file. The default behavior for a dependency check failure is to issue an error and stop the initialization process. If you want to use different versions of the same shared library, you can set the `mainifest.dependencies.warn.only` property to `true` at startup to have OC4J issue a warning and continue initializing when a dependency check fails.

This property is useful if you have a methodology of performing simultaneous project releases and builds that have different dependencies on the same application library.

When `mainifest.dependencies.warn.only` is set to `true`, OC4J issues a warning message by default. The logged message goes to the locations specified in *ORACLE_HOME*`j2ee/`*instance*`/config/j2ee-logging.xml` configuration file for the `oracle` logger. This logger has two output locations, the console and an ODL-formatted XML file in `j2ee/`*instance*`/log/oc4j/log.xml`. In standalone OC4J, the *instance* name is `home`. In an OPMN-managed, Oracle Application Server environment, all console output gets redirected to the *ORACLE_HOME*`/opmn/logs/`*instance*_default_group_1 file (or some variant of this name).

## Preventing the Use of Symbolic Links

By default, OC4J10*g* (10.1.3.5.0) ships with the `http.file.allowAlias` property set to `false`. This setting prevents the use of symbolic links. Oracle strongly recommends that this setting not be changed to `true`, which might allow JSP source code to be visible to end users in some circumstances.

Instead of changing the property setting, you can use one of the following work-arounds:

- Temporarily switch from using the OC4J lightweight HTTP listener to front ending the OC4J application through Oracle HTTP Server, so that browsers access the pages indirectly through `mod_oc4j` and Apache JServ Protocol (AJP), rather than directly through HTTP.
- Replace all symbolic links in an application with the names of the real files they represent.

  You can use a shell script to automate the replacement of symbolic links. For example:

```
#!/bin/ksh

PROGNAME="${0##*/}"
LN_EXTN="ln"

function displaySyntax
{
echo "${PROGNAME}! SYNTAX: ${PROGNAME} some_dir_path"
exit 1
}

if [[ $# < 0 ]]
then
displaySyntax
fi

DIR="$1"

if [[ ! -d ${DIR}
then
displaySyntax
fi

find ${DIR} -type l|while read filepath
do
echo "FIXING: ${filepath} (=> ${filepath}.${LN_EXTN})"
mv ${filepath} ${filepath}.${LN_EXTN}
cp -L ${filepath}.${LN_EXTN} ${filepath}
done
```

This example KSH script would be invoked in a UNIX environment as follows:

```
$ fixLinks web_module_root
```

The script will go through any directory recursively and rename every file it finds that is a symbolic link with an additional `.ln` extension. Then the script will place a copy of the link target in the original location where the link was found.

## Managing the stdout and stderr Log Files

The following properties are used to manage standard `stderr` and `stdout` log files. You can specify the types and locations of the log files to which the properties pertain with the `-out` and `-err` options of the `oc4j.jar` command.

For example, the following command specifies rotation of the `stdout` and `stderr` files when the file size reaches 2.5 MB. 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
```

The next example specifies rotation of the `stdout` log file at 2:30 p.m. every day and limits the log archive to a maximum of 10 files:

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

***Table 4–2 Archive Management Properties for stdout and stderr Files***

| Property | Description |
|---|---|
| `stdstream.filesize=`*`max_file_size`* | The maximum size for file in the archive, in megabytes. A file will be rotated when it reaches this maximum. |
| `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. |

## Overview of General System Properties

Table 4–3 describes the general system properties that can be set for OC4J.

***Table 4–3 -D General System Properties for OC4J***

| Property | Description |
|---|---|
| `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`.<br>The deployment wizard uses 20 MB in swap space of the temporary directory for storing information during the deployment process. At completion, the deployment wizard cleans up the temp directory.<br>However, if the wizard is interrupted, it may not have the time or opportunity to clean up the temporary 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.<br>If you receive an `Out of Memory` error, check for space available in the temporary directory. |
| `java.awt.headless=true\|false` | If `true`, specifies checking on whether or not a display, keyboard, and mouse are supported in an environment. If `false`, this check is not performed. The default is `true`. |
| `oracle.home` | Sets the root directory into which you will install the OC4J distribution.<br>Instead of using the `oracle.home` system property, you can set the value of the `ORACLE_HOME` environment variable to the root directory. |
| `oracle.j2ee.home` | Sets the J2EE home directory to the installed directory of the `oc4j.jar` and `admin.jar` files, *`ORACLE_HOME`*`/j2ee/`*`instance`*. The value of *`ORACLE_HOME`* is the root directory into which you will install the OC4J distribution.<br>Setting this system property or the `J2EE_HOME` environment variable to the J2EE home directory enables you to invoke `oc4j.jar` and `admin.jar` from any 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`. |

*Table 4–3 (Cont.) -D General System Properties for OC4J*

| Property | Description |
|---|---|
| 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. |
| dedicated.rmicontext=true\|false | This property replaced the deprecated dedicated.connection property. The default is false.<br><br>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.<br><br>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.<br><br>You can also set this in the JNDI properties.<br><br>The oracle.j2ee.rmi.loadBalance property replaces the dedicated.rmicontext, dedicated.connection, and LoadBalanceOnLookup properties, which are deprecated in OC4J 10*g* (10.1.3.5.0). |
| manifest.dependencies.warn.only | This property can override the default behavior of halting OC4J initialization when JAR dependencies in its system libraries are not satisfied. The default property value is false.<br><br>When OC4J initializes, it validates the dependencies of the libraries defined in the container-level configurations by verifying their versions specified in the MANIFEST.MF file.<br><br>When this property is set to true, the system will only issue a warning message describing the problem and continue to initialize. For more information about using this property, see "Allowing Different Dependencies on a Shared Library with manifest.dependencies.warn.only" on page 4-5. |
| oracle.j2ee.rmi.loadBalance | This property configures replication-based load balancing, with one of these settings:<br><br>■ client: The client interacts with the OC4J process that was initially chosen at the first lookup for the entire conversation.<br>■ context: The client goes to a new server when a separate context is used (similar to the deprecated dedicated.rmicontext property).<br>■ lookup: The client goes to a new server for every lookup.<br><br>The default setting is client.<br><br>The oracle.j2ee.rmi.loadBalance property replaces the deprecated dedicated.rmicontext, dedicated.connection, and LoadBalanceOnLookup properties. |

*Table 4–3 (Cont.) -D General System Properties for OC4J*

| Property | Description |
|---|---|
| `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 in a Windows environment.<br>Normally when you use an MDB, it is blocked in a receive state waiting for incoming messages. However, if you shut down OC4J while the MDB is in a wait state in a Windows environment, the OC4J instance cannot be stopped and the applications are not undeployed since the MDB is blocked.<br>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 shut down OC4J using CTRL-C, then the OC4J process will hang for at least 2.5 hours.<br>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:<br>■ `none`: Disables metrics<br>■ `normal`: Medium number of metrics (default)<br>■ `heavy`: High number of metrics<br>■ `all`: Every possible metric<br>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 whether a third database table is used to manage the relationship. Set to `false` if you do not want a third association table. The default is `false`.<br>This property has been deprecated and works only with OrionCMP. |
| `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`.<br>When you change the value of this option and restart OC4J, it is valid only for applications deployed after the change. Any applications deployed before the change are not affected.<br>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.<br>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. |

*Table 4–3 (Cont.) -D General System Properties for OC4J*

| Property | Description |
|---|---|
| `oc4j.formauth.redirect=true\|false` | This property is applicable when form-based authentication is used by a Web application.<br><br>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.<br><br>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`. |
| `file.encoding=`*`value`* | This property specifies the default character set for file encoding. If an encoding is not specified, OC4J uses ISO-8859-1 as the default character set.<br><br>In a Windows environment, the default character set for file encoding should be Windows-1252, which is a superset of the ISO 8859-1 character set. To use the Windows-1252 character set as the default, specify `-Dfile.encoding=Cp1252` as an OC4J startup property. This prevents the replacement of a Windows-1252 character with a question mark ("?") in PrintWriter output. |
| `http.file.allowAlias` | This property controls the use of symbolic links. The default value is `false`, to prevent using symbolic links.<br><br>Oracle strongly recommends that this setting not be changed to `true`, which might allow JSP source code to be visible to end users in some circumstances.<br><br>For more information, see "Preventing the Use of Symbolic Links" on page 4-5. |
| `http.maxFileInfoCacheEntries` | This property controls the cache in which OC4J stores file information it collects when a client accesses an application using path information. The following settings enable you to control the cache:<br><br>■ Set `http.maxFileInfoCacheEntries < 0` to never cache the file information.<br><br>■ Set `http.maxFileInfoCacheEntries == 0` to store all the file information and not free objects from the cache.<br><br>■ Set `http.maxFileInfoCacheEntries > 0` to specify the maximum number of cached entries.<br><br>The default value is `2000`. |

*Table 4–3 (Cont.) -D General System Properties for OC4J*

| Property | Description |
|---|---|
| `http.proxyHost=`*`proxy_host`*<br>`http.proxyPort=`*`proxy_port`* | If your HTTP traffic goes through a proxy Web server, specify the proxy host and optionally the proxy port in the command line. If *proxy_port* is omitted, the default is port 80. |
| `http.webdir.enable=true\|false` | This property enables or disables servlet class name invocation for all servlets within the OC4J instance.<br><br>If set to `true`, any servlet running in the OC4J instance can be invoked by class name by default. If set to `false`, servlets cannot be invoked by class name. The default is `false`.<br><br>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.<br><br>The value set for `servlet-webdir` in `orion-web.xml` overrides the default value set for this attribute in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config/global-web-application`, which is `servlet-webdir="/servlet"`. |
| `HTTPClient.log.level` | This property enables logging for the Oracle `HTTPClient` package if set to one of these `java.util.logging.Level` values:<br><br>`CONFIG`<br>`FINE`<br>`FINER`<br>`FINEST`<br>`ALL`<br><br>To disable HTTPClient logging, set this property to `OFF`.<br><br>HTTPClient logs messages only at trace levels (`CONFIG` or lower) and does not use the application log levels (`SEVERE`, `WARNING`, and `INFO`).<br><br>For more information about setting log levels and enabling HTTPClient logging, see Chapter 4, "Logging Implementation Guidelines," in *Oracle Containers for J2EE Developer's Guide*. |

## Overview of Debug Properties

**Note:** The debug properties listed in this section are deprecated.

See "Using and Configuring the OC4J Component Loggers" on page 11-8 for details on using the component loggers provided with OC4J.

You can use the following properties for debugging applications running within OC4J. Debug messages are printed to the console. All properties take a Boolean value.

Preface all properties with a `-D`.

*Table 4–4 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. |

*Table 4–4 (Cont.) OC4J Debug Properties*

| Debug Property | Description |
|---|---|
| `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 debugging of Web services. |

# Enabling Remote Debugging from an Integrated Development Environment

You can debug applications on OC4J remotely, from an Integrated Development Environment (IDE), if you start the OC4J instance or instances with JVM debug commands, specified as start parameters, so that a remote debugger can connect. The following topics describe how to specify these parameters:

- Enabling Remote Debugging for an OC4J Instance with Application Server Control
- Specifying Debug Start Parameters in the opmn.xml File
- Specifying Debug Start Parameters on a Startup Command Line

## Enabling Remote Debugging for an OC4J Instance with Application Server Control

To enable remote debugging for a single OC4J instance with Application Server Control:

1. Navigate to the OC4J Home page.
2. Click **Administration** to display the OC4J Administration page.
3. Under Properties in the table of administration tasks, click the task icon in the Server Properties row.

   Enterprise Manager displays the Server Properties page.
4. In the `Start-parameters: Java Options` section under Command Line Options, click **Add Another Row** to add each of the following debug start parameters:
   - `-Xdebug`
   - `-Xnoagent`
   - `-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=4000`
5. Click **Apply** to apply your changes to the OC4J configuration.

When you make changes to the server properties, you must restart the OC4J instance before the changes take effect.

## Specifying Debug Start Parameters in the opmn.xml File

For OPMN-managed OC4J instances, you can put the debug parameters in the `opmn.xml` file, as the value of a `<data>` subelement where the `id` attribute is `java-options`, within a `<category>` element where the `id` attribute is `start-parameters`, and then restart the instance. The entry in `opmn.xml` should look like this one:

```
<module-data>
  <category id="start-parameters">
     <data id="java-options" value="-server -Xdebug -Xnoagent
      -Djava.compiler=NONE
      -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=4000
      -Djava.security.policy=$ORACLE_HOME/j2ee/home/config/java2.policy
      -Djava.awt.headless=true"/>
```

Make sure you never use `suspend=y`, which specifies not to start OC4J until the debugger is attached. If you used this debug parameter, OPMN would attempt to restart the OC4J instance or instances continuously because OPMN would not get a response from its query pings.

Attach to the server the port to which you set `address`, such as port 4000.

> **Note:** The port value of `4000` is arbitrary. You should set a value suitable for your connection. The specified port is the port that must be set in the remote debugging client to connect to the server.

## Specifying Debug Start Parameters on a Startup Command Line

For a standalone OC4J instance, you can specify the debug start parameters on a startup command line, as follows:

```
java -Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=4000 -jar oc4j.jar
```

## Debugging Web Applications Remotely

For debugging Web applications from an IDE, you need to set up for servlet and JSP debugging, as the following topics describe:

- Setting Up for Remote Debugging of Servlets
- Setting Up for Remote Debugging of JSPs

### Setting Up for Remote Debugging of Servlets

To set up remote debugging for a servlet:

1. Mark your project run or debug configuration to do remote debugging.
2. Set the attach to JPDA in the remote-debug specific runtime configuration node.
3. Start your OC4J instance with the debug parameters, as the example under "Specifying Debug Start Parameters on a Startup Command Line" on page 4-13 shows.
4. Set a breakpoint in your servlet.
5. Run a remote debugger.

   After you invoke the servlet from a Web browser, the servlet should reach the breakpoint.

### Setting Up for Remote Debugging of JSPs

For JSPs, you can set up as described in the preceding topic, "Setting Up for Remote Debugging of Servlets," but one more step is needed. You need to edit the `global-web-application.xml` file, which is installed in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` by default, and have at least the following parameters set for the JSP part:

```
<init-param>
   <param-name>debug</param-name>
   <param-value>true</param-value>
</init-param>
<init-param>
   <param-name>developer_mode</param-name>
   <param-value>true</param-value>
</init-param>
<init-param>
   <param-name>encode_to_java</param-name>
   <param-value>true</param-value>
</init-param>
<init-param>
   <param-name>reduce_tag_code</param-name>
   <param-value>false</param-value>
</init-param>
<init-param>
   <param-name>extra_imports</param-name>
   <param-value></param-value>
</init-param>
<init-param>
   <param-name>main_mode</param-name>
   <param-value>recompile</param-value>
</init-param>
<init-param>
   <param-name>debug_mode</param-name>
   <param-value>true</param-value>
</init-param>
```

# 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 one of the `oc4j` command scripts or the executable `oc4j.jar` archive.

### Starting OC4J with an oc4j Script

To start OC4J using an `oc4j` script, issue the following command from the *ORACLE_HOME*/`bin` directory:

```
oc4j -start
```

Before you can use this command, the `ORACLE_HOME` and `JAVA_HOME` environment variables must be set. See "Meeting Installation Prerequisites for a Standalone OC4J Server" on page 2-1 for details.

### 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 in this command starts OC4J using the default `server.xml` configuration file, which you can find in the `j2ee/home/config` directory. To start OC4J using a nondefault version of the `server.xml` file, issue the following command. You must supply the path to the modified configuration file.

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

You can 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 4-4.

## Starting OC4J in an Oracle Application Server Environment

In a managed configuration, all Oracle Application Server components, including OC4J and Oracle HTTP Server, must be started using OPMN, either from the Cluster Topology page in Application Server Control or with 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 startall
```

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

```
opmnctl startproc ias-component=default_group
```

In a cluster topology that includes multiple OC4J instances, if the EARs that the OC4J instances will use are in a shared directory at a single location, you should start the instances with the -sequential flag:

```
opmnctl startproc ias-component=default_group -sequential
```

This option prevents resource contention that might occur if you started all the OC4J instance in parallel.

Alternatively, to start the OC4J instances sequentially, you can specify the sequential option in opmn.xml, the OPMN configuration file for the cluster, as follows:

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

For more information about opmnctl commands, see the *Oracle Process Manager and Notification Server Administrator's Guide*.

---

**Note:** Before attempting to connect to Oracle Application Server in a Windows environment, you need to update the `C:\WINDOWS\system32\drivers\etc\hosts` file with the server IP address.

---

## Stopping OC4J in a Standalone Environment

You can stop a standalone OC4J server by invoking the `-shutdown` command in the `admin_client.jar` or `admin.jar` command-line utility or an `oc4j.cmd` or `oc4j` executable script.

---

**Note:** You should not use operating system commands such as `Control-C` in a Windows environment or `kill` in a Linux or UNIX environment to stop OC4J.

This is especially true when applications utilizing EJB modules are actively running within OC4J, as such commands do not allow EJB method calls or timer operations to complete before shutting down the server.

---

### Stopping Standalone OC4J with admin_client.jar

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

```
java -jar admin_client.jar uri adminId adminPassword -shutdown
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin password -shutdown
```

This command shuts down the entire OC4J server, terminating all threads immediately, as if the host machine were unplugged. If you use this command, the current state for clustered applications will not be replicated.

For descriptions of the *`uri`*, *`adminId`*, and *`adminPassword`* variables, see "Understanding the admin_client.jar Syntax and URI Specification" on page 6-2.

On a standalone OC4J instance, the `-shutdown` option of `admin_client.jar` is equivalent to the `-shutdown force` option of the `admin.jar` utility, which"Stopping OC4J with admin.jar" on page 5-3 describes.

### 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]
```

You can specify the following options:

- `ordinary|force`

  The type of shutdown. The default is `ordinary`, which allows each thread to terminate normally.

  The `force` option terminates all threads immediately. It is essentially the same as unplugging the host machine. If this option is used, the current state for clustered applications will not be replicated.

- *reason*

  You can 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. The string entered as the reason for the shutdown is written to the *ORACLE_HOME*/j2ee/home/config/server.log file.

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

### Stopping OC4J with an oc4j Script

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

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

For example:

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

The ORACLE_HOME and JAVA_HOME environment variables must be set to use this command. See "Meeting Installation Prerequisites for a Standalone OC4J Server" on page 2-1 for details.

## Stopping OC4J in an Oracle Application Server Environment

In a managed configuration, you can stop an OC4J instance from the Cluster Topology page of Application Server Control, with opmnctl, the OPMN command-line tool, or with the admin_client.jar command-line utility, which notifies OPMN that the instance has been stopped. The OPMN tool is installed in the *ORACLE_HOME*/opmn/bin directory. The admin_client.jar utility is installed by default in the *ORACLE_HOME*/j2ee/*instance* directory.

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

```
opmnctl stopall
```

You can use the following command to stop a specific managed component, in this case OC4J, on a local Oracle Application Server instance:

```
opmnctl stopproc ias-component=default_group
```

For more information about opmnctl commands, see the *Oracle Process Manager and Notification Server Administrator's Guide*.

Alternatively, you can use admin_client.jar to stop an OC4J instance, with the following command:

```
java -jar admin_client.jar uri adminId adminPassword -shutdown
```

For example:

```
java -jar admin_client.jar deployer:oc4j:opmn://localhost/home oc4jadmin password
-shutdown
```

This command shuts down the entire OC4J instance, terminating all threads immediately. For an OPMN-managed OC4J instance, `admin_client.jar` notifies OPMN that the server is being shut down on purpose, to prevent OPMN from attempting to restart it. If you use this command, the current state for clustered applications will not be replicated.

For descriptions of the *uri*, *adminId*, and *adminPassword* variables, see "Understanding the admin_client.jar Syntax and URI Specification" on page 6-2.

## Restarting an OC4J Instance in a Standalone Environment

You can use the `admin_client.jar` utility to restart a standalone OC4J server, with the following command:

```
java -jar admin_client.jar uri adminId adminPassword -restart
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin password -restart
```

For descriptions of the *uri*, *adminId*, and *adminPassword* variables, see "Understanding the admin_client.jar Syntax and URI Specification" on page 6-2.

Alternatively, you can use the `admin.jar` command-line utility to restart a standalone OC4J instance, with the following command:

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

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

## Restarting an OC4J Instance in an Oracle Application Server Environment

In a managed configuration, you can restart OC4J from the Cluster Topology page of Application Server Control, with `opmnctl`, the OPMN command-line tool, or with the `admin_client.jar` command-line utility. The OPMN tool is installed in the *ORACLE_HOME*/`opmn/bin` directory. The `admin_client.jar` utility is installed by default in the *ORACLE_HOME*/`j2ee/`*instance* directory.

You can use the following command to restart all OPMN-managed processes, including OC4J, on a local Oracle Application Server instance:

```
opmnctl startall
```

Alternatively, you can use the following command to restart a specific managed process, in this case OC4J, on a local Oracle Application Server instance:

```
opmnctl restartproc ias-component=home
```

In a cluster topology that includes multiple OC4J instances, you should restart the instances with the `opmnctl` command and `-sequential` flag:

```
opmnctl startproc ias-component=default_group -sequential
```

The `-sequential` flag prevents resource contention that might occur if you started all the OC4J instance in parallel, especially if the EARs that the OC4J instances will use are in a shared directory at a single location. If the `opmn.xml` configuration file for the cluster includes the `sequential` option, as described in "Starting OC4J in an Oracle

Application Server Environment" on page 5-2, you need not specify the `-sequential` flag.

You can use `admin_client.jar` to restart an OC4J instance, with the following command:

```
java -jar admin_client.jar uri adminId adminPassword -restart
```

For descriptions of the *`uri`*, *`adminId`*, and *`adminPassword`* variables, see "Understanding the admin_client.jar Syntax and URI Specification" on page 6-2.

## Understanding the Server Startup and Shutdown Sequence

This section provides a detailed sequence of events that occur during server startup and server shutdown.

### Server Startup Sequence of Events

The following events occur when the server starts up:

- The JAZN security framework starts
- The Application Server Thread Pool starts and creates the System and HTML thread pools
- The Container Service Manager starts and creates the Event Service, Timer Service, and Scheduler Service
- The OC4J class loader initializes
- The JMS Server starts
- The RMI Server starts
- The `system` application starts:
  - The `system.root:0.0.0` class loader initializes
  - Resource providers initialize
  - The EJB container initializes
  - Data sources initialize
  - Application clients initialize
- The `default` Application Starts:
  - The `default.root:0.0.0` class loader initializes
  - The `oc4jjms` resource provider initializes
  - The EJB container initializes
  - The data source connector initializes
  - Data sources initialize
  - The transaction manager starts
  - The `OracleASjms` connector initializes
  - Application clients initialize
- System MBeans are registered
- The `javasso` Application Starts:

  - The javasso.root:0.0.0 class loader initializes
  - resource providers initialize
  - The EJB container initializes
  - The data source connector initializes
  - Data sources initialize
  - Application clients initialize
- The ascontrol application starts:
  - The ascontrol.root:0.0.0 class loader initializes
  - resource providers initialize
  - The EJB container initializes
  - The data source connector initializes
  - Data sources initialize
  - Application clients initialize
- Your deployed applications are started. The sequence is the same as above for each application:
  - The class loader initializes
  - resource providers initialize
  - The EJB container initializes
  - Connectors initialize
  - The data source connector initializes
  - Data sources initialize
  - Application clients initialize
- The Recovery Manager starts
- The HTTP Server starts
- The default Web site starts
  - The default Web Application starts
  - The dms0 Web Application Starts
  - The JMXSoapAdapter-web Web Application starts
  - The jmsrouter_web Web Application starts
  - The javasso-web Web Application starts
  - The ascontrol Web Application starts
  - Your deployed Web applications start
- Additional Web sites (if applicable) start
- The Task Manager starts
- The OC4J Server initialization is complete

## Server Shutdown Sequence of Events

The following events occur when the server shuts down:

- The Task Manager stops
- The Container Services stop (Event Service, Timer Service, and Scheduler Service)
- The HTTP Server is destroyed
- The `javasso` Application is destroyed
- The `ascontrol` Application is destroyed
- Your deployed applications are destroyed
- The `default` Application is destroyed
- The `system` Application is destroyed
- The RMI Server is destroyed
- The JMS Server is destroyed
- The Transaction Manager shuts down
- Server Thread Pools are destroyed including the System and HTML thread pools
- Class Loaders are destroyed
- The IIOP Server shuts down
- The OC4J Server shutdown completes

# 6

# Using the admin_client.jar Utility

OC4J provides a command-line utility, `admin_client.jar`, for performing configuration, administration, and deployment tasks on active OC4J instances in an Oracle Application Server clustered environment as well as on a standalone OC4J server. In addition, you can use `admin_client.jar` to restart or stop an OC4J instance or group of instances.

The `admin_client.jar` utility is also part of the Administrative Client Utility for performing operations remotely, available on the companion CD or for downloading from Oracle Technology Network.

You can perform operations on a specific OC4J instance or simultaneously on all OC4J instances in a group. A **group** is a synchronized set of OC4J instances that belong to the same **cluster topology**, which is two or more loosely connected Oracle Application Server nodes. With the `admin_client.jar` command-line utility, you can perform the following operations on an OC4J instance or group of OC4J instances:

- Add Web sites
- Deploy an enterprise application archive (EAR), standalone Web module (WAR), Enterprise JavaBeans (EJB) module (EJB JAR), or standalone resource adapter (RAR)
- Undeploy an application, Web module, EJB module, or resource adapter
- Incrementally update a deployed EJB module with modified classes
- Bind and Unbind Web modules to a Web site and List details about Web bindings
- Create, modify, or remove shared libraries for an application
- Start, restart, or stop applications and list status and details for applications
- Restart or stop an OC4J instance or group of instances
- Add, test, list, and remove data sources and data source connection pools
- Add and remove JMS connection pools and destinations

You can perform similar operations with Application Server Control or OC4J Ant tasks. For more information, see "Using Application Server Control for Deployment" or "Using OC4J Ant Tasks for Deployment" in the *Oracle Containers for J2EE Deployment Guide*.

This chapter includes the following topics:

- Preparing to Use admin_client.jar
- Adding Web Sites
- Deploying an Archive

- Managing Web Bindings
- Redeploying an Archive
- Undeploying an Archive
- Updating Modified Classes in a Deployed EJB Module
- Creating and Managing Shared Libraries
- Managing Application Lifecycle
- Restarting and Stopping OC4J Instances
- Managing Data Sources
- Managing JMS Resources
- Managing OC4J Through a Remote Client

## Preparing to Use admin_client.jar

The `admin_client.jar` utility is installed by default in the *`ORACLE_HOME`*`/j2ee/`*`instance`* directory in each OC4J instance. This is the preferred command-line tool for performing operations on OC4J. This utility is also in the Administrative Client Utility for performing operations remotely, available on the companion CD for Oracle Application Server 10*g* Release 3 (10.1.3.5.0) or for downloading from Oracle Technology Network.

Before this utility can perform operations on an OC4J instance, the instance must be started.

This section covers these topics:

- Understanding the admin_client.jar Syntax and URI Specification
- Downloading and Extracting the Remote Administration Client
- Printing Usage Text to the Console
- Enabling Logging

### Understanding the admin_client.jar Syntax and URI Specification

The `admin_client.jar` utility uses the following syntax:

```
java -jar admin_client.jar uri adminId adminPassword command
```

The key parameter passed on the command line is *`uri`*, which specifies the target for the command or commands supplied. The syntax for the URI varies depending on the instance or instances being targeted. See the following topics for the format of this URI:

- Performing Operations on a Group of OC4J Instances Within a Cluster
- Performing Operations on a Specific OC4J Instance
- Performing Operations on a Standalone OC4J Server
- Validating a URI

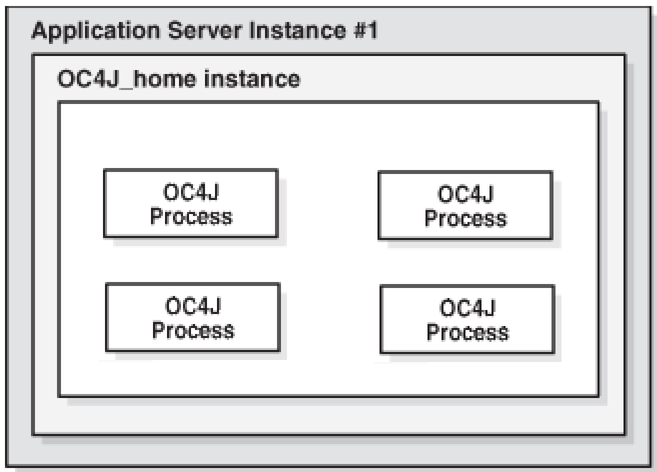
The OC4J administration user name and password are also passed to the `admin_client.jar` utility. The user name for the default administrator account is `oc4jadmin`.

As an example, the following command will start the `petstore` application, which is installed in the OC4J instance named `oc4j_2` on `node1`, a member of a cluster:

```
java -jar admin_client.jar deployer:oc4j:opmn://node1.company.com/oc4j_2
oc4jadmin password -application petstore -start
```

Figure 6–1 shows four processes that are configured to run from an OC4J instance named `OC4J_home` in one of the Oracle Application Server instances within a cluster.

***Figure 6–1 OC4J Instance Running on Multiple JVMs in an Oracle Application Server Instance Within a Cluster***




---

**Note:** The OC4J instance named `home` typically cannot be configured to run with multiple processes because it hosts the Application Server Control application, which is not suitable for running in the multiple-process model.

---

### Performing Operations on a Group of OC4J Instances Within a Cluster

Use the following URI to specify all OC4J instances in a group as the target. A **group** is a synchronized set of OC4J instances that belong to the same cluster topology. You can perform configuration, administration, and deployment operations simultaneously on all OC4J instances in the group. For example, you could stop all OC4J instances that belong to a group named `oc4j_soa` simultaneously within an Oracle Application Server cluster.

The URI utilizes the OPMN-based clustering framework, in which cluster nodes are aware of one another. You need to supply only the host name and, optionally, the OPMN request port for any Oracle Application Server node within the cluster. The application is then able to retrieve the host names and OPMN ports for all other nodes within the cluster.

The URI syntax follows:

```
deployer:cluster:[rmis]:opmn://opmnHost[:opmnPort]/groupName
```

For example:

```
deployer:cluster:opmn://node1.company.com/oc4j_soa
```

*Table 6–1 URI Parameters for Targeting a Group*

| Parameter | Description |
|---|---|
| `rmis` | Optional. Include if the target utilizes ORMI over SSL, or ORMIS. |
| *`opmnHost`* | Required. The host name of an Oracle Application Server node within a cluster. Any node can be specified; the list of other nodes in the cluster will be retrieved from this node. |
| *`opmnPort`* | Optional. The OPMN request port, as specified in `opmn.xml`. If no port is specified, the default port, `6003`, will be used. |
| *`groupName`* | Required. The name of the group to which the OC4J instances belong, within a cluster. |

### Performing Operations on a Specific OC4J Instance

Use the following URI syntax to target a specific OPMN-managed OC4J instance, including an instance within a cluster. In the prefix, `oc4j` replaces `cluster`.

Specify the host name for the Oracle Application Server node hosting the instance. If you are not sure of the host name or port for the node, you can specify the host name for another node within the cluster, as well as the name of the Oracle Application Server instance. The application will then use the OPMN clustering framework to locate the node hosting the Oracle Application Server instance.

The URI syntax follows:

```
deployer:oc4j:[rmis]:opmn://host[:opmnPort]/[iASInstanceName]/oc4jInstanceName
```

For example:

```
deployer:oc4j:opmn://server.company.com:6004/instance2/home
```

*Table 6–2 URI Parameters for Targeting a Specific Instance*

| Parameter | Description |
|---|---|
| `rmis` | Optional. Include if the target utilizes ORMI over SSL, or ORMIS. |
| `host` | Required. The host name of the Oracle Application Server node to target within the cluster to use as the OPMN server. |
| *`opmnPort`* | Optional. The OPMN request port, as specified in `opmn.xml`. If no port is specified, the default port, `6003`, will be used. |
| *`iASInstanceName`* | Optional. The name of the Oracle Application Server instance to target, if it does not reside on the node specified for *`host`*. |
| *`oc4jInstanceName`* | Required. The name of the target OC4J instance. |

### Performing Operations on a Standalone OC4J Server

Use one of the following URIs to target a standalone OC4J server instance.

If you are using RMI, specify the URI as follows:

```
deployer:oc4j:host:rmiPort
```

For example:

```
deployer:oc4j:myserver:23791
```

If you are using ORMI over SSL (ORMIS), specify the URI as follows:

```
deployer:oc4j:rmis:host:ormisPort
```

For example:

```
deployer:oc4j:rmis:myserver:23943
```

***Table 6–3 URI Parameters for Targeting Standalone OC4J***

| Parameter | Description |
|---|---|
| `rmis` | Required if the target utilizes ORMI over SSL, or ORMIS. |
| *`host`* | Required. The host name of an Oracle Application Server node within the cluster. Any node can be specified; the list of other nodes in the cluster will be retrieved from this node. |
| *`rmiPort`* | Required if RMI used. The RMI port, as specified in the instance-specific `rmi.xml` file. |
| *`ormisPort`* | Required if ORMIS is used. The SSL port, as specified in the instance-specific `rmi.xml` file. |

### Validating a URI

You can validate a URI using the `-validateURI` command.

```
java -jar admin_client.jar uri adminId adminPassword -validateURI
```

For example:

```
java -jar admin_client.jar deployer:cluster:opmn://node1.company.com/default_group
oc4jadmin password -validateURI
```

## Downloading and Extracting the Remote Administration Client

The Administrative Client Utility distribution contains the `admin_client.jar` command-line utility. This utility can connect to OC4J or Oracle Application Server targets and perform a range of life cycle, deployment, and resource configuration operations.

Consider the scenario in which a remote system needs to perform regular operations against an Oracle Application Server instance. For example, a remote system might have some automated build or test process, such as deployment operations or querying or manipulating some application-specific or server JMX MBeans for administrative purposes. Or perhaps the remote system performs a regularly scheduled test-to-production set of configuration and deployment operations. The Administrative Client Utility can be used to do this, removing the need for the remote system to have a full OC4J or Oracle Application Server installation.

The Administrative Client Utility, a separate distribution for Oracle Application Server 10*g* Release 3 (10.1.3.5.0), is available for downloading from Oracle Technology Network and is on the Oracle Application Server companion CD. The distribution file, `oc4j_admin_client_101350.zip`, contains all you need to manage an OC4J instance remotely:

- The Java libraries required to establish remote JMX connections, using the ORMI protocol, to either an OC4J or Oracle Application Server target
- The executable `admin_client.jar` utility with the libraries it requires to operate
- The standard J2EE libraries relevant to the remote client role

**To download and extract the Administrative Client Utility:**

1. Download `oc4j_admin_client_101350.zip` from the Oracle Technology Network:

   ```
   http://download.oracle.com/otn/java/oc4j/10131/oc4j_admin_client_101350.zip
   ```

2. Extract the contents of `oc4j_admin_client_101350.zip` into a local directory. For example:

   ```
   >mkdir oc4j_admin_client
   >cd oc4j_admin_client
   >jar xvf d:\software\oc4j_admin_client_101350.zip
   ```

   The resulting directory structure looks like this:

   ```
   \j2ee
      \home
         oc4jclient.jar
         admin_client.jar
         \lib
            ejb.jar
            mail.jar
            adminclient.jar
            javax88.jar
            javax77.jar
            jmx_remote_api.jar
            jmxri.jar
   \lib
      xmlparserv2.jar
      dms.jar
   \opmn
      \lib
      optic.jar
   \jlib
      oraclepki.jar
      ojpse.jar
   ```

   The following URIs use different patterns for different OC4J targets:

   - Standalone OC4J server:

     ```
     deployer:oc4j:test-cycle.oracle.com:23791
     ```

   - Specific OC4J instance on Oracle Application Server:

     ```
     deployer:oc4j:opmn://test-cycle.oracle.com/testunit
     ```

   - Group of OC4J instances within a cluster:

     ```
     deployer:cluster:opmn://test-cycle.oracle.com/[groupName]
     ```

3. Connect `admin_client.jar` to a target OC4J instance or instances and test the connection. For example:

   ```
   >cd j2ee\home
   >java -jar admin_client.jar
     deployer:oc4j:opmn://test-cycle.oracle.com/testunit
       oc4jadmin welcome1
     -validateURI

   URI deployer:oc4j:opmn://test-cycle.oracle.com/testunit is valid and connected
   ```

## Printing Usage Text to the Console

To print the online help text for the `admin_client.jar` commands to the console, simply type `-help` on the command line. For example:

```
java -jar admin_client.jar -help
```

To view detailed help for a specific command, type `-usage` followed by the command identifier. For example:

```
java -jar admin_client.jar -usage [command]
```

## Enabling Logging

To help troubleshoot errors that occur when running `admin_client.jar`, you can enable Java logging when running this tool. Log messages will be output to the console.

To enable logging:

1. Create a `logging.properties` file containing a single line:

   ```
   oracle.oc4j.admin.jmx.client.CoreRemoteMBeanServer.level=INFO
   ```

   If you create this file in a location other than *ORACLE_HOME*/j2ee/*instance*, you must include the path to the file in the following command.

2. Set `-Djava.util.logging.config.file=logging.properties` on the `admin_client.jar` command line as follows:

   ```
   java -Djava.util.logging.config.file=logging.properties -jar admin_client.jar
   uri adminId adminPassword command
   ```

You can set the value in the `logging.properties` file to one of the Java log-level values in Table 6–4.

***Table 6–4 Java Log Levels***

| Java Log Level | Description |
|---|---|
| SEVERE | Log system errors requiring attention from the system administrator. |
| WARNING | Log actions or a conditions discovered that should be reviewed and might require action before an error occurs. |
| INFO | Log normal actions or events. This could be a user operation, such as *login completed*, or an automatic operation, such as a *log file rotation*. |
| CONFIG | Log messages or problems related to log configuration. |
| FINE | Log trace or debug messages used for debugging or performance monitoring. Typically contains detailed event data. |
| FINER | Log fairly detailed trace or debug messages. |
| FINEST | Log highly detailed trace or debug messages. |

For example:

```
oracle.oc4j.admin.jmx.client.CoreRemoteMBeanServer.level=FINE
```

# Adding Web Sites

You can use the -addWebSite command to add a Web site on a standalone OC4J instance or on an OC4J instance within a cluster. The new Web site will include the default Web application from the default Web site. See Chapter 13, "Managing Web Sites in OC4J," for detailed information about OC4J Web sites and how to manually add Web sites.

> **Note:** The -addWebSite command cannot be used to create a Web site on multiple OC4J instances within a group.

The syntax for this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -addWebSite -webSiteName
site-name -protocol protocol -port port [-keystorePath path]
[-keystorePassword password] [-sslProvider class-name]
```

The following example creates a new Web site called test-web-site on an OC4J standalone instance:

```
java -jar admin_client.jar deployer:oc4j:localhost:23791 oc4jadmin welcome1
-addWebSite -webSiteName test-web-site.xml -protocol http -port 8899
```

***Table 6–5 -addWebSite Command Parameters***

| Parameter | Description |
|---|---|
| -webSiteName | Required. The name for the Web site. The name must use the form *name*-web-site. For example, test-web-site. In addition, the Web site name must be unique on the OC4J instance. |
| -protocol | Required. The protocol to be used by the Web site. The protocol can be http, https, ajp, ajps. The ajp protocol can only be used by one Web site on an OC4J instance. |
| -port | Required. The port number to be used by the Web site. Two Web sites can share the same port number only if they both use the http or https protocol. |
| -keystorePath | Optional. The filename, including the path, of the keystore file. This parameter is required when using https or ajps and should not be specified when using http or ajp. |
| -keystorePassword | Optional. The password of the keystore file. This parameter is required when using https or ajps and should not be specified when using http or ajp. |
| -sslProvider | Optional. The third-party SSLServerSocketFactory implementation. The default SSLServerSocketFactory implementation is used if no implementation is specified. |

# Deploying an Archive

You can use the admin_client.jar utility to deploy an application (EAR or application directory), a standalone Web module (WAR), a standalone EJB module (JAR), or a standalone resource adapter (RAR) to a specific OC4J instance or to a group of OC4J instances.

This section covers the following topics:

- Deploying a J2EE Application (EAR)

- Deploying a J2EE Application from a Remote Client
- Deploying a Standalone Web Module (WAR)
- Deploying a Standalone EJB Module (JAR)
- Deploying a Standalone Resource Adapter (RAR)
- Using a Script File for Batch Deployment

---

**Note:** Deploying an archive across a group requires that all instances have the same `oc4jadmin` account password.

---

## Deploying a J2EE Application (EAR)

Use the `-deploy` command to deploy a J2EE application that is packaged as an EAR file or a J2EE application that is in the standard enterprise application directory structure. A J2EE application's modules can be packaged or left in their directory structure as well. The syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword -deploy -file path/filename
-deploymentName appName [-bindAllWebApps [webSiteName]] [-targetPath path]
[-parent appName] [-deploymentDirectory path] [sequential [waitsec]] [-enableIIOP]
[-iiopClientJar path/filename] [-deploymentPlan path/filename] [-removeArchive]
```

You can include the `-bindAllWebApps` parameter to bind all Web modules within the EAR to the Web site through which they will be accessed. If no Web site is specified, modules will be bound to the `default` Web site.

For example, the following command deploys the `utility` application to all OC4J instances that belong to the group `default_group` within a cluster. All Web modules within the application will be bound to the `default` Web site.

```
java -jar admin_client.jar deployer:cluster:opmn://node1.company.com/default_group
oc4jadmin password -deploy -file C:/dev/utility.ear -deploymentName utility
-bindAllWebApps
```

The application may also be assembled in a standard J2EE application directory structure. Indicate the directory using the `-file` parameter. The following example deploys the application located in the `utility` directory:

```
java -jar admin_client.jar deployer:cluster:opmn://node1.company.com/default_group
oc4jadmin password -deploy -file C:/dev/utility -deploymentName utility
-bindAllWebApps
```

***Table 6–6 -deploy Command Parameters for EAR Deployment***

| Parameter | Description |
|---|---|
| `-file` | Required. The file path of the archive or application directory to be deployed. The application directory must be assembled in a standard J2EE application directory structure when using directory-based deployment. |
| `-deploymentName` | Required. The user-defined application deployment name, used to identify the application within OC4J. |

*Table 6–6 (Cont.) -deploy Command Parameters for EAR Deployment*

| Parameter | Description |
|---|---|
| `-bindAllWebApps` | Optional. Binds all Web modules in the EAR to the specified Web site or, if none is specified, to the `default` Web site.<br><br>You can supply a value for *`webSiteName`*, which is the *`name`* portion of the *`name`*`_web-site.xml` file that contains the Web site configuration.<br><br>If this parameter is not specified, you can use the `-bindAllWebApps` command after deployment. For more information about this command, see "Binding All Web Modules to a Single Web Site" on page 6-16. |
| `-targetPath` | Optional. The directory to deploy the EAR to. If a directory is not specified, the EAR is deployed to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications` directory by default.<br><br>The deployed EAR file is also copied to this directory. Each successive deployment will cause this EAR file to be overwritten. |
| `-parent` | Optional. The parent application of this application. The default is the `default` application or global Web application. |
| `-deploymentPlan` | Optional. The path and file name for a deployment plan to apply to the application. The plan would have been saved during a previous deployment as an XML file. The file must exist on the local host. |
| `-deploymentDirectory` | Optional. The directory containing the OC4J-specific deployment descriptors and generated files, such as compiled JSP classes and EJB wrapper classes.<br><br>The default directory is *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications`. |
| `-sequential` [*`waitsec`*] | Optional. Specify to deploy the archive to each OC4J instance in a group. The deployment to each target OC4J instance must complete before deployment begins on the next target instance. Requests will not be routed to an OC4J instance while the EAR is being deployed to it.<br><br>You can use the *`waitsec`* option to specify a number of seconds to wait between deployments, as follows:<br><br>`-sequential 15`<br><br>For more information about the *`waitsec`* option, see "Specifying a Delay Between Sequential Redeployments in a Cluster" on page 6-20.<br><br>If this parameter is not specified, the archive will be simultaneously deployed to all OC4J instances in the group by default.<br><br>This option is valid only in a clustered environment. It is not valid for standalone OC4J. |

*Table 6–6 (Cont.) -deploy Command Parameters for EAR Deployment*

| Parameter | Description |
|---|---|
| `-enableIIOP` | Optional. Specify this parameter to generate IIOP client stubs on the OC4J server.<br>The application-level stubs generated for all EJB modules are output to an archive named `_iiopClient.jar` in the *ORACLE_HOME*/j2ee/*instance*/application-deployments/*appName* directory. In addition, stubs for each individual EJB module are generated in an archive with the same name in the *ORACLE_HOME*/j2ee/*instance*/application-deployments/*appName*/*ejbModuleName* directory.<br>The `GenerateIIOP` system property must be enabled at OC4J startup to use this feature. This property is set as `-DGenerateIIOP=true` on the OC4J command line for standalone OC4J or as an `oc4j-options` value in `opmn.xml`. |
| `-iiopClientJar` | Optional. The path and file name of the JAR to output IIOP client stubs to.<br>The application-level stubs generated for all EJB modules are output to an archive named `_iiopClient.jar` in the *ORACLE_HOME*/j2ee/*instance*/application-deployments/*appName* directory. If a path is supplied, the archive is also set on this path.<br>In addition, stubs for each individual EJB module are generated in an archive with the same name in the *ORACLE_HOME*/j2ee/*instance*/application-deployments/*appName*/*ejbModuleName* directory.<br>The `GenerateIIOP` system property must be enabled at OC4J startup to use this feature. This property is set as `-DGenerateIIOP=true` on the OC4J command line for standalone OC4J or as an `oc4j-options` value in `opmn.xml`. |
| `-removeArchive` | Optional. Specify to delete the EAR file from the server's file system after deployment. |

## Deploying a J2EE Application from a Remote Client

The following example shows how to deploy an EAR from a remote client to a specific OC4J instance on Oracle Application Server:

```
cd j2ee/home
>java -jar admin_client.jar
 deployer:oc4j:opmn://test-cycle.oracle.com/testunit
 oc4jadmin welcome1
 -deploy
 -file d:\temp\rupg\testru.ear
 -deploymentName testru -bindAllWebApps

06/06/20 17:00:16 Notification ==>Uploading file testru.ear ...
06/06/20 17:00:18 Notification ==>Application Deployer for testru STARTS.
06/06/20 17:00:19 Notification ==>Copy the archive to /scratch/sbutton/m1_
060618/j2ee/admin/applications/testru.ear
06/06/20 17:00:19 Notification ==>Initialize /scratch/sbutton/m1_
060618/j2ee/admin/applications/testru.ear begins...
06/06/20 17:00:19 Notification ==>Unpacking testru.ear
06/06/20 17:00:20 Notification ==>Done unpacking testru.ear
06/06/20 17:00:20 Notification ==>Unpacking testru-web.war
06/06/20 17:00:20 Notification ==>Done unpacking testru-web.war
06/06/20 17:00:20 Notification ==>Initialize /scratch/sbutton/m1_
060618/j2ee/admin/applications/testru.ear ends...
```

```
06/06/20 17:00:20 Notification ==>Starting application : testru
06/06/20 17:00:20 Notification ==>Initializing ClassLoader(s)
06/06/20 17:00:20 Notification ==>Initializing EJB container
06/06/20 17:00:20 Notification ==>Loading connector(s)
06/06/20 17:00:20 Notification ==>Starting up resource adapters
06/06/20 17:00:20 Notification ==>Initializing EJB sessions
06/06/20 17:00:20 Notification ==>Committing ClassLoader(s)
06/06/20 17:00:20 Notification ==>Initialize testru-web begins...
06/06/20 17:00:20 Notification ==>Initialize testru-web ends...
06/06/20 17:00:21 Notification ==>Started application : testru
06/06/20 17:00:21 Notification ==>Binding web application(s) to site
default-web-site begins...
06/06/20 17:00:21 Notification ==>Binding testru-web web-module for application
testru to site default-web-site under context root /testru
06/06/20 17:00:22 Notification ==>Binding web application(s) to site
default-web-site ends...
06/06/20 17:00:22 Notification ==>Application Deployer for testru COMPLETES.
Operation time: 3785 msecs
```

## Deploying a Standalone Web Module (WAR)

Use the `-deploy` command to deploy a standalone Web module packaged as a WAR file.

> **Note:** The `-deploy` command does not support directory-based deployment for standalone Web modules. The Web module must be packaged as a WAR file. However, directory-based deployment of a Web module is supported if the Web module directory is included within a J2EE application directory structure with a respective `META-INF/application.xml` file. In this case, deploy the application instead of the Web module.

The WAR-specific syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword -deploy -file
path/filename -deploymentName appName [-bindAllWebApps [webSiteName]]
[-targetPath path] [-parent appName] [-deploymentDirectory path]
[-contextRoot context]
[-removeArchive]
```

The WAR can be designated a child of another deployed application that does not already contain a Web module component; otherwise, the WAR will be deployed to the `default` application.

A WAR cannot be deployed as the child of an application that already contains a Web module. That is, if the `acme` application already contains `acme-web.war`, an additional WAR file cannot be deployed into that application. Repackage the WAR in the application's EAR file and redeploy the application instead.

The following command deploys the standalone `acme-web.war` Web module to the `default` application in all OC4J instances that belong to `default_group` within the cluster of which `node1` is a member. Because the `-bindAllWebApps` parameter is included, but a Web site to bind to is not specified, the module will be bound to the `default` Web site.

```
java -jar admin_client.jar deployer:cluster:opmn://node1.company.com/default_group
oc4jadmin password -deploy -file C:/dev/acme-web.war -deploymentName utility
```

```
-bindAllWebApps -parent default
```

***Table 6–7 -deploy Command Parameters for WAR Deployment***

| Parameter | Description |
|---|---|
| `-file` | Required. The path and file name of the archive to deploy. |
| `-deploymentName` | Required. The user-defined name for the Web module, used to identify it within OC4J. |
| `-bindAllWebApps` | Optional. Binds the Web module to the specified Web site or, if none is specified, to the `default` Web site.<br><br>You can supply a value for *`webSiteName`*, which is the *`name`* portion of the *`name`*`_web-site.xml` file that contains the Web site configuration. |
| `-targetPath` | Optional. The directory to deploy the archive to. If a directory is not specified, the archive is deployed to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications` directory by default.<br><br>The generated EAR file containing the standalone WAR file is also copied to this directory. Each successive deployment will cause this archive to be overwritten. |
| `-parent` | Optional. The parent application the module will be deployed to. The default is the `default` application. |
| `-deploymentDirectory` | Optional. The directory containing the OC4J-specific deployment descriptors and generated files, such as compiled JSP classes.<br><br>The default directory is *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application-deployments`. |
| `-contextRoot` | Optional. The Web module context root, which will be appended to the URL used to access the application through a Web browser. If the contest root is not specified, the value passed in for `-deploymentName` will be used.<br><br>For example, if you supply `/petstore` as the context root, the module could be accessed with the following URL:<br><br>`http://node1.company.com:7777/petstore` |
| `-removeArchive` | Optional. Include to delete the WAR file from the server's file system after deployment. |

## Deploying a Standalone EJB Module (JAR)

Use the `-deploy` command to deploy a standalone EJB module packaged as a JAR file. The syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword -deploy -file path/filename
-deploymentName jarName [-targetPath path] [-parent appName] [-deploymentDirectory
path] [-removeArchive]
```

The following command deploys the standalone `acme-ejb.jar` EJB module to the `default` application in all OC4J instances that belong to `default_group` within the cluster of which `node1` is a member.

```
java -jar admin_client.jar deployer:cluster:opmn://node1.company.com/default_group
oc4jadmin password -deploy -file C:/dev/acme-ejb.jar -deploymentName acme
```

***Table 6–8 -deploy Command Parameters for EJB JAR Deployment***

| Parameter | Description |
|---|---|
| `-file` | Required. The path and file name of the archive to deploy. |

*Table 6–8 (Cont.) -deploy Command Parameters for EJB JAR Deployment*

| Parameter | Description |
|---|---|
| `-deploymentName` | Required. The user-defined name for the EJB module, used to identify it within OC4J. |
| `-targetPath` | Optional. The directory to deploy the EJB JAR to. If a directory is not specified, the EJB JAR is deployed to the *ORACLE_HOME*/j2ee/*instance*/applications directory by default.<br>The deployed EJB JAR file is also copied to this directory. Each successive deployment will cause this EJB JAR file to be overwritten. |
| `-parent` | Optional. The parent application the EJB module will be deployed to. The default is the `default` application. |
| `-deploymentDirectory` | Optional. The directory containing the OC4J-specific deployment descriptors. The default directory is *ORACLE_HOME*/j2ee/*instance*/applications-deployments. |
| `-removeArchive` | Optional. Delete the JAR file from the server's file system after deployment. |

## Deploying a Standalone Resource Adapter (RAR)

Use the `-deploy` command to deploy or redeploy a Java Connector Architecture-compliant resource adapter packaged as a RAR file. By default, resource adapters are deployed to the *ORACLE_HOME*/j2ee/*instance*/connectors directory.

Redeploying or undeploying a standalone RAR does not require a restart of the `default` application.

The RAR-specific syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword -deploy -file path/filename
-deploymentName connectorName [-nativePathLib path] [-grantAllPermissions]
[-removeArchive]
```

The following command deploys the `acme-rar.rar` module to all OC4J instances that belong to `default_group` within a cluster.

```
java -jar admin_client.jar deployer:cluster:opmn://node1.company.com/default_group
oc4jadmin password -deploy -file /dev/acme-rar.rar -deploymentName acme-rar
-grantAllPermissions -removeArchive
```

*Table 6–9 -deploy Command Parameters for RAR Deployment*

| Parameter | Description |
|---|---|
| `-file` | Required. The path and file name of the RAR file to deploy. |
| `-deploymentName` | Required. The user-defined connector name, used to identify the connector within OC4J. |
| `-nativeLibPath` | Optional. The path to the directory containing native libraries (such as DLLs) within the RAR file. |
| `-grantAllPermissions` | Optional. Include this parameter to grant all runtime permissions requested by the resource adapter, if required. |
| `-removeArchive` | Optional. Include this parameter to delete the RAR file from the server's file system after deployment. |

For more information, see "Deploying Resource Adapters" in the *Oracle Containers for J2EE Deployment Guide*.

### Using a Script File for Batch Deployment

You can specify a script file that contains deployment commands on the `admin_client.jar` command line. If you specify a file in the `-script` command, `admin_client.jar` can do a list of commands with only one connection to the deployment manager. The syntax for batch deployment follows:

```
java -jar admin_client.jar uri adminId adminPassword
-script filename
```

The script file, *`filename`*, contains multiple lines, like the lines in this example:

```
-deploy -file /scratch/rpan/apps/hello-planet.ear -deploymentName hello-planet
-bindWebApp -appName hello-planet -webModuleName hello-planet-web
-stop hello-planet
-start hello-planet
-redeploy -file /scratch/rpan/apps/hello-planet.ear
-deploymentName hello-planet -bindAllWebApps
-undeploy hello-planet
-validateURI
```

You can convert to batch mode by looking at the script or logs from an installation and extracting the relevant lines used by an existing configuration assistant.

## Managing Web Bindings

You can use the `admin_client.jar` utility to: bind Web modules to a Web site; unbind Web modules from a Web site; and list the current Web module bindings for a Web site. These commands can be run for a specific OC4J instance or for a group of OC4J instances in a cluster.

This section covers the following topics:

- Binding Web Modules to a Web Site After Deployment
- Unbinding Web Modules from a Web Site
- Listing Web Bindings

### Binding Web Modules to a Web Site After Deployment

Every Web module deployed to OC4J must be bound to a Web site through which it will be accessed.

Typically, you will bind Web modules packaged as WAR files within an EAR at the time the EAR is deployed using the `-bindAllWebApps` parameter on the `-deploy` command. However, if the `-bindAllWebApps` parameter was not specified when the EAR was deployed, you can bind modules to a Web site after deployment, as the following topics describe:

- Binding All Web Modules to a Single Web Site
- Binding a Specific Web Module to a Web Site and Setting the Context Root

### Binding All Web Modules to a Single Web Site

Use the `-bindAllWebApps` command to bind all Web modules within a J2EE application to the same Web site, or to `default-web-site` by default. The syntax for this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -bindAllWebApps
-appName appName -webSiteName siteName -shared true/false -loadOnStartup
true/false -accessLog true/false
```

***Table 6–10 -bindAllWebApps Command Parameters***

| Parameter | Description |
|---|---|
| `-appName` | Required. The name of the parent application as specified at deployment time. |
| `-webSiteName` | Optional. The Web site name to which the Web module tries to bind. The *`siteName`* is the same as the Web site XML configuration file name. For example, `default-web-site`.<br>Web modules are bound to the default Web site (`default-web-site`) on the target OC4J instances if this parameter is not specified. |
| `-shared` | Optional. The application is allowed to be shared between HTTP/HTTPS. The default value is `false`. |
| `-loadOnStartup` | Optional. The application is allowed to be loaded on startup. The default value is `true`. |
| `-accessLog` | Optional. The application is allowed to enable access logging. The default value is `true`. |

### Binding a Specific Web Module to a Web Site and Setting the Context Root

Use the `-bindWebApp` command to bind a specific Web module within a J2EE application to a Web site you specify or to the `default` Web site. You can also set the context root that will be used to access the Web module.

The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -bindWebApp
-appName name -webModuleName name -webSiteName name
-contextRoot contextRoot -shared true/false -loadOnStartup true/false -accessLog
true/false
```

***Table 6–11 -bindWebApp Command Parameters***

| Parameter | Description |
|---|---|
| `-appName` | Required. The name of the parent application as specified at deployment time. |
| `-webModuleName` | Required. The name of the Web module to be bound. This should be the name of the WAR file contained within the EAR file, without the `.war` extension. |
| `-webSiteName` | Optional. The Web site name to which the Web module tries to bind. The *`name`* is the same as the Web site XML configuration file name. For example, `default-web-site`.<br>Web modules are bound to the default Web site (`default-web-site`) on the target OC4J instances if this parameter is not specified. |

*Table 6–11 (Cont.) -bindWebApp Command Parameters*

| Parameter | Description |
|---|---|
| `-contextRoot` | Optional. The context root for the Web module. This will be appended to the URL used to access the application through a Web browser; for example:<br>`http://localhost:8888/petstore`.<br>If a context root is not supplied, the context root specified in the parent application's `application.xml` deployment descriptor will be used. |
| `-shared` | Optional. The application is allowed to be shared between HTTP/HTTPS. The default value is `false`. |
| `-loadOnStartup` | Optional. The application is allowed to be loaded on startup. The default value is `true`. |
| `-accessLog` | Optional. The application is allowed to enable access logging. The default value is `true`. |

## Unbinding Web Modules from a Web Site

Web Modules can be unbound from a Web site after deployment. You can unbind all Web Modules from a Web site or you can unbind a specific Web module from a Web site.

- Unbinding All Web Modules
- Unbinding a Specific Web Module

### Unbinding All Web Modules

Use the `-unbindAllWebApps` command to remove all Web module bindings from a Web site in an OC4J instance or in a group of OC4J instances that are part of a cluster.

The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -unbindAllWebApps
-appName appname  [-webSiteName web-site-name]
```

The following example unbinds all Web modules belonging to the `hello` application from all Web sites named `default-web-site` in a cluster that consists of multiple OC4J instances that are listening on the OPMN request port `6003`:

```
java -jar admin_client.jar deployer:cluster:opmn://localhost:6003/default_group
oc4jadmin welcome1 -unbindAllWebApps -appName hello -webSiteName default-web-site
```

*Table 6–12 -unbindAllWebApps Command Parameters*

| Parameter | Description |
|---|---|
| `-appName` | Required. The name of the parent application as specified at deployment time. |
| `-webSiteName` | Optional. The Web site name from which the Web modules try to unbind. The *`web-site-name`* is the same as the Web site XML configuration file name. For example, `default-web-site`.<br>Web modules are unbound from the default Web site (`default-web-site`) on the target OC4J instances if this parameter is not specified. |

### Unbinding a Specific Web Module

Use the `-unbindWebApp` command to remove a specific Web module binding from a specific Web site in an OC4J instance or in a group of OC4J instances that are part of a cluster.

The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -unbindWebApp
-appName appname  -webModuleName web-module-name [-webSiteName web-site-name]
```

The following example unbinds the Web module `hello-web` for the `hello` application from the Web site `default-web-site` on an OC4J standalone instance:

```
java -jar admin_client.jar deployer:oc4j:localhost:23971 oc4jadmin welcome1
-unbindWebApp -appName hello -webModuleName hello-web
```

***Table 6–13 -unbindWebApp Command Parameters***

| Parameter | Description |
|---|---|
| `-appName` | Required. The name of the parent application as specified at deployment time. |
| `-webModuleName` | Required. The name of the Web module to be unbound. This should be the name of the WAR file contained within the EAR file, without the `.war` extension. |
| `-webSiteName` | Optional. The Web site name from which the Web module tries to unbind. The *`web-site-name`* is the same as the Web site XML configuration file name. For example, `default-web-site`.<br><br>The Web module is unbound from the default Web site (`default-web-site`) on the target OC4J instances if this parameter is not specified. |

## Listing Web Bindings

Use the `-listWebBindings` command to display the Web site bindings for each Web module in an OC4J instance or in a group of OC4J instances that are part of a cluster. The following information is listed by default: application name, module name, context root, and Web site name. For more detailed information, use the `-verbose` option, which is described below.

The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -listWebBindings [webSiteName
web-site-name][-verbose]
```

The following example displays detailed information for the Web modules bound to all Web sites named, `default-web-site,` in a cluster that consists of multiple OC4J instances that are listening on the OPMN request port `6003`:

```
java -jar admin_client.jar deployer:cluster:opmn://localhost:6003/default_group
 oc4jadmin welcome1 -listWebBindings -webSiteName default-web-site -verbose
```

*Table 6–14 -listWebBindings Command Parameters*

| Parameter | Description |
|---|---|
| `-webSiteName` | Optional. The Web site name for which to view all Web bindings. The *`web-site-name`* is the same as the Web site XML configuration file name. For example, `default-web-site`. All Web bindings for all Web sites are displayed if no Web site is specified. |
| `-verbose` | Optional. Displays more details. The additional details include: pre-load, shared, access log, and maximum inactivity time. |

# Redeploying an Archive

Use the `-redeploy` command to redeploy a previously deployed archive.

This operation performs a *graceful* redeployment because it stops the application if it is running and then undeploys the archive. It then deploys and restarts the application. Redeploying an archive with the `-deploy` command, in contrast, does not stop the application but simply undeploys, redeploys, and then restarts it.

The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -redeploy -file path/filename
-deploymentName appName [-bindAllWebApps] [-isConnector] [-keepSettings]
[-sequential [waitsec]] [-removeArchive]
```

*Table 6–15 -redeploy Command Parameters*

| Parameter | Description |
|---|---|
| `-file` | Required. The path and file name of an EAR, WAR, or RAR file to redeploy. |
| `-deploymentName` | Required. The user-defined application deployment name, used to identify the application within OC4J. This value must match the name of the existing application on the server. |
| `-isConnector` | Required for redeploying a standalone RAR. |
| `-bindAllWebApps` | Optional. Binds all Web modules in an EAR to the specified Web site or, if none is specified, to the `default` Web site.<br><br>You can supply a value for *`webSiteName`*, which is the *`name`* portion of the *`name`*`_web-site.xml` file that contains the Web site configuration.<br><br>Alternatively, you can bind all Web modules to a Web site later, as described in "Binding Web Modules to a Web Site After Deployment" on page 6-15. |
| `-keepSettings` | Optional. If this parameter is specified, the redeployed application will fetch and use the deployment plan from the previous deployment. Values set in deployment descriptors packaged within the archive will be ignored.<br><br>If this parameter is not specified, values will be set to those in the deployment descriptors packaged with the archive. |

*Table 6–15 (Cont.) -redeploy Command Parameters*

| Parameter | Description |
|---|---|
| `-sequential` [*waitsec*] | Optional. Specify to redeploy the archive to each OC4J instance in a group in sequence. The redeployment to each target OC4J instance must complete before redeployment begins on the next target instance. Requests will not be routed to an OC4J instance while the archive is being redeployed to it.<br><br>You can use the *waitsec* option to specify a number of seconds to wait between deployments, as follows:<br><br>`-sequential 15`<br><br>If this parameter is not specified, the archive will be simultaneously deployed to all OC4J instances in the group by default.<br><br>This option is valid only in a clustered environment. It is not valid for standalone OC4J. |
| `-removeArchive` | Optional. Specify to delete the EAR, WAR, or RAR file from the server's file system after deployment. |
| `-failureRecovery` | Optional. Enable recovery from a failed redeployment. The previous archive is redeployed if possible. |

## Specifying a Delay Between Sequential Redeployments in a Cluster

When an application is redeployed to a group with the `-sequential` parameter of the `admin_client.jar -redeploy` command, the redeployment operation is serialized, with redeployment done to one OC4J instance at a time so that the target application is never entirely in a stopped state. In a sequential redeployment, the deployment manager immediately commences redeployment on the next OC4J instance that is running a member of an application cluster as soon as the redeployment operation completes on the current OC4J instance. The result is that the system might not be able to stabilize itself so that the new application instance is fully active before the next redeployment commences, which introduces these possible side effects:

- The application can become inaccessible while it is stopped on one OC4J instance and before `mod_oc4j` is notified that the application is available on another instance.
- Session replication activities might not have had an opportunity to execute.

In some circumstances, the session state of an application might be lost when you redeploy an application to a cluster with the `admin_client.jar -redeploy` command, even if you specify the `-sequential` and `-keepsettings` parameters.

You can use the *waitsec* option of the `-sequential` parameter to specify a number of seconds between redeployments to different OC4J instances that are running an application cluster. This delay can provide enough time for replication of session state.

If you specify the optional *waitsec* value, the deployment manager waits the specified number of seconds between redeployment operations on OC4J instances within a group. This delay enables the system to stabilize as redeployment operations occur across the group, reducing the opportunities for applications to be inaccessible or session state to be lost.

For example, the following `admin_client.jar -redeploy` command specifies a delay of 15 seconds between redeployments to different OC4J instances:

```
java -jar admin_client.jar deployer:cluster:opmn://host:port/home oc4jadmin
password -redeploy -file "myapp.ear" -deploymentName rolling -sequential 15
```

```
-keepsettings
```

The new *waitsec* option also applies to the `-sequential` parameter of the `admin_client.jar -deploy` command.

### Redeploying an Application with Scheduled Jobs

If you redeploy an application that has scheduled jobs, the jobs will not run as scheduled unless you remove all the jobs before the redeployment and resubmit them after it.

**To redeploy an application with scheduled jobs:**

1. Remove all scheduled jobs.
2. Redeploy the application.
3. Resubmit all the jobs.

## Undeploying an Archive

The `-undeploy` command removes an application, standalone Web, standalone EJB, or standalone connector module from the target OC4J instances, as the following topics describe:

- Undeploying an EAR, Standalone WAR, and Standalone EJB JAR
- Undeploying a Standalone RAR

### Undeploying an EAR, Standalone WAR, and Standalone EJB JAR

Undeploying an EAR, standalone Web module, or standalone EJB JAR removes it from the OC4J runtime. Existing Web site bindings are also deleted.

The syntax for undeploying an EAR, standalone WAR, or standalone EJB JAR follows. The name of the application or module must be supplied.

```
java -jar admin_client.jar uri adminId adminPassword -undeploy appName
```

### Undeploying a Standalone RAR

The syntax for undeploying a standalone RAR follows. The `-isConnector` parameter must be included along with name of the connector.

```
java -jar admin_client.jar uri adminId adminPassword -undeploy connectorName
-isConnector
```

Undeploying a standalone RAR does not require a restart of the `default` application.

## Updating Modified Classes in a Deployed EJB Module

The `-updateEJBModule` command performs incremental or partial redeployment of EJB modules within an application running in an OC4J instance or in a group of OC4J instances. This feature makes it possible to redeploy only those beans within an EJB JAR that have changed.

---

**Note:** Incremental redeployment may be more efficient than redeploying the entire application for CMP or BMP entity beans but not for session beans, message-driven beans, or EJB 3.0 JPA entities. For details about whether to use this feature, see "Incremental Redeployment of Updated EJB Modules" in the *Oracle Containers for J2EE Deployment Guide*.

---

The syntax for updating modified classes in a deployed EJB module follows. The name of the application the EJB JAR is part of must be supplied. If updating a standalone EJB module, specify the `default` application.

```
java -jar admin_client.jar uri adminId adminPassword -updateEJBModule
-appName appName -ejbModuleName ejbJarName -file path/ejbJarName
```

For example:

```
java -jar admin_client.jar deployer:cluster:opmn://node1.company.com/default_group
oc4jadmin password -updateEJBModule -appName petstore
-ejbModuleName customerEjb.jar -file build/customerEjb.jar
```

***Table 6–16 -updateEJBModule Syntax***

| Option | Description |
|---|---|
| `-appName` | Required. The name of the application that the EJB module is part of. If you are updating a standalone EJB module, specify the `default` application. |
| `-ejbModuleName` | Required. The name of the EJB JAR file to be updated, as defined in `application.xml`. |
| `-file` | Required. The path and file name of the updated EJB JAR. |

# Creating and Managing Shared Libraries

You can use the `admin_client.jar` utility to create and manage shared libraries in an OC4J instance or in a group of OC4J instances, as the following topics describe:

- Installing a Shared Library
- Modifying an Existing Shared Library
- Viewing the Contents of a Shared Library
- Listing All Shared Libraries
- Removing a Shared Library
- Importing an Existing Shared Library
- Deleting an Imported Shared Library
- Stopping a Shared Library from being Inherited
- Allowing a Shared Library to be Inherited

## Installing a Shared Library

You can use the `-publishSharedLibrary` command to create the shared library directory structure and install the binaries that compose the library within it in a specific OC4J instance or in a group of OC4J instances. The shared library will be created in the *ORACLE_HOME*/j2ee/*instance*/shared-lib directory of each OC4J instance.

The command will also declare the shared library within a <shared-library> element in the server.xml file on each OC4J instance, making it available to applications.

The syntax for installing a shared library follows. The path and file names for multiple code sources, binaries that will compose the shared library, can be specified, each separated from the next by a space.

```
java -jar admin_client.jar uri adminId adminPassword -publishSharedLibrary
-name libName -version libVersion [-parentName parentLibName]
[-parentVersion parentLibVersion] [-installCodeSources path [path ...]]
[-addCodeSources path [path ...]] [-imports sharedLibName
[:min-version][,max-version] [sharedLibName ...]]
```

The following command deploys the acme.common:2.5 shared library to a group of OC4J instances (all the members of default_group) within a cluster.

```
java -jar admin_client.jar
deployer:cluster:opmn://server.company.com:6004/default_group
oc4jadmin password -publishSharedLibrary -name acme.common -version 2.5
-installCodeSources /myserver/tmp/acme-apis.jar /myserver/tmp/acmeImpl.jar
```

The resulting directory structure within a target OC4J server would be as follows:

```
ORACLE_HOME/j2ee/home/shared-lib
  /acme.common
    /2.5
      acme-apis.jar
      acmeImpl.jar
```

***Table 6–17 -publishSharedLibrary Command Parameters***

| Parameter | Description |
|---|---|
| -name | Required. The name of the shared library.<br>Where common APIs are implemented by multiple vendors, the name should include both the vendor name and the name of the technology; for example, oracle.jdbc or xerces.xml. |
| -version | Required. The version number of the shared library. This value should ideally reflect the code implementation version. |
| -parentName | Optional. The name of the parent shared library, if applicable. |
| -parentVersion | Optional. The version number of the parent shared library, if applicable. |
| -installCodeSources | The path and file names for one or more JAR or ZIP files to be uploaded to the OC4J instance or instances and installed as part of the shared library. Separate each path/file name string from the next with a space. |
| -addCodeSources | Optional. The path and file names for JAR or ZIP files that have already been uploaded to the OC4J instance or instances to add to the shared library. Separate each path/file name string from the next with a space. |
| -imports | Optional. The name of one or more existing shared libraries to import into this shared library. Separate each name string from the next with a space.<br>You can specify the maximum or minimum version, or both, of the library to import. |

## Modifying an Existing Shared Library

You can use the `-modifySharedLibrary` command to modify the contents of an existing shared library. The command will also update the shared library definition within the `server.xml` file on each OC4J instance.

The syntax for modifying an existing shared library follows. The path and file names for multiple code sources, binaries that will compose the shared library, can be specified, each separated from the next by a space.

```
java -jar admin_client.jar uri adminId adminPassword -modifySharedLibrary
-name libName -version libVersion [-installCodeSources path [path ...]]
[-addCodeSources path [path ...]] [-removeCodeSources path [path ...]]
[-addImports sharedLibName[:min-version][,max-version] [sharedLibName ...]]
[-removeImports sharedLibName[:min-version][,max-version] [sharedLibName ...]]
```

The following command updates the `acme.common:2.5` shared library.

```
java -jar admin_client.jar
deployer:cluster:opmn://server.company.com:6004/default_group
oc4jadmin password -modifySharedLibrary -name acme.common -version 2.5
-addCodeSources /myserver/tmp/acme-helpers.jar
```

***Table 6–18 -modifySharedLibrary Command Parameters***

| Parameter | Description |
|---|---|
| `-name` | Required. The name of the shared library to update. |
| `-version` | Required. The version number of the shared library to update. |
| `-installCodeSources` | Optional. The path and file name to a JAR or ZIP file to be uploaded to the OC4J instance or instances and installed as part of the shared library. Separate each path/file name string from the next with a space. |
| `-addCodeSources` | Optional. The path and file name for one or more JAR or ZIP files that have already been uploaded to the OC4J instance or instances to add to the shared library. Separate each path/file name string from the next with a space. |
| `-removeCodeSources` | Optional. The path and file name for one or more JAR or ZIP files to remove from the shared library. Separate each path/file name string from the next with a space. |
| `-addImports` | Optional. The name of one or more existing shared libraries to import into this shared library. Separate each name string from the next with a space.<br><br>You can specify the maximum or minimum version, or both, of the library to import. |
| `-removeImports` | Optional. The name of one or more existing shared libraries to remove from this shared library.<br><br>You can specify the maximum or minimum version, or both, of the library to remove. |

## Viewing the Contents of a Shared Library

Use the `-describeSharedLibrary` command to view the code sources and imported shared libraries that compose the specified shared library. The syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword -describeSharedLibrary
-name libName -version libVersion
```

*Table 6–19 -describeSharedLibrary Command Parameters*

| Parameter | Description |
|---|---|
| `-name` | Required. The name of the shared library. |
| `-version` | Required. The version number of the shared library. |

## Listing All Shared Libraries

Use the `-listSharedLibraries` command to output a list of all shared libraries defined in the target OC4J instance or instances. The syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword -listSharedLibraries
```

**Note:** If you are using JDK1.4, Oracle Application Server 10*g* Release 3 (10.1.3.5.0) does not support using the Xalan library shipped with the JDK as a shared library. To use the Xalan library, you have two alternatives:

- Use JDK 5.0 (JDK 1.5) or JDK 6, in which the embedded Xalan library *is* supported as a shared library.
- With JDK1.4, use a standalone distribution of the Xalan library instead of the embedded version.

## Removing a Shared Library

Use the `-removeSharedLibrary` command to remove a shared library from the OC4J target. The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -removeSharedLibrary
-name libName -version libVersion
```

*Table 6–20 -removeSharedLibrary Command Parameters*

| Parameter | Description |
|---|---|
| `-name` | Required. The name of the shared library to remove. |
| `-version` | Required. The version number of the shared library to remove. |

## Importing an Existing Shared Library

Use the `-addImportSharedLibrary` command to import an existing shared library to an application's classloader. The command is equivalent to adding an `<import-shared-library>` element to an application's `orion-application.xml` descriptor. This command requires an application restart for the change to take effect. Refer to "Installing a Shared Library" on page 6-22 for instructions on installing a shared library. The syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword -addImportSharedLibrary
-appName application -name name -minVer MinVersion -maxVer MaxVersion
```

The following example imports the `oracle.jdbc` shared library to an application named `Myapp`:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-addImportSharedLibrary -appName Myapp -name oracle.jdbc
```

*Table 6–21 -addImportSharedLibrary Command Parameters*

| Parameter | Description |
|---|---|
| `-appName` | Required. The name of the application, as defined at deployment time, to which the shared library is imported. |
| `-name` | Required. The name of an existing shared library to add to the given application. |
| `-minVer` | Optional. The minimum version number of the library required by an application. |
| `-maxVer` | Optional. The maximum version number of the library required by an application. |

## Deleting an Imported Shared Library

Use the `-deleteImportSharedLibrary` command to delete a shared library from an application's classloader. The command is equivalent to deleting an `<import-shared-library>` element from an application's `orion-application.xml` descriptor. This command requires an application restart for the change to take effect. The syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword -deleteImportSharedLibrary
-appName application -name name
```

The following example deletes the `oracle.jdbc` shared library from the application named `Myapp`:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-deleteImportSharedLibrary -appName Myapp -name oracle.jdbc
```

*Table 6–22 -deleteImportSharedLibrary Command Parameters*

| Parameter | Description |
|---|---|
| `-appName` | Required. The name of the application, as defined at deployment time, from which the shared library is deleted. |
| `-name` | Required. The name of the shared library to remove from the given application. |

## Stopping a Shared Library from being Inherited

Use the `-addRemoveInheritedSharedLibrary` command to stop a shared library from being inherited by an application's classloader. The command is equivalent to adding a `<remove-inherited>` element to an application's `orion-application.xml` descriptor. This command requires an application restart for the change to take effect. The syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword
-addRemoveInheritedSharedLibrary -appName application -name name
```

The following example stops the `oracle.jdbc` shared library from being inherited by the application named `Myapp`:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-addRemoveInheritedSharedLibrary -appName Myapp -name oracle.jdbc
```

*Table 6–23 -addRemoveInheritedSharedLibrary Command Parameters*

| Parameter | Description |
|---|---|
| -appName | Required. The name of the application, as defined at deployment time, that will not inherit the shared library. |
| -name | Required. The name of the shared library to stop from being inherited. |

## Allowing a Shared Library to be Inherited

Use the -deleteRemoveInheritedSharedLibrary command to allow a shared library to be inherited by an application's classloader. The command is equivalent to deleting a <remove-inherited> element from an application's orion-application.xml descriptor. This command requires an application restart for the change to take effect. The syntax follows:

```
java -jar admin_client.jar uri adminId adminPassword
-deleteRemoveInheritedSharedLibrary -appName application -name name
```

The following example allows the oracle.jdbc shared library to be inherited by the application named Myapp:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-deleteRemoveInheritedSharedLibrary -appName Myapp -name oracle.jdbc
```

*Table 6–24 -deleteRemoveInheritedSharedLibrary Command Parameters*

| Parameter | Description |
|---|---|
| -appName | Required. The name of the application, as defined at deployment time, that will inherit the shared library. |
| -name | Required. The name of the shared library to be inherited. |

# Managing Application Lifecycle

You can use the admin_client.jar utility to start, restart, or stop an application and its child applications in a specific OC4J instance or in a group of OC4J instances. You can also list the status of deployed applications in a specific OC4J instance or in a group of OC4J instances. The following topics are included in this section:

- Starting Applications
- Stopping Applications
- Restarting Applications
- Listing Applications

## Starting Applications

Use the -start command to start an application and its child applications on target OC4J instances. Applications are automatically redeployed at startup if a file within the application has been modified.

The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -start appName
```

The -start command requires the application name as specified at deployment time. The following example starts the ascontrol application on node2 within a cluster:

```
java -jar admin_client.jar deployer:oc4j:opmn://node2.company.com:6004/home
oc4jadmin password -start ascontrol
```

## Stopping Applications

Use the `-stop` command to stop an application and its child applications on target OC4J instances. By default, applications are stopped immediately. Any requests that are currently being processed are lost. For planned shutdown scenarios, an application can have a specified amount of time to complete request processing before the application is stopped.

The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -stop appName [-timeout
timeInSeconds] [-graceful true|false]
```

The `-stop` command requires the application name as specified at deployment time. The following example stops the `ascontrol` application on `node2` within a cluster. The application is allowed 5 seconds to complete requests before the application is forcefully stopped.

```
java -jar admin_client.jar deployer:oc4j:opmn://node2.company.com:6004/home
oc4jadmin password -stop ascontrol -timeout 5
```

***Table 6–25 -stop Command Parameters***

| Parameter | Description |
|---|---|
| `-timeout` | Optional. The amount of time to wait for the application to stop gracefully. The application is stopped forcefully after the timeout is reached. The default timeout is `0` if no timeout is specified. |
| `-graceful` | Optional. The graceful option specifies the method used to stop the application. The value `true` implies that the application server waits for the in-flight requests to complete before stopping the application. The value `false` implies that in-flight requests terminate and the application is stopped immediately (forcefully). The default setting is `true`.<br><br>The `-graceful` parameter takes precedence over the `-timeout` parameter if the value is set to `false`. |

## Restarting Applications

Use the `-restartApp` command to stop and then start an application and its child applications on target OC4J instances. Applications are automatically redeployed at startup if a file within the application has been modified. By default, applications are stopped immediately. Any requests that are currently being processed are lost. For planned shutdown scenarios, an application can have a specified amount of time to complete request processing before the application is stopped.

The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -restartApp appName [-timeout
timeInSeconds] [-graceful true|false]
```

The `-restartApp` command requires the application name as specified at deployment time. The following example restarts the `ascontrol` application on `node2` within a cluster. The application is allowed 5 seconds to complete requests before the application is forcefully stopped and then started.

```
java -jar admin_client.jar deployer:oc4j:opmn://node2.company.com:6004/home
oc4jadmin password -restartApp ascontrol -timeout 5
```

***Table 6–26 -restartApp Command Parameters***

| Parameter | Description |
|---|---|
| `-timeout` | Optional. The amount of time to wait for the application to stop gracefully. The application is stopped forcefully after the timeout is reached. The default timeout is `0` if no timeout is specified. |
| `-graceful` | Optional. The graceful option specifies the method used to stop the application. The value `true` implies that the application server waits for the in-flight requests to complete before stopping the application. The value `false` implies that in-flight requests terminate and the application is stopped immediately (forcefully). The default setting is `true`.<br>The `-graceful` parameter takes precedence over the `-timeout` parameter if the value is set to `false`. |

## Listing Applications

Use the `-listApplications` command to display the status of applications that are currently deployed in an OC4J instance or in a group of OC4J instances that are part of a cluster. The following status information is listed by default: application name, contained modules, application type, application state, and parent application. For more detailed information, use the `-verbose` option, which is described below.

The syntax of this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -listApplications [-verbose]
```

The following example displays detailed information for the deployed applications on a cluster that consists of multiple OC4J instances that are listening on the OPMN request port `6003`:

```
java -jar admin_client.jar deployer:cluster:opmn://localhost:6003/default_group
 oc4jadmin welcome1 -listApplications -verbose
```

***Table 6–27 -listApplications Command Parameters***

| Parameter | Description |
|---|---|
| `-verbose` | Optional. Displays more details. The additional details include: application context root binding, routing enabled, group name, and state replication. |

# Restarting and Stopping OC4J Instances

You can use the `admin_client.jar` utility to stop a standalone OC4J server, a specific OC4J instance in a managed environment, or a group of OC4J instances. The `-shutdown` command shuts down the specified OC4J instance or instances and for any OPMN-managed instance, notifies OPMN that it is being shut down. The `-restart` command restarts the specified instance or instances.

The following topics provide the syntax and examples for these commands:

- Restarting an OC4J Instance or Group of Instances
- Stopping an OC4J Instance or Instances

### Restarting an OC4J Instance or Group of Instances

Use the `admin_client.jar -restart` command, as follows, to restart an OC4J instance or group of OC4J instances:

```
java -jar admin_client.jar uri adminId adminPassword -restart
```

For example, the following command restarts a standalone OC4J server:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin password -restart
```

The following command restarts all of the OC4J instances that are members of `default_group` in each Oracle Application Server within the cluster topology:

```
java -jar admin_client.jar deployer:cluster:opmn://node1.company.com/default_group
oc4jadmin password -restart
```

### Stopping an OC4J Instance or Instances

Use the `admin_client.jar -shutdown` command, as follows, to stop an OC4J instance or group of OC4J instances:

```
java -jar admin_client.jar uri adminId adminPassword -shutdown
```

For example, the following command stops a standalone OC4J server:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin password -shutdown
```

This command shuts down the entire OC4J server, terminating all threads immediately, as if the host machine were unplugged. If you use this command, the current state for clustered applications will not be replicated.

The following command stops the specified OC4J instance in an OPMN-managed Oracle Application Server environment:

```
java -jar admin_client.jar deployer:oc4j:opmn://localhost/home oc4jadmin password
-shutdown
```

The next command stops all of the OC4J instances that are members of `default_group` in each Oracle Application Server within the cluster topology:

```
java -jar admin_client.jar deployer:cluster:opmn://node1.company.com/default_group
oc4jadmin password -shutdown
```

These commands shut down the specified instance or instances and terminate all threads immediately. If you use the `-shutdown` command, the current state for clustered applications will not be replicated. For each OPMN-managed OC4J instance, `admin_client.jar` notifies OPMN that the server is being shut down on purpose, to prevent OPMN from attempting to restart it.

## Managing Data Sources

You can use the `admin_client.jar` utility to manage data sources in an OC4J instance or in a group of OC4J instances, as the following topics describe:

- Adding, Testing, Listing, and Removing Data Source Connection Pools
- Adding, Testing, Listing, and Removing Data Sources

## Adding, Testing, Listing, and Removing Data Source Connection Pools

You can use the `admin_client.jar` utility to add, test, list, and remove data source connection pools in an OC4J instance or in a group of OC4J instances, as the following topics describe:

- Adding a Data Source Connection Pool
- Testing a Data Source Connection Pool
- Listing Data Source Connection Pools
- Removing a Data Source Connection Pool

### Adding a Data Source Connection Pool

Use the `-addDataSourceConnectionPool` command to add a data source connection pool for an application in an OC4J instance or in each OC4J instance of a group within a cluster.

The syntax for adding a data source connection pool follows:

```
java -jar admin_client.jar uri adminId adminPassword -addDataSourceConnectionPool
-applicationName applicationName -name name -factoryClass factoryClass
-dbUser dbUser -dbPassword dbPassword -url url
[-factoryProperties name1 value1 [name2 value2 [...]]]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-addDataSourceConnectionPool -applicationName default -name ScottConnectionPool
-factoryClass oracle.jdbc.pool.OracleDataSource
-dbUser scott -dbPassword tiger -url jdbc:oracle:thin:@localhost:1521:xe
```

***Table 6–28 -addDataSourceConnectionPool Command Parameters***

| Parameter | Description |
|---|---|
| `-applicationName` | Required. The name of the application for which to add the data source connection pool. |
| `-name` | Required. The name of the connection pool. |
| `-factoryClass` | Required. The fully qualified path of the connection factory implementation. |
| `-dbUser` | Required. The default user name to use to get connections. |
| `-dbPassword` | Required. The default password to use to get connections. |
| `-url` | Required. The connection factory URL to use to get connections. |
| `-factoryProperties` | Optional. One or more property name and value pairs to set on the connection factory definition. |

### Testing a Data Source Connection Pool

Use the `-testDataSourceConnectionPool` command to test an application's connection to a data source connection pool in an OC4J instance or in each OC4J instance of a group within a cluster.

The syntax for testing a connection to a data source connection pool follows:

```
java -jar admin_client.jar uri adminId adminPassword -testDataSourceConnectionPool
-name name -sqlStatement sqlStatement [-applicationName applicationName]
[-dbUser dbUser] [-dbPassword dbPassword]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-testDataSourceConnectionPool -sqlStatement "select * from dual"
-applicationName default -name ScottConnectionPool
```

***Table 6–29 -testDataSourceConnectionPool Command Parameters***

| Parameter | Description |
|---|---|
| `-name` | Required. The name of the connection pool. |
| `-sqlStatement` | Required. The SQL statement to use to test the connection |
| `-applicationName` | Optional. The name of the application for which to test the data source connection pool. |
| `-dbUser` | Optional. The default user name to use to get connections. |
| `-dbPassword` | Optional. The default password to use to get connections. |

### Listing Data Source Connection Pools

Use the `-listDataSourceConnectionPools` command to view a list of data source connection pools that are configured for an application. The list includes each connection pool's configured properties.

The syntax for listing data source connection pools follows:

```
java -jar admin_client.jar uri adminId adminPassword
 -listDataSourceConnectionPools [-applicationName applicationName]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost:23791
 oc4jadmin oc4j -listDataSourceConnectionPools -applicationName default
```

***Table 6–30 -listDataSourceConnectionPools Command Parameters***

| Parameters | Description |
|---|---|
| `-applicationName` | Optional. The name of the application for which to list configured data source connection pools. The `default` application's connection pools are listed if no application name is specified. |

### Removing a Data Source Connection Pool

Use the `-removeDataSourceConnectionPool` command to remove a data source connection pool from an application in an OC4J instance or in each OC4J instance of a group within a cluster. The syntax for removing a data source connection pool follows:

```
java -jar admin_client.jar uri adminId adminPassword
-removeDataSourceConnectionPool -name name [-applicationName applicationName]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-removeDataSourceConnectionPool -name ScottConnectionPool -applicationName default
```

***Table 6–31 -removeDataSourceConnectionPool Command Parameters***

| Parameter | Description |
|---|---|
| `-name` | Required. The name of the connection pool. |

*Table 6–31 (Cont.) -removeDataSourceConnectionPool Command Parameters*

| Parameter | Description |
|---|---|
| `-applicationName` | Optional. The name of the application from which to remove the data source connection pool. |

## Adding, Testing, Listing, and Removing Data Sources

You can use the `admin_client.jar` utility to add, test, list, and remove data sources in an OC4J instance or in a group of OC4J instances, as the following topics describe:

- Adding a Managed Data Source
- Removing a Managed Data Source
- Adding a Native Data Source
- Removing a Native Data Source
- Testing a Database Connection
- Testing a Data Source
- Listing Data Sources
- Getting the Data Sources Descriptor for an Application

### Adding a Managed Data Source

Use the `-addManagedDataSource` command to add a managed data source for an application in an OC4J instance or in each OC4J instance of a group within a cluster. The syntax for adding a managed data source follows:

```
java -jar admin_client.jar uri adminId adminPassword -addManagedDataSource
-applicationName applicationName -name name
-jndiLocation jndiLocation -connectionPoolName connectionPoolName
[-dbUser dbUser] [-dbPassword dbPassword] [-loginTimeout loginTimeout]
[-txLevel txLevel] [-dbSchema dbSchema] [-manageLocalTransactions true|false]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-addManagedDataSource -applicationName default -name ScottDataSource
-jndiLocation jdbc/ScottDataSource -connectionPoolName ScottConnectionPool
```

*Table 6–32 -addManagedDataSource Command Parameters*

| Parameter | Description |
|---|---|
| `-applicationName` | Required. The name of the application for which to add the data source. |
| `-name` | Required. The name of the data source. |
| `-jndiLocation` | Required. The location to use to bind the new data source into JNDI. |
| `-connectionPoolName` | Required. The name of the connection pool with which the data source interacts. |
| `-dbUser` | Optional. The default user for the new data source. |
| `-dbPassword` | Optional. The default password for the new data source. |
| `-loginTimeout` | Optional. The login timeout for the new data source. |
| `-txLevel` | Optional. The transaction level (`local` or `global`). |

*Table 6–32 (Cont.) -addManagedDataSource Command Parameters*

| Parameter | Description |
|---|---|
| `-dbSchema` | Optional. The database schema to use if the EJB CMP implementation being used is Orion CMP. (TopLink CMP is the default.) |
| `-manageLocalTransactions` | Optional. Indicates whether or not OC4J should manage local transactions. The default value is `true`. |

### Removing a Managed Data Source

Use the `-removeManagedDataSource` command to remove a managed data source from an application in an OC4J instance or in each OC4J instance of a group within a cluster. The syntax for removing a managed data source follows:

```
java -jar admin_client.jar uri adminId adminPassword -removeManagedDataSource
-name name [-applicationName applicationName]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-removeManagedDataSource -name ScottDataSource -applicationName default
```

*Table 6–33 -removeManagedDataSource Command Parameters*

| Parameter | Description |
|---|---|
| `-name` | Required. The name of the data source to remove. |
| `-applicationName` | Optional. The name of the application from which to remove the data source. |

### Adding a Native Data Source

Use the `-addNativeDataSource` command to add a native data source for an application in an OC4J instance or in each OC4J instance of a group within a cluster. The syntax for adding a native data source follows:

```
java -jar admin_client.jar uri adminId adminPassword -addNativeDataSource
-name name -dbUser dbUser -dbPassword dbPassword
-jndiLocation jndiLocation -loginTimeout loginTimeout
-dataSourceClass dataSourceClass -url url [-applicationName applicationName]
[-properties name1 value1 [name2 value2 ][...]]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-addNativeDataSource -name ScottDataSource
-dbUser scott -dbPassword tiger -jndiLocation jdbc/ScottNativeDataSource
-loginTimeout 5 -dataSourceClass com.acme.DataSourceImpl
-url jdbc:oracle:thin:@localhost:1521:xe
```

*Table 6–34 -addNativeDataSource Command Parameters*

| Parameter | Description |
|---|---|
| `-name` | Required. The name of the new data source. |
| `-dbUser` | Required. The default user for the new data source. |
| `-dbPassword` | Required. The default password for the new data source. |
| `-jndiLocation` | Required. The location to use to bind the new data source into JNDI. |

*Table 6–34 (Cont.) -addNativeDataSource Command Parameters*

| Parameter | Description |
|---|---|
| `-loginTimeout` | Required. The login timeout for the new data source. |
| `-dataSourceClass` | Required. The fully qualified class of the new data source. |
| `-url` | Required. The url used by the new data source to connect to the database. |
| `-applicationName` | Optional. The name of the application for which to add the data source. |
| `-properties` | Optional. The property or properties for the new data source. |

### Removing a Native Data Source

Use the `-removeNativeDataSource` command to remove a native data source from an application in an OC4J instance or in each OC4J instance of a group within a cluster. The syntax for removing a native data source follows:

```
java -jar admin_client.jar uri adminId adminPassword -removeNativeDataSource
-name name [-applicationName applicationName]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-removeNativeDataSource -name ScottDataSource
```

*Table 6–35 -removeNativeDataSource Command Parameters*

| Parameter | Description |
|---|---|
| `-name` | Required. The name of the data source to remove. |
| `-applicationName` | Optional. The name of the application from which to remove the data source. |

### Testing a Database Connection

Use the `-testDatabaseConnection` command to test an application's connection to a database in an OC4J instance or in each OC4J instance of a group within a cluster.

The syntax for testing a database connection follows:

```
java -jar admin_client.jar uri adminId adminPassword -testDatabaseConnection
-sqlStatement sqlStatement -factoryClass factoryClass -dbUser dbUser
-dbPassword dbPassword -url url [-applicationName applicationName]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-testDatabaseConnection -sqlStatement "select * from dual"
-factoryClass oracle.jdbc.pool.OracleDataSource -dbUser scott
-dbPassword tiger -url jdbc:oracle:thin:@localhost:1521:xe
-applicationName default
```

*Table 6–36 -testDatabaseConnection Command Parameters*

| Parameter | Description |
|---|---|
| `-sqlStatement` | Required. The SQL statement to use to test the connection. |
| `-factoryClass` | Required. The JDBC factory to test (instance of `Driver`, `DataSource`, `ConnectionPoolDataSource`, or `XADataSource`). |

*Table 6–36 (Cont.) -testDatabaseConnection Command Parameters*

| Parameter | Description |
|---|---|
| `-dbUser` | Required. The user name to use to test the connection. |
| `-dbPassword` | Required. The password to use to test the connection. |
| `-url` | Required. The URL to set on the JDBC factory. |
| `-applicationName` | Optional. The name of the application. |

### Testing a Data Source

Use the `-testDataSource` command to test an application's connection to a data source in an OC4J instance or in each OC4J instance of a group within a cluster.

The syntax for testing a data source follows:

```
java -jar admin_client.jar uri adminId adminPassword -testDataSource
-name name -sqlStatement sqlStatement [-applicationName applicationName]
[-dbUser dbUser] [-dbPassword dbPassword]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-testDataSource -name ScottDataSource -sqlStatement "select * from dual"
-applicationName default -dbUser scott -dbPassword tiger
```

*Table 6–37 -testDataSource Command Parameters*

| Parameter | Description |
|---|---|
| `-name` | Required. The data source to test. |
| `-sqlStatement` | Required. The SQL statement to use to test the connection. |
| `-applicationName` | Optional. The name of the application. |
| `-dbUser` | Optional. The user to use to use to test the connection. |
| `-dbPassword` | Optional. The password to use to use to test the connection. |

### Listing Data Sources

Use the `-listDataSources` command to view a list of data sources that are configured for an application. The list includes each data source's configured properties.

The syntax for listing data sources follows:

```
java -jar admin_client.jar uri adminId adminPassword
 -listDataSources [-applicationName applicationName]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost:23791
 oc4jadmin oc4j -listDataSources -applicationName default
```

*Table 6–38 -listDataSources Command Parameters*

| Parameter | Description |
|---|---|
| `-applicationName` | Optional. The name of the application for which to list configured data sources. The `default` application's data sources are listed if no application name is specified. |

### Getting the Data Sources Descriptor for an Application

Use the -getDataSourcesDescriptor command to retrieve an application's data sources descriptor. The syntax for getting a data sources descriptor follows:

```
java -jar admin_client.jar uri adminId adminPassword -getDataSourcesDescriptor
[-applicationName applicationName]
```

***Table 6–39 -getDataSourcesDescriptor Command Parameter***

| Parameter | Description |
|---|---|
| -applicationName | Optional. The name of the application to which the data sources descriptor belongs. |

# Managing JMS Resources

You can use the admin_client.jar utility to manage JMS resources in an OC4J instance or in a group of OC4J instances, as the following topics describe:

- Managing JMS Connection Factories
- Managing JMS Destinations

## Managing JMS Connection Factories

You can use the admin_client.jar utility to manage the OC4J JMS connection factories, as the following topics describe:

- Adding a JMS Connection Factory
- Removing a JMS Connection Factory
- Getting Information About JMS Connection Factories

### Adding a JMS Connection Factory

Use the -addJMSConnectionFactory command to add a JMS connection factory to an OC4J instance or to each instance of a group within a cluster. The syntax for this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -addJMSConnectionFactory
-domain domain -jndiLocation jndiLocation [-host host] [-port port]
[-username username] [-password password] [-clientID clientID] [-isXA true|false]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-addJMSConnectionFactory -domain Queue -jndiLocation jms/ExampleQueueCF
```

***Table 6–40 -addJMSConnectionFactory Command Parameters***

| Parameter | Description |
|---|---|
| -domain | Required. The JMS domain of this connection factory (`QUEUE', `TOPIC', or `UNIFIED'). |
| -jndiLocation | Required. The JNDI location to which this connection factory will be bound. |
| -host | Optional. The host name associated with this connection factory (defaults to the containing OC4J JMS server host). |
| -port | Optional. The port number associated with this connection factory (defaults to the containing OC4J JMS server port). |

*Table 6–40 (Cont.) -addJMSConnectionFactory Command Parameters*

| Parameter | Description |
| --- | --- |
| `-username` | Optional. The user name associated with this connection factory (defaults to `anonymous`). |
| `-password` | Optional. The password associated with this connection factory (defaults to null). |
| `-clientID` | Optional. The JMS client ID associated with this connection factory (defaults to null). |
| `-isXA` | Optional. Whether or not this is an XA connection factory (defaults to `false`). |

### Removing a JMS Connection Factory

Use the `-removeJMSConnectionFactory` command to remove a JMS connection factory from an OC4J instance or instances. The syntax for this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -removeJMSConnectionFactory
-jndiLocation jndiLocation
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-removeJMSConnectionFactory -jndiLocation jms/ExampleQueueCF
```

*Table 6–41 -removeJMSConnectionFactory Command Parameter*

| Parameter | Description |
| --- | --- |
| `-jndiLocation` | Required. The JNDI location of the connection factory to remove. |

### Getting Information About JMS Connection Factories

Use the `-getJMSConnectionFactories` command to return the attributes for each of the JMS connection factories in an OC4J instance or in a group of OC4J instances within a cluster. The syntax for this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -getJMSConnectionFactories
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-getJMSConnectionFactories
```

## Managing JMS Destinations

You can use the `admin_client.jar` utility to manage the OC4J JMS destinations, as the following topics describe:

- Adding a JMS Destination
- Removing a JMS Destination
- Getting Information About JMS Destinations

### Adding a JMS Destination

Use the `-addDestination` command to add a JMS destination. The syntax for this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -addDestination
-domain domain -name name -jndiLocation jndiLocation [-persistenceFile
```

```
persistenceFile] [-description description]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-addDestination -domain Queue -name ExampleQueue -jndiLocation jms/ExampleQueue
```

***Table 6–42 -addDestination Command Parameters***

| Parameter | Description |
|---|---|
| `-domain` | Required. The JMS domain of this destination (`` `QUEUE' `` or `` `TOPIC' ``). |
| `-name` | Required. The OC4J JMS provider-specific name of the destination. |
| `-jndiLocation` | Required. The JNDI location to which this destination will be bound. |
| `-persistenceFile` | Optional. The persistence file associated with this destination (defaults to null). |
| -description | Optional. A textual description of this destination (defaults to null). |

### Removing a JMS Destination

Use the `-removeDestination` command to remove a JMS destination from an OC4J instance or from each OC4J instance in a group. The syntax for this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -removeDestination
-name name [-force true|false] [-removePFile true|false]
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-removeDestination -name ExampleQueue -removePFile true
```

***Table 6–43 -removeDestination Command Parameters***

| Parameter | Description |
|---|---|
| `-name` | Required. The OC4J JMS provider-specific name of the destination to remove. |
| `-force` | Optional. Removes the destination regardless of whether messages or consumers exist on it (defaults to `false`). |
| -removePFile | Optional. Removes the persistence file from the file system (defaults to `false`). |

### Getting Information About JMS Destinations

Use the `-getDestinations` command to return the attributes for each of the OC4J JMS destinations from an OC4J instance or from each OC4J instance in a group. The syntax for this command follows:

```
java -jar admin_client.jar uri adminId adminPassword -getDestinations
```

For example:

```
java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1
-getDestinations
```

# Managing OC4J Through a Remote Client

You can use a remote client to manage OC4J after you install the files from the remote Administrative Client Utility, as described in "Downloading and Extracting the Remote Administration Client" on page 6-5. Then you can use `admin_client.jar` through the command-line tool or the JMX Remote API.

## Using admin_client.jar Commands Remotely

After you connect to an OC4J application server target, as explained in "Downloading and Extracting the Remote Administration Client" on page 6-5, you can issue `admin_client.jar` commands from a remote client. Use the same syntax that you would use from within an OC4J instance.

## Connecting to a Remote Oracle Application Server Instance Using JConsole

JConsole is a JMX GUI console included in JDK 5.0. JConsole can connect to any JVM and hook into its running MBeanServer, displaying a series of pages on which various system details such as Thread and Memory usage of the JVM are displayed. JConsole can connect to a local JVM, or it can use the JMX Remote API and connect to a remote JVM.

The Administrative Client Utility distribution contains the libraries required to enable JConsole to connect to a remote OC4J or Oracle Application Server instance. To connect to the target instance, the JConsole utility (which is provided as a native executable in a Windows environment) needs to be configured with the relevant details of the Administrative Client Utility distribution.

**To connect to an Oracle Application Server instance:**

1. Add `/j2ee/`*`instance`*`/admin_client.jar` to the CLASSPATH environment variable:

   ```
   set CLASSPATH=j2ee/home/admin_client.jar
   ```

2. Add the JConsole libraries to the CLASSPATH environment variable:

   ```
   set CLASSPATH=%CLASSPATH%;%JAVA_HOME%\lib\jconsole.jar
   set CLASSPATH=%CLASSPATH%;%JAVA_HOME%\lib\tools.jar
   ```

3. Configure the JMX connector to use the OC4J ORMI protocol:

   ```
   set PROPS=jmx.remote.protocol.provider.pkgs=oracle.oc4j.admin.jmx.remote
   ```

4. Run jconsole:

   ```
   %JAVA_HOME%\bin\jconsole
   -J-Djava.class.path=%CLASSPATH%
   -J-D%PROPS%
   ```

   This will launch JConsole.

5. On the Advanced tab of the Connect to Agent screen, enter the connect string for the OC4J or Oracle Application Server target as well as the administration user name and password for the target.

   The pattern of the JMX URL is different for OC4J targets from the pattern for Oracle Application Server targets. Table 6–44 shows examples of these URL patterns.

*Table 6–44 JMX URLs for OC4J and Oracle Application Server Targets*

| Target | JMX URL |
|---|---|
| Standalone OC4J Server | `service:jmx:rmi://test-cycle.oracle.com:23791` |
| OC4J Instance on Oracle Application Server | `service:jmx:ormi:///opmn://test-cycle.oracle.com:6010/test1` |
| Oracle Application Server Cluster | service:jmx:rmis:///opmn://stadp69:6003/cluster/as101/admin |

6. The JConsole utility will show the OC4J MBeans from the target instance. These MBeans can be used to view and manage the configuration of the OC4J instance.

In a Windows environment, the environment used by JConsole can be modified by using a special System property form:

```
-J-Dname=value
```

A sample command script follows:

```
setlocal

set URL=service:jmx:rmi:///opmn://test-cycle.oracle.com:6010/testunit

set JAVA_HOME=C:\java\jdk150_07

set JCONSOLE_CP
set JCONSOLE_CP=%JCONSOLE_CP%;%JAVA_HOME%\lib\jconsole.jar
set JCONSOLE_CP=%JCONSOLE_CP%;%JAVA_HOME%\lib\tools.jar

set ORACLE_HOME=D:\oc4j_admin_client
set ORACLE_CP=
set ORACLE_CP=%ORACLE_CP%;%ORACLE_HOME%\j2ee\home\admin_client.jar;

set CLASSPATH=%JCONSOLE_CP%;%ORACLE_CP%
set PROPS=
set PROPS=%PROPS%
-J-Djmx.remote.protocol.provider.pkgs=oracle.oc4j.admin.jmx.remote

set PROPS=%PROPS% -J-Djava.class.path=%CLASSPATH%

jconsole %PROPS% %URL%

endlocal
```

## Using a JMX Programmatic Client to Manage OC4J Remotely

The Administrative Client Utility distribution provides a full client environment for JMX client applications to connect to remote OC4J instances. You can use a JMX programmatic client to manage OC4J remotely through the JMX Remote API (JSR160), which can establish a connection to the MBeanServer. The only JAR files you need to run with JDK 5.0 are `oc4jclient.jar` and `admin_client.jar`, which the Administrative Client Utility distribution provides.

The following example uses these JAR files with the JMX API:

```
// A URL is of the form "service:jmx:rmi://127.0.0.1:23791"
             JMXServiceURL serviceURL = new JMXServiceURL(_url);

             Hashtable credentials = new Hashtable();
              credentials.put("login", _username);
              credentials.put("password", _password);

             // Properties required to use the OC4J ORMI protocol
             Hashtable env = new Hashtable();
             env.put(JMXConnectorFactory.PROTOCOL_PROVIDER_PACKAGES,
 "oracle.oc4j.admin.jmx.remote");
             env.put(JMXConnector.CREDENTIALS, credentials);
             JMXConnector jmxCon =
 JMXConnectorFactory.newJMXConnector(serviceURL, env);
             jmxCon.connect();

             MBeanServerConnection mbeanServer =
 jmxCon.getMBeanServerConnection();
```

In JDK 5.0 this code compiles with no Oracle libraries required, just the libraries provided by the JDK:

```
clear
 @echo off
 @setlocal

 set J2EE_HOME=c:\java\oc4j-1013-prod\j2ee\home
 set JAVA_HOME=c:\java\jdk50
 set CLASSPATH=.

 rem
 rem Uncomment below if using JDK14
 rem set CLASSPATH=%CLASSPATH%;%J2EE_HOME%\lib\jmxri.jar
 rem set CLASSPATH=%CLASSPATH%;%J2EE_HOME%\lib\jmx_remote_api.jar
 rem set CLASSPATH=%CLASSPATH%;%J2EE_HOME%\lib\javax77.jar
 rem

 %JAVA_HOME%\bin\javac -classpath %CLASSPATH% -d . *.java
 @endlocal
```

To run the code with the `oc4j_admin_client_101350.zip` distribution:

1. Create a runnable JAR file.
2. Drop the JAR file into the `j2ee/home` directory of the Administrative Client Utility distribution.
3. Connect to a remote OC4J instance.

The code runs in JDK 5.0 with `$ORACLE_HOME/j2ee/home/oc4jclient.jar` and `$ORACLE_HOME/j2ee/home/admin_client.jar`:

```
@echo off
@setlocal
clear
set J2EE_HOME=c:\java\oc4j-1013-prod\j2ee\home
set JAVA_HOME=c:\java\jdk50

rem Runtime classpath
set CLASSPATH=.
set CLASSPATH=%CLASSPATH%;%J2EE_HOME%\oc4jclient.jar;
```

```
set CLASSPATH=%CLASSPATH%;%J2EE_HOME%\admin_client.jar;

rem
rem Uncomment if using JDK14
rem set CLASSPATH=%CLASSPATH%;%J2EE_HOME%\lib\jmxri.jar
rem set CLASSPATH=%CLASSPATH%;%J2EE_HOME%\lib\jmx_remote_api.jar
rem set CLASSPATH=%CLASSPATH%;%J2EE_HOME%\lib\javax77.jar
@endlocal
```

The connection URL in the main method of the example is set to connect to a local OC4J instance. If you want to connect to Oracle Application Server through an ORMI port, use a Service URL of the following form:

```
service:jmx:rmi|ormi:///opmn://stadp57.us.oracle.com:6003/home
```

A service URL will obtain the ORMI port from the OPMN daemon. The ORMI port is assigned at runtime. Using the OPMN connection string path will connect you to the specified OC4J instance.

For more information about how to use a JMX client to manage OC4J instances remotely, see "Remote Management Using the JMX Remote API (JSR-160)" in the *Oracle Containers for J2EE Developer's Guide*.

# 7

# Using the admin.jar 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 OC4J installation. Among other things, you can use this utility to stop and restart OC4J, deploy applications, and gather information on current resource usage.

This chapter includes the following topics:

- Overview of admin.jar Usage
- Managing a Standalone OC4J Instance
- Deploying or Undeploying Applications
- Managing Applications
- Managing Data Sources
- Deploying or Undeploying Connectors

---

**Note:** The OC4J web-site-related options (accessible with the `-site` command) that were provided in the `admin.jar` utility in previous releases are no longer available. For information on how to create and manage OC4J Web site configurations, see Chapter 13, "Managing Web Sites in OC4J".

---

## Overview of admin.jar Usage

The `admin.jar` utility is installed by default in `*ORACLE_HOME*/j2ee/home` in a standalone OC4J instance.

OC4J must be started before this utility can be used, except for converting data sources, as "Converting Existing Data Sources to the New Configuration" on page 7-10 describes. Also, the utility cannot be used to start OC4J, although it can be used to stop and then restart an instance, as "Stopping and Restarting OC4J in a Standalone Environment" on page 7-3 describes.

This section covers the following topics:

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

> **Note:** The `admin.jar` utility can be used only to manage a single OC4J instance in a standalone OC4J installation.
>
> Use Oracle Process Manager and Notification Server (OPMN) to manage OC4J instances running as components of Oracle Application Server.
>
> Due to its more advanced capabilities, the `admin_client.jar` utility should be used instead of `admin.jar`. See Chapter 6, "Using the admin_client.jar Utility" for details on using this utility.

## Understanding the admin.jar Syntax

The `admin.jar` utility uses the following syntax. The parameters are described in Table 7–1.

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

As an example, the following command will force a graceful shutdown of the OC4J server. The value supplied for *`oc4jOrmiPort`* is the default, `23791`. The user name supplied for *`adminId`* is the user name for the default administrator account, `oc4jadmin`.

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

Some of these commands include an `-application` parameter that takes the name of the application to affect. The value is the name of the specific application to affect, as defined within the appropriate `<application>` element in the `server.xml` configuration file.

***Table 7–1 Setting the Host and Login Information***

| Parameter | Description |
|---|---|
| *`oc4jHost`*:*`oc4jOrmiPort`* | The host name and port number for the OC4J server on which you are invoking `admin.jar`.<br><br>The `admin.jar` tool uses the OC4J Remote Method Invocation (ORMI) protocol to communicate with the OC4J server. Therefore, the host and port identified by these variables are defined in the `rmi.xml` file for the OC4J server to which you are directing the request.<br><br>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:<br><br>`<rmi-server port="oc4jOrmiPort" host="oc4jHost" />` |
| *`adminId adminPassword`* | The OC4J administration user name and password. The user name for the default administrator account is `oc4jadmin`. |

## Printing Help Text to the Console

To print the online 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 password -help
```

# Managing a Standalone OC4J Instance

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 and 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 password -shutdown force
  need_to_reboot_host_machine
```

***Table 7–2 Options for OC4J Server Shutdown and Restart***

| Option | Description |
|---|---|
| `-shutdown` | Shuts down the OC4J server.<br>`[ordinary|force]`:<br>The type of shutdown. The default is `ordinary`, which allows each thread to terminate normally. The `force` option terminates all threads immediately.<br>`[reason]`:<br>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`.<br>`[reason]`:<br>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

You can force OC4J to check the server directory structure for modified files and reload any that have changed, using the `-updateConfig` option.

> **Note:** The value of the `checkForUpdates` flag must be set to either `all` or `adminClientOnly` (the default setting) to use this feature. See *Oracle Containers for J2EE Deployment Guide* for details on the `checkForUpdates` flag.

*Table 7–3 Option for Checking for Updated Files*

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

## Deploying or Undeploying Applications

You can use `admin.jar` to deploy or undeploy J2EE applications to or from a standalone OC4J instance.

---

**Notes:**

- `admin.jar` cannot be used to deploy applications to an OPMN-managed OC4J instance.
- `admin.jar` supports deployment of EAR files only. It does not allow deployment of standalone modules, such as a Web module packaged in a WAR file.
- `admin.jar` does not accept a deployment plan. Any archive deployed using this utility must include the required OC4J-specific deployment descriptor files, such as `orion-application.xml` or `orion-web.xml`.

---

Deploying an application is a two-step process: You must first deploy the archive to 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 to OC4J:

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

*Table 7–4 Options for Application Deployment*

| Option | Description |
|---|---|
| `-deploy` | Deploys an application. Supply relevant information using the following parameters:<br><br>`-file` *`filename`*:<br>Required. The path and file name of the EAR file to deploy.<br><br>`-deploymentName` *`appName`*:<br>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.<br><br>`-targetPath` *`path`*:<br>Optional. The path on the server node to deploy the EAR to. If not specified, the EAR is deployed to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications` directory by default.<br><br>The deployed EAR file is also copied to this directory. Each successive deployment will cause this EAR file to be overwritten.<br><br>`-parent` *`appName`*:<br>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.<br><br>`-deploymentDirectory` *`path`*:<br>Optional. The directory containing the OC4J-specific deployment descriptors and generated files, such as compiled JSP classes and EJB wrapper classes.<br><br>The default directory is *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application-deployments`.<br><br>`-iiopClientJar` *`path/filename`*:<br>Optional. Include to generate IIOP stubs for the home, remote and local interfaces packaged within each EJB JAR included in the EAR.<br><br>You can optionally specify the path and file name of the JAR to output the generated stubs to. Otherwise, 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/`.<br><br>Note that the `GenerateIIOP` system property must be enabled at OC4J startup to use this feature. For example:<br><br>`java -DGenerateIIOP=true -jar oc4j.jar` |

*Table 7–4 (Cont.) Options for Application Deployment*

| Option | Description |
|---|---|
| `-bindWebApp` | Binds a Web application to the specified Web site and context root.<br>■ *appName*: The application name, which is the same name set as the value for `-deploymentName` in the `-deploy` option.<br>■ *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.<br>■ *webSiteName*: The name of the *name*`_web-site.xml` file that denotes the Web site that this Web application should be bound to.<br>■ *contextRoot*: The context root for the Web module. This value will be appended to the URL used to access the application through a Web browser; for example `http://localhost:8888/utility`.<br>This option creates an entry in the *name*`-web-site.xml` configuration file that was denoted in the *web_site_name* variable. |
| `-undeploy` *appName* | Removes the deployed J2EE application from the OC4J instance. The value of *appName* is the name of the application within OC4J, as defined in an *application* element within *ORACLE_HOME*`/j2ee/home/config/server.xml`.<br>Undeploying an application results in the following:<br>■ The application is removed from the OC4J runtime and the `server.xml` file.<br>■ Bindings for all the application's Web modules are removed from all the Web sites to which the Web modules were bound.<br>■ Application files are removed from both the `applications` and `application-deployments` directories.<br>The optional `-keepFiles` parameter is deprecated. |

# Managing Applications

This section outlines the functionality provided by `admin.jar` for managing applications in a standalone OC4J instance. It includes the following sections:

- Starting, Stopping, or Restarting an Application
- Updating an EJB Module Within an Application

## Starting, Stopping, or Restarting an Application

You can use `admin.jar` to start, stop, or restart an application that has been stopped in a standalone 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 password -application
   myapplication -restart
```

*Table 7–5 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 and any child applications.<br>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 enables incremental or partial redeployment of EJB modules within an application running in an OC4J instance. This option is primarily intended to be used by an application developer to redeploy the JAR file directly from a development environment.

> **Note:** Incremental redeployment may be more efficient than redeploying the entire application for CMP or BMP entity beans but not for session beans, message-driven beans, or EJB 3.0 JPA entities. For details about whether to use this feature, see "Incremental Redeployment of Updated EJB Modules" in the *Oracle Containers for J2EE Deployment Guide*.

The syntax 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 descriptor, the following command could be issued from the `/dist` directory:

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

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

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

*Table 7–6 Options for Updating an EJB Module*

| Option | Description |
|---|---|
| `-application` *`appName`* `-updateEJBModule` | Updates the specified EJB module with new EJB modules.<br>■ *`relativePath`*:<br>The relative path to the EJB JAR containing the updated beans as defined in the application's `application.xml` J2EE deployment descriptor.<br>■ `-file` *`path`*:<br>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 Data Sources

Use `admin.jar` to create, remove, list or test data sources for a specific application. You can also convert a pre-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]
[-xaLocation jndiName] [-ejbLocation jndiName] -username name
-password password [-connectionDriver className] -className className
[-sourceLocation jndiName][-xaSourceLocation jndiName]
```

An example follows:

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin password -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 7–7 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 parameters:<br><br>`-jar` *`path`*:<br>Required. The path to the JAR file containing the JDBC driver that is to be added to the OC4J server.<br><br>`-url` *`url`*:<br>Required. The JDBC database URL.<br><br>`-location` *`jndiName`*:<br>Required. The JNDI name for the raw data source. For example, `"jdbc/DefaultPooledDS"`.<br><br>`-pooledLocation` *`jndiName`*:<br>Optional. The JNDI name for the pooled data source. For example, `"jdbc/DefaultPooledDS"`.<br><br>`-xaLocation` *`jndiName`*:<br>Optional. The JNDI name for the XA source. For example, `"jdbc/xa/DefaultXADS"`. Required if `-ejbLocation` is specified.<br><br>`-ejbLocation` *`jndiName`*:<br>Optional. The JNDI name for the container-managed transactional data source. This is the only data source that can perform global JTA transactions. For example, `"jdbc/DefaultDS"`.<br><br>`-username` *`name`*:<br>Required. The user name to log in to the database.<br><br>`-password` *`password`*:<br>Required. The password to log in to the database.<br><br>`-connectionDriver` *`className`*:<br>Optional. The JDBC database driver class.<br><br>`-className` *`className`*<br>Required. The data source class name, such as `oracle.jdbc.pool.OracleDataSource`.<br><br>`-sourceLocation` *`jndiName`*:<br>Optional. The JNDI name of the underlying data source of this specialized data source.<br><br>`-xaSourceLocation` *`jndiName`*:<br>Optional. The JNDI name of the underlying XA data source of this specialized data source. |

## Listing, Testing, and Removing Existing Data Sources

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

*Table 7–8 Options for Application and Data Source Management*

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

*Table 7–8 (Cont.) Options for Application and Data Source Management*

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

## Converting Existing Data Sources to the New Configuration

The OC4J 10*g* (10.1.3.5.0) implementation understands the 10.1.3 and the pre-10.1.3 (10.1.2 and 9.0.4) formats of the `data-sources.xml` file. For an application that was used in a pre-10.1.3 OC4J implementation and contains its own `data-sources.xml` file, the OC4J 10*g* (10.1.3.5.0) implementation automatically converts the `data-sources.xml` file from the pre-10.1.3 format to the 10.1.3 format when you use Application Server Control to change anything in the `data-sources.xml` file, such as modifying an existing data source or creating or deleting a data source.

### Converting a data-sources.xml File with Standalone OC4J Running or Not Running

The `-convertDataSourceConfiguration` option of the `admin.jar` command converts a pre-10.1.3 `data-sources.xml` file to the new file format.

With an active OC4J instance in a standalone environment, you can use `admin.jar` with the following syntax to manually convert a pre-10.1.3 `data-sources.xml` file to the 10.1.3 format.

```
java -jar admin.jar ormi://oc4jHost:oc4jOrmiPort adminId adminPassword
-convertDataSourceConfiguration old-data-sources.xml new-data-sources.xml
```

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

```
java -jar admin.jar ormi://localhost:23791 oc4jadmin password
-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 might be needed for reference. After the *new* file has been generated, copy it into the directory containing the legacy file.

In the syntax, the ORMI URL is optional. You can specify an ORMI URL only when OC4J is running.

You can also convert a `data-sources.xml` file before deployment, without a running OC4J instance. The syntax for this offline conversion is as follows:

```
java -jar admin.jar -convertDataSourceConfiguration
old-data-sources.xml new-data-sources.xml
```

---

**Notes:**

- If you include the ORMI port, then OC4J must be running. When OC4J is not running, you must omit the ORMI URL from the `admin.jar` command line.
- If you do not include the ORMI port, then the `admin.jar` command will work either way; that is, with OC4J running or with OC4J not running.
- The `admin.jar` utility works only in a standalone OC4J environment. This utility is installed in the Oracle Application Server environment but does not work in an OPMN-managed environment.
- The newer `admin_client.jar` utility works in both environments, standalone and managed Oracle Application Server. However, the `admin_client.jar` utility does not convert `data-sources.xml` files.

---

### Checking Consistency Between the Application and the New data-sources.xml File

After conversion, whether manual or automatic, visually inspect the new `data-sources.xml` file to confirm that there is consistency between your application and the new file regarding the JNDI location used to refer to a data source.

This consistency check is advisable because the new file may contain data source definitions that are not used, which happens because the old format uses multiple location attributes (such as `location`, `ejb-location`, and `xa-location`). The conversion to the new 10.1.3 format creates a separate data source in the new `data-sources.xml` file corresponding to each location attribute specified in the old `data-sources.xml` file. In most cases, client applications will use only the data source defined by either the `location` or `ejb-location` attribute. The converted `data-sources.xml` file may have definitions that are not used by the applications and can be removed from the file.

For examples of the new `data-sources.xml` format, see the "Data Sources" chapter of the *Oracle Containers for J2EE Services Guide*.

## Deploying or Undeploying Connectors

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

***Table 7–9 Options for Application Deployment***

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

*Table 7–9 (Cont.) Options for Application Deployment*

| Option | Description |
|---|---|
| `-undeployconnector` | Undeploys the specified connector.<br><br>`name` *`name`*:<br>The name of the connector to undeploy.<br><br>Undeploying a standalone RAR does not require a restart of the `default` application. |

# 8

# Configuring and Managing Clusters and OC4J Groups

This chapter explains how to configure and manage cluster topologies in an Oracle Application Server environment and groups of OC4J instances within Oracle Application Server clusters. It includes the following topics:

- Clustering Overview
- Creating and Managing OC4J Groups Within Oracle Application Server Clusters
- Configuring a Cluster
- Viewing the Status of a Cluster
- Configuring Routing and Load Balancing with Oracle HTTP Server
- Configuring Application Mount Points
- Running an OC4J Instance on Multiple JVMs

Application clustering, the clustering of applications deployed to Oracle Application Server nodes for the purpose of session or state replication, is covered in Chapter 9, "Application Clustering in OC4J".

## Clustering Overview

This section provides an overview of the clustering mechanisms supported in Oracle Application Server 10*g* Release 3 (10.1.3.5.0) and notes the significant changes in functionality between the 10.1.3 release and previous releases. It includes the following topics:

- How Clustering Works
- Supported Clustering Models
- Changes in Clustering

### How Clustering Works

In the current release, a cluster topology is defined as two or more loosely connected Oracle Application Server nodes.

The connectivity provided within a cluster is a function of Oracle Notification Server (ONS), which manages communications between Oracle Application Server components, including OC4J and Oracle HTTP Server. The ONS server is a component of Oracle Process Manager and Notification Server (OPMN), which is installed by default on every Oracle Application Server host. When configuring a cluster topology, you are actually connecting the ONS servers running on each Oracle Application Server node.

Previous releases of Oracle Application Server supported clustering of a fully connected set of server nodes only, which meant that each node had to be explicitly specified in the ONS configuration file (`ons.conf`). When a node was added or removed from the cluster, the configuration had to be updated on each server node and the server restarted.

The current release supports a new **dynamic discovery** mechanism, enabling the cluster to essentially manage itself. In this framework, each ONS maintains a map of the current cluster topology. When a new ONS is added to the cluster, each existing ONS adds the new node and its connection information to its map. At the same time, the new ONS adds all of the existing nodes to its map. Alternatively, when an ONS is removed from the cluster, the maps for the remaining nodes are updated with this change.

As of Oracle Application Server Release 3 (10.1.3.0.0), the ONS configuration file (`ons.conf`) is no longer used. Instead, ONS configuration data is set in the `<notification-server>` element within `opmn.xml`, the OPMN configuration file located in the *ORACLE_HOME*`/opmn/conf` directory on each node. Clustering configuration in turn is set within a `<topology>` subelement. Only one `<topology>` subelement is allowed within a `<notification-server>` element.

The following example illustrates a cluster topology configuration in `opmn.xml`:

```
<notification-server>
  <topology>
    <discover list="*225.0.0.20:8001"/>
  </topology>
  ...
</notification-server>
```

The clustering configuration specified in the `<topology>` element applies to all instances of Oracle Application Server components, including Oracle HTTP Server and OC4J, installed on the node. All nodes within a cluster topology must have the same configuration specified in the `opmn.xml` file.

## Supported Clustering Models

The following clustering models are supported:

- Dynamic node discovery

  In this configuration, each ONS node within the same subnet announces its presence with a multicast message. The cluster topology map for each node is automatically updated as nodes are added or removed, enabling the cluster to be self-managing.

  See "Configuring Dynamic Node Discovery Using Multicast" on page 8-13 for configuration instructions.

- Static hubs as **discovery servers**

  Specific nodes within a cluster are configured to serve as discovery servers, which maintain the topology map for the cluster. The remaining nodes then connect with one another through a discovery server. A discovery server hub in one topology can be connected to hubs in other topologies.

  See "Configuring Static Discovery Servers" on page 8-16.

- Connection of isolated topologies via gateways

This configuration is used to connect topologies separated by firewalls or on different subnets using specified *gateway* nodes.

See "Configuring Cross-Topology Gateways" on page 8-18 for details.

- Manual node configuration

  In this configuration, the host address and port for each node in the cluster are manually specified in the configuration. This is the same clustering mechanism supported in Oracle Application Server 10*g* Release 3 (10.1.2) and is supported primarily to provide backward compatibility.

  See "Configuring Static Node-to-Node Communication" on page 8-21 for instructions.

## Changes in Clustering

The following are changes in cluster configuration in Oracle Application Server 10*g* Release 3 (10.1.3) from previous releases.

- As of Oracle Application Server 10*g* (10.1.3.1.0), OC4J instances belong to groups within the cluster topology, enabling you to perform group deployment, configuration, and administration operations across an Oracle Application Server cluster.

  In Oracle Application Server 10*g* (10.1.3.5.0), groups are explicitly created by administrators using any desired name. Once a group has been created, it can be populated with any of the OC4J instances that are resident within the cluster topology.

  > **Note:** The procedures for creating and managing groups have changed since Oracle Application Server 10*g* (10.1.3.0.0). If you have been using the 10.1.3.0.0 release, be sure to review the new procedures for creating and managing groups in the 10.1.3.5.0 release, described in "Creating and Managing OC4J Groups Within Oracle Application Server Clusters" on page 8-4.

- The Distributed Configuration Management (DCM) framework, used in prior releases of Oracle Application Server to replicate common configuration information across a cluster, is not included in the current release. This means that:
  - Configuration using the `dcmctl` command-line utility or Application Server Control is no longer supported.
  - Cluster configurations must now be manually replicated in the `opmn.xml` file installed on each node within the cluster.
- The ONS configuration file (`ons.conf`) is no longer used. ONS connection data is now set in the `<notification-server>` element within `opmn.xml`, the OPMN configuration file located in the *`ORACLE_HOME`*`/opmn/conf` directory on each node containing an OC4J or Oracle HTTP Server instance.
- Each node is no longer required to be manually configured to connect to every other node in the cluster.

## Creating and Managing OC4J Groups Within Oracle Application Server Clusters

All OC4J instances in an OPMN-managed environment must be part of a **group**, which is a set of OC4J instances that belong to the same cluster topology. Groups enable you to perform some common configuration, administration, and deployment tasks simultaneously on all OC4J instances in a group.

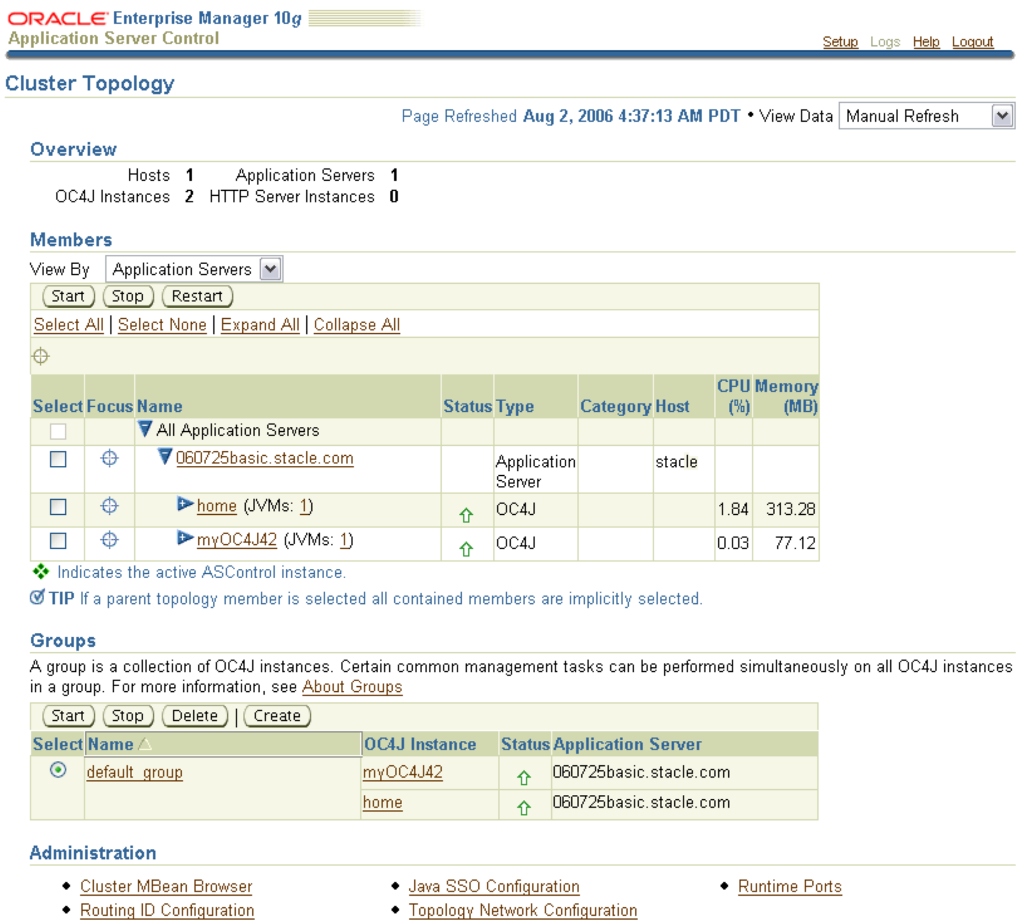
> **Note:** All OC4J instances in a group within an Oracle Application Server cluster must have the same version, such as 10.1.3.5.0.

With Application Server Control, you can create additional groups and, from the Group page, perform the following tasks on a group of OC4J instances:

- Process management operations, such as start, stop, and restart
- Deployment operations, such as deploy, undeploy, and redeploy
- JDBC management operations, such as creating, modifying, or removing JDBC data sources and connection pools
- JMS Provider operations, such as creating and removing JMS destinations, and creating, modifying, or removing JMS connection factories

To display the Group page, click the name of the group in the Groups section of the Cluster Topology page.

***Figure 8–1 Group Section of the Cluster Topology Page***




The default OC4J group (`default_group`) is created automatically when you install an application server instance. When you install Oracle Application Server 10*g* (10.1.3.5.0), the installer creates a default OC4J instance that resides in the default group. Later, you can add OC4J instances and organize them into groups.

For example, you can create a new group for the deployment of a particular application to all OC4J instances of the group across the Oracle Application Server cluster. Then you can use the Group page in Application Server Control to make application-specific configuration changes to all instances of the application in the OC4J group, across the cluster.

In the following topics, this section describes how to create and manage groups of OC4J instances for group operations on applications replicated across one or more Oracle Application Server clusters:

- Creating Groups of OC4J Instances
- Managing OC4J Instances in a Group
- Replicating Changes Across a Cluster
- Running an OC4J Instance on Multiple JVMs

## Creating Groups of OC4J Instances

With groups, you can perform each of the following tasks once across multiple OC4J instances:

- Modify the OC4J server properties for all OC4J instances in a group
- Start or stop all the OC4J instances in a group
- Deploy, undeploy, or redeploy applications on all OC4J instances in a group
- Perform JDBC management operations, such as creating, modifying, or removing JDBC data sources or connection pools
- Perform JMS Provider operations, such as creating or removing JMS destinations and creating, modifying, or removing JMS connection factories

**To administer a group with Application Server Control:**

1. In the Cluster Topology page, under Groups, choose the group.
2. Select the Administration tab.
3. The Administration page provides administration features for the group as a whole. These features do not include Security Provider administration.

**To create a new OC4J group with Application Server Control:**

1. In the Cluster Topology page, under Groups, choose **Create**.
2. In the Create Group page:

   a. Specify a name for the group.

   A group name can contain only alphanumeric characters and underscores and cannot contain any special characters, such as parentheses, periods, dollar signs ($), asterisks (*), or commas. The name must start with a letter or an underscore.

   Table 8–1 lists some examples of valid and invalid names for OC4J instances and groups.

***Table 8–1 OC4J Instance and Group Names***

| Valid Instance or Group Name | Invalid Instance or Group Name |
|---|---|
| OC4J1 | $OC4J_2 |
| _production_apps | 32_PROD_test |
| test_environment_42 | !deployGroup2 |
| Deployment_Group3 | deployment_(group3) |

   b. Select the OC4J instances to move to the group.

   When you move an OC4J instance into the new group, the instance is removed from its previous group. The instance must be stopped before it can be moved.

   c. Choose **Create**.

---

**Note:** You can also move an OC4J instance into a group after the group is created, as follows:

1. In the Cluster Topology page, under Groups, select the group.
2. In the Group: *`groupname`* page, choose **`Add`**.

---

After you create a group, it appears in the list of groups on the Cluster Topology page. You can later add OC4J instances to the group or remove instances from the group, as "Managing OC4J Instances in a Group" on page 8-7 describes.

You can also create a group during the following operations:

- Creating a new OC4J instance

  When you create a new OC4J instance, you can create a new group or identify an existing group for the instance. If you do not specify a group, the new instance is assigned to `default_group`.

- Removing an OC4J instance from a group

  When you remove an OC4J instance from a group, you create a new group or identify an existing group for the instance.

---

**Notes:** The following restrictions apply to moving OC4J instances between groups:

- An OC4J instance must be stopped before you can move it into or out of a group.
- At least one OC4J instance in a group must be running when you move an instance out of the group.
- If a group has only one OC4J instance, before you can move that instance, you must stop it, create another instance, and start the new instance.

---

Consider the following examples of using multiple OC4J instances and groups to manage your Oracle Application Server environment:

- Create OC4J instances for specific purposes. For example, use the default OC4J instance as your administration OC4J and be sure you use it exclusively for deploying Application Server Control. Create another OC4J instance to deploy your production applications.
- Create additional OC4J instances to improve performance and provide load balancing for your production applications.
- Group OC4J instances on which you deploy the same application so you can make application-specific modifications to the group instead of to individual OC4J instances. You can also deploy an application to the group once instead of multiple times to the individual OC4J instances.

## Managing OC4J Instances in a Group

OC4J includes tools for creating additional OC4J instances in a group and removing instances from a group within an Oracle Application Server cluster. Once created, new OC4J instances can be accessed and managed with Application Server Control.

This section includes the following topics:

- Creating an Additional OC4J Instance
- Accessing and Managing a New Instance
- Removing an OC4J Instance from a Group

### Creating an Additional OC4J Instance

You can add an OC4J instance to a group in the following ways:

- Through an Application Server page in Application Server Control
- With the `createinstance` utility, which is installed in the *ORACLE_HOME*/bin directory

**Creating an OC4J Instance Through Application Server Control** To create an OC4J instance through Application Server Control:

1. On the Cluster Topology page, click the name of an Oracle Application Server instance to navigate to an Application Server: `instance_name` page.
2. Click **Create OC4J Instance**.
3. On the Create OC4J Instance page, enter the following information:
   - **OC4J Instance Name:** Enter a name for the instance.

     **Note:** You cannot enter `home` as the instance name because `home` is reserved for the name of the default OC4J instance.

   - Select one of the following items:
     – **Add to an existing group with name:** Select a group from **Existing Group Name**.
     – **Add to a new group with name:** In the **New Group Name** field, enter a name for the new group.
   - Select **Start this OC4J instance after creation**.
4. Click **Create**.

A confirmation screen is displayed after the instance has been created. The password for this OC4J instance is the same as the password used for the `oc4jadmin` user for the installation.

**Creating an OC4J Instance with the createinstance Utility** The `createinstance` utility enables you to create additional OC4J instances in a group with the following syntax:

```
createinstance -instanceName instanceName [-port httpPort] [-groupName group]
```

**Note:** You cannot specify `home` for `instanceName` because `home` is reserved for the name of the default OC4J instance.

You must supply an HTTP listener port as the value for `httpPort` when creating a new instance in a standalone OPMN-managed OC4J instance (J2EE Server and Process Management install type). This HTTP listener port will be set in the `default-web-site.xml` Web site configuration file created for the instance.

Every new OC4J instance is assigned to a group. If the specified group does not exist, it is created. If the `-groupName` parameter is not provided, the instance goes into the `default_group` group.

As part of the creation process, you will be asked to enter a password. This password will be tied to the `oc4jadmin` user for this instance. Oracle recommends that you enter the same password used by the oc4jadmin user to access Application Server Control in the administration instance to prevent problems with OPMN.

As part of the creation operation, the new instance is added to the existing opmn.xml file. To ensure that OPMN is aware of the new instance, an OPMN reload is performed at the end of the create operation. For this reload, the createinstance utility must connect to the MBeanServer used to configure OPMN. The password of the new OC4J instance is used for authentication. If the password of the new instance is not the same as the instance running the MBeanServer, an error is returned. This does not prevent the instance from being created, but it does cause problems when OPMN or other components need to connect to the new instance. Therefore, Oracle recommends that you create all OC4J instances in the target Oracle Application Server cluster with the same password.

You also need to specify the same password for the oc4jadmin user in each OC4J instance of a group within an Oracle Application Server cluster so the user can perform group operations.

**Notes:**

- You can use the createinstance utility regardless of whether the Oracle Application Server instance is in a running state or stopped state.
- If the new OC4J instance will be required to accept ORMI over SSL (ORMIS) requests, you must configure ORMIS in the instance-specific rmi.xml file and update opmn.xml with the ORMIS port information, as described in the *Oracle Containers for J2EE Security Guide*.

You can supply an HTTP port for the value of -port. This feature is required when the Oracle Application Server instance does not include Oracle HTTP Server. Setting an HTTP port makes it possible to access the OC4J instance's home page directly.

The new instance will be created within a new *ORACLE_HOME*/j2ee/*instance* directory, the same location as the default home OC4J instance. A new <process-type> element containing the instance configuration will also be added to the opmn.xml configuration file.

The <process-type> element for the home instance serves as a template for the new OC4J instance, which is created with the same settings as the home instance. You can change the configuration for the new instance by changing settings such as the heapsize, numprocs, and timeout attribute values of the <process-type> element for the instance in the opmn.xml file. If you change the configuration for an OC4J instance, you need to restart it for the changes to take effect.

The following directories and files are generated in the new *ORACLE_HOME*/j2ee/*instance* directory structure:

```
applib/
applications/
config/
  contains default versions of all server-level configuration files
config/database-schemas/
  contains all database schema XML files packaged with OC4J
connectors/
  contains RAR files packaged with OC4J
log/
persistence/
```

The new instance does not include the OC4J binary libraries; instead, the instance will utilize the libraries installed in the `home` instance. The `default` application is deployed to the new instance; however, binaries and configuration files for other deployed applications, including Application Server Control, are not copied to the instance.

### Accessing and Managing a New Instance

Once the new instance is started by OPMN, you can access it through the Cluster Topology page in Application Server Control.

Log in as the `oc4jadmin` user and supply the password set when the instance was created using the `createinstance` utility.

Once logged in, you can perform the full range of administrator tasks on the instance, including deploying applications to it.

### Removing an OC4J Instance from a Group

You can remove an OC4J instance from a group by moving it to another group, as described in "Creating Groups of OC4J Instances" on page 8-6, or by deleting it. You can delete an OC4J instance in the following ways:

- In Application Server Control, through the Application Server Page for Oracle Application Server on which the OC4J instance is installed
- With the `removeinstance` utility, which is installed in the *`ORACLE_HOME`*`/bin` directory

Both methods delete the directory created for the instance from the `j2ee` directory structure and remove configuration data for the instance from `opmn.xml`. The following guidelines apply to deleting an OC4J instance.

- You cannot delete the OC4J `home` instance that was created by Oracle Application Server during installation.
- You can delete OC4J instances that were created by a user after installation.
- The OC4J instance to be deleted must be in a stopped state (which Application Server Control does for you).
- If OPMN is running when the `removeinstance` tool is in use, you must invoke `opmnctl reload` to reload the updated `opmn.xml` into the runtime.

**Deleting an OC4J Instance Through Application Server Control** To delete an OC4J instance through Application Server Control:

1. On the Cluster Topology page, click the name of the Oracle Application Server instance where the OC4J instance is running to navigate to the Application Server: `instance_name` page.
2. Click the **Delete** icon for the instance you want to delete.
3. On the confirmation page, click **Yes**.

A confirmation screen is displayed after the instance has been deleted.

**Deleting an OC4J Instance with the removeinstance Utility** You can delete an OC4J instance by using the `removeinstance` utility, which deletes the directory created for the instance from the *`ORACLE_HOME`*`/j2ee` directory structure and removes configuration data for the instance from `opmn.xml`.

The `removeinstance` utility is installed in the *ORACLE_HOME*`/bin` directory. The syntax is as follows:

```
removeinstance -instanceName instanceName
```

To delete an instance with the utility, take the following steps:

1. Stop the instance:

   ```
   ORACLE_HOME/opmn/bin/opmnctl stopproc process-type=oc4J_instanceName
   ```

2. Delete the instance:

   ```
   ORACLE_HOME/bin/removeinstance -instanceName oc4J_instanceName
   ```

## Replicating Changes Across a Cluster

Because the Distributed Configuration Management (DCM) framework is not provided in Oracle Application Server Release 3 (10.1.3), configuration file synchronization within a cluster has changed in Oracle Application Server 10.1.3. Table 8–2 summarizes the files that might need to be replicated.

Using the OC4J grouping feature introduced in release 10.1.3.1.0 (described in "Creating Groups of OC4J Instances" on page 8-6), it is possible to deploy EARs, WARs, RARs, and shared libraries consistently across groups of OC4J instances using Application Server Control, the `admin_client.jar` command-line utility, or OC4J Ant tasks. This ensures consistent configuration at a module level within groups of OC4J instances. For information about deploying to groups of OC4J instances using these tools, see Chapter 6, "Using the admin_client.jar Utility," and the *Oracle Containers for J2EE Deployment Guide*.

For specific configuration files, the group feature also enables administrators to configure data sources, connection pools, and JMS resources across groups of OC4J instances from Application Server Control, the `admin_client.jar` command-line utility, and OC4J Ant tasks. Specifically, the configuration files that support this are `data-sources.xml` and `jms.xml`.

To achieve consistent configuration across multiple OC4J processes, you can use the multiple JVM feature of Oracle Application Server. This feature enables you to set the number of JVM instances, *n*, on which a single OC4J configuration will run simultaneously. The result is that from a single consistent configuration set, *n* OC4J process running the same OC4J instance will be started. Changing any file in that single configuration set will update all the OC4J processes that started, corresponding to the number of JVMs set. Configuring the number of JVMS per OC4J instance is covered in "Running an OC4J Instance on Multiple JVMs" on page 8-31.

Beyond these specific features, Table 8–2 summarizes the complete set of configuration files and their usage in case manual configuration across a cluster is determined to be necessary for an application configuration change.

*Table 8–2 Configuration Files to Replicate Across a Cluster*

| File | Location in *ORACLE_HOME* | Data to Replicate or Manage |
|---|---|---|
| `application.xml` | `/j2ee/`*instance*`/config` | ■ Changes made to configuration data applied by default to all deployed applications.<br>■ References to data sources or other shared resources.<br>■ Shared library definitions within the `<imported-shared-libraries>` element.<br>Code sources for custom shared libraries must be installed on the OC4J host, and the libraries must be referenced in the `server.xml` file for the OC4J instance. |
| `data-sources.xml` | `/j2ee/`*instance*`/config` | ■ Configuration data for custom data sources that must be made available to deployed applications. |
| `default-web-site.xml` | `/j2ee/`*instance*`/config` | ■ Secure Web site (HTTPS) configuration, if applicable. |
| `*-web-site.xml` | `/j2ee/`*instance*`/config` | ■ Copy the configuration files for any additional Web sites that will be utilized on the OC4J instance to the specified location.<br>Note that references to Web site configuration files must be added to `opmn.xml` or `server.xml`, as outlined in "Creating a New Web Site in OC4J" on page 13-6. |
| `global-web-application.xml` | `/j2ee/`*instance*`/config` | ■ Any new servlet definitions or servlet configuration changes, such as `<init-param>` modifications.<br>■ Any modified JSP container properties. See the *Oracle Containers for J2EE Support for JavaServer Pages Developer's Guide* for details. |
| `j2ee-logging.xml` | `/j2ee/`*instance*`/config` | ■ Any logging configuration changes. |
| `javacache.xml` | `/j2ee/`*instance*`/config` | ■ Any Java cache configuration changes. |
| `jazn.xml` | `/j2ee/`*instance*`/config` | ■ Configuration for either XML-based or LDAP-based security providers. For more information about the `jazn.xml` file, see the *Oracle Containers for J2EE Security Guide*. |
| `jazn-data.xml` | `/j2ee/`*instance*`/application-deployments/`*appName* | ■ Replicate the XML-based provider configuration to the specified location for all applications using this provider. Not required for applications using an LDAP-based provider. For more information about the `jazn-data.xml` file, see the *Oracle Containers for J2EE Security Guide*. |
| `jms.xml` | `/j2ee/`*instance*`/config` | ■ Any destination or connection factory additions. |

*Table 8–2 (Cont.) Configuration Files to Replicate Across a Cluster*

| File | Location in *ORACLE_HOME* | Data to Replicate or Manage |
|---|---|---|
| rmi.xml | /j2ee/*instance*/config | ■ Any RMI configuration changes, such as logging configuration. |

# Configuring a Cluster

This section contains instructions on configuring the following clustering models:

- Configuring Dynamic Node Discovery Using Multicast
- Configuring Static Discovery Servers
- Configuring Cross-Topology Gateways
- Configuring Static Node-to-Node Communication

## Configuring Dynamic Node Discovery Using Multicast

Dynamic node discovery is the most straightforward clustering configuration. In this model, each ONS node broadcasts a simple multicast message announcing its presence, enabling nodes within the cluster to dynamically *discover* one another.

The following tools can be used to add OC4J instances to a cluster using multicast discovery:

- `opmnctl`

  This utility includes commands for updating `opmn.xml` with the multicast *`port:address`* and Web site configuration data needed to add an instance to a cluster. See "Configuring Multicast Discovery with opmnctl" on page 8-15 for details.

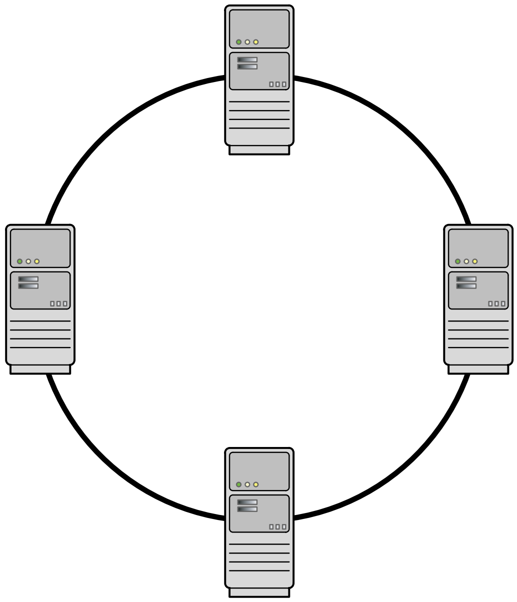
- `opmnassociate`

  This utility provides a one-step solution for adding an OC4J instance to a cluster. See "Configuring Multicast Discovery with opmnassociate" on page 8-16 for details.

> **Note:** An Oracle Application Server instance can be added to a cluster at installation time.

Each ONS maintains its own map of the cluster topology. When a new ONS is added to the cluster, each existing ONS adds the new node and its connection information to its map. At the same time, the new ONS adds all of the existing nodes to its map. Alternatively, when an ONS is removed from the cluster, the maps for the remaining nodes are updated with this change.

***Figure 8–2 Dynamic Discovery Model***



Because multicast messages may be restricted by different network configurations, dynamic node discovery might be an option only for ONS nodes that are on the same subnet. However, multiple subnets using dynamic node discovery may be connected using gateway servers. See "Configuring Cross-Topology Gateways" on page 8-18 for details.

---

**Notes:**

- All nodes within the topology must be configured to use the same multicast address and port.
- The multicast address must be within the valid address range, which is `224.0.1.0` to `239.255.255.255`.

  Ideally, multicast address and port assignments should be managed by your systems administration staff to avoid potential conflicts with other applications.

---

The dynamic discovery configuration is set within a `<discover>` subelement of the `<topology>` element in the `opmn.xml` file on each Oracle Application Server instance in the topology. To add a new node to the cluster, simply add this element to its `opmn.xml` file. To remove a node from the cluster, remove this element.

Set the multicast IP address and port as the value for the `list` attribute. The asterisk (`*`) preceding the IP address is critical because it informs OPMN that the value specified is a multicast address. Multiple values can be specified, each separated from the next by a comma.

```
<opmn>
 <notification-server>
  <port ...  />
  <ssl ... />
  <topology>
    <discover list="*225.0.0.20:8001"/>
  </topology>
 </notification-server>
```

```
 ...
</opmn>
```

**Note:** The `opmn.xml` file must be reloaded for changes made to take effect. Run the following command on the affected node to reload `opmn.xml`:

```
opmnctl reload
```

This command will not affect OPMN-managed components, including Oracle HTTP Server, OC4J, and deployed applications.

### Configuring Multicast Discovery with opmnctl

The OPMN command-line tool, `opmnctl`, supports a new `config topology` command that enables you to specify, update, or delete the multicast `<discover>` entry within `opmn.xml`.

The `opmnctl` tool is installed in the *ORACLE_HOME*`/opmn/bin` directory on each node. The tool must be run individually on each node and will update only the `opmn.xml` file on that node.

**Note for Adding OPMN-Managed Standalone OC4J Instances:**

An OPMN-managed OC4J instance does not include Oracle HTTP Server (J2EE Server and Process Management install type). The `default` Web site is configured to listen for HTTP requests by default.

When adding the instance to a cluster, you must configure the Web site to use the Apache JServ Protocol (AJP). This modification is necessary to enable the OC4J instance to receive and respond to requests from Oracle HTTP Server.

The protocol and ports used by the default Web site can be configured using the Runtime Ports page in Application Server Control. The `opmnctl config port update` command can also be used to modify the `default` Web site configuration defined in `opmn.xml`. For details, see "Configuring Web Sites with opmnctl" on page 13-5.

#### Inserting or Updating Discovery Data

The `update` command inserts or updates the `<discover>` element with the specified values. The syntax is as follows:

```
opmnctl config topology update discover="*multicastAddress:multicastPort"
```

For example:

```
opmnctl config topology update discover="*225.0.0.20:8001"

opmnctl reload
```

#### Deleting Discovery Data

The `delete` command removes the `<discover>` element from `opmn.xml`, effectively removing the node from the cluster. If the `<topology>` element contains no other subelements, it will be removed as well.

```
opmnctl config topology delete discover

opmnctl reload
```

### Configuring Multicast Discovery with opmnassociate

The `opmnassociate` utility adds the default `home` OC4J instance to a cluster using multicast discovery. This utility performs the following steps:

1. Inserts or updates the `<discover>` element in `opmn.xml` with the specified multicast address and port.
2. Configures the `default` Web site to receive and respond to requests from Oracle HTTP Server using the Apache JServ Protocol (AJP), by modifying the corresponding `<port>` element in `opmn.xml`.
3. Restarts OPMN to load the new configuration into the runtime environment.

The `opmnassociate` tool is installed in the *`ORACLE_HOME`*`/bin` directory on each OC4J instance. The tool must be run individually on each instance and will update only the `opmn.xml` file on that instance.

In a Linux or UNIX environment, the syntax is as follows:

```
opmnassociate.sh "*multicastAddress:multicastPort" [-restart]
```

For example:

```
opmnassociate.sh "*225.0.0.20:8001" -restart
```

In a Windows environment, the syntax is as follows:

```
opmnassociate "*multicastAddress:multicastPort" [-restart]
```

For example:

```
opmnassociate "*225.0.0.20:8001" -restart
```

The asterisk (`*`) preceding the IP address is required.

---

**Note:** You can use the `opmnassociate` utility only to add the default `home` OC4J instance to a cluster. If you want to add another OC4J instance, such as `home2`, use the `opmnctl` utility, as described in "Configuring Multicast Discovery with opmnctl" on page 8-15. In general, `opmnassociate` is a simplified form of the more complete `opmnctl` command set for configuring multicast discovery. Using `opmnctl` for configuring multicast discovery is the recommended approach.

---

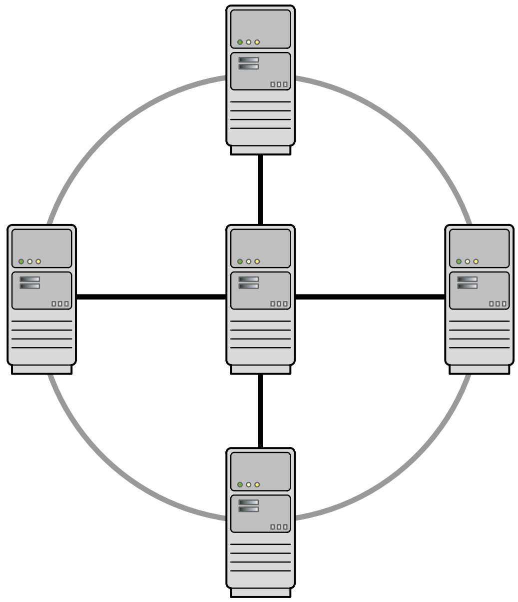
## Configuring Static Discovery Servers

This configuration is similar to a peer-to-peer clustering model, with one or more ONS nodes within the same cluster configured to serve as static hubs, or **discovery servers.**

Each ONS node in the cluster establishes a connection with a discovery server, which maintains the topology map for the cluster. The discovery server provides the connecting node with the current topology map, enabling the connecting node to communicate with the other ONS nodes within the cluster.

You can use opmnctl to configure the connection to a static discovery server. See "Configuring a Static Discovery Server Connection with opmnctl" on page 8-18 for details.

***Figure 8–3 Static Discovery Server Model***



Set the TCP/IP connection information for the discovery server within the `<discover>` element in the `opmn.xml` file on each static hub node within the cluster. For example:

```
<opmn>
 <notification-server>
  <port ...  />
  <ssl ... />
  <topology>
   <discover list="node1.company.com:6200"/>
  </topology>
 </notification-server>
 ...
</opmn>
```

The required information is as follows:

- The host name or IP address of the static discovery server
- The OPMN `remote` port, which is defined in the `<port>` element within the `opmn.xml` file installed on the static server, as follows:

  ```
  <port local="6100" remote="6200" request="6003"/>
  ```

---

**Note:** The `opmn.xml` file must be reloaded for changes to take effect in the OPMN runtime. Run the following command on the affected node to reload `opmn.xml`:

```
opmnctl reload
```

This command will not affect OPMN-managed components, including Oracle HTTP Server, OC4J, and deployed applications.

---

### Configuring a Static Discovery Server Connection with opmnctl

The OPMN command line tool, `opmnctl`, supports a new `config topology` command which allows you to specify, update or delete the `<discover>` entry within `opmn.xml`.

The `opmnctl` tool is installed in the *ORACLE_HOME*/`opmn/bin` directory on each node. The tool must be run individually on each node, and will only update the `opmn.xml` file on that node.

#### Inserting or Updating Discovery Data

The `update` command inserts or updates the `<discover>` element with the specified values. The syntax is as follows:

```
opmnctl config topology update discover="serverHost:opmnRemotePort"
```

For example:

```
opmnctl config topology update discover="node.company.com:6200"

opmnctl reload
```

#### Deleting Discovery Data

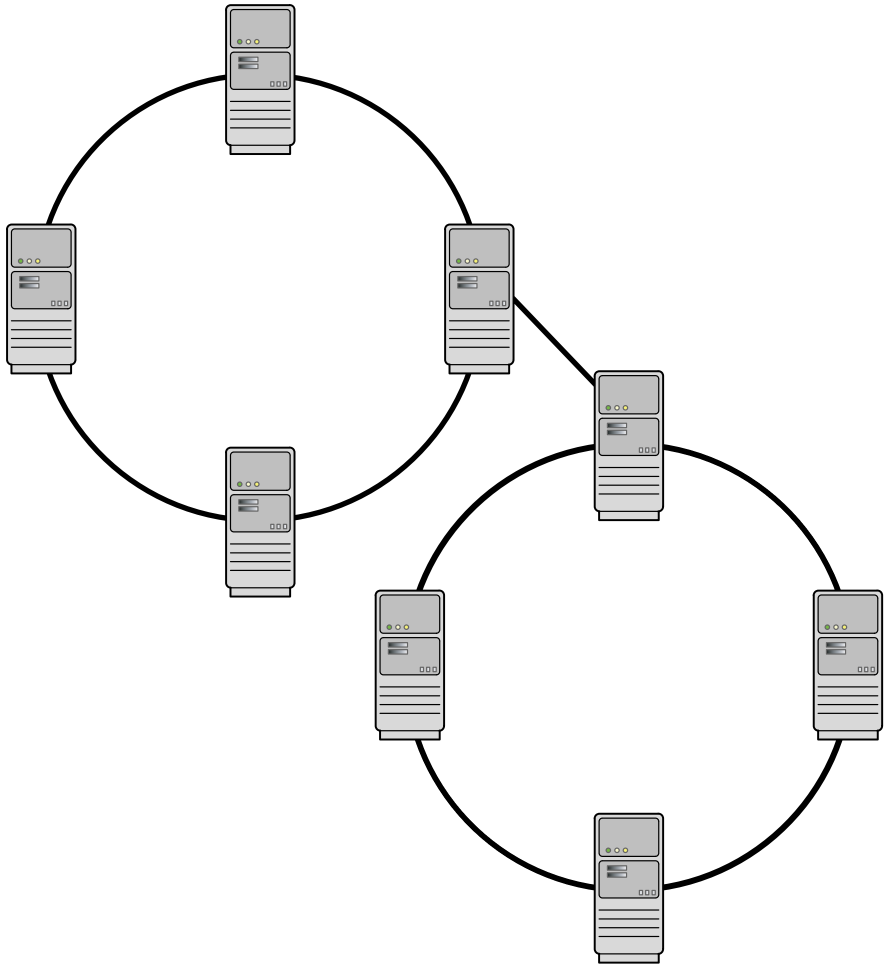
The `delete` command removes the `<discover>` element from `opmn.xml`, effectively removing the node from the cluster. If the `<topology>` element contains no other subelements, it will be removed as well.

```
opmnctl config topology delete discover

opmnctl reload
```

## Configuring Cross-Topology Gateways

For situations in which cluster topologies are on different subnets or are isolated by firewalls or physical locations, specific ONS nodes can be configured as **gateways**, enabling ONS notifications to be sent across the disparate topologies.

***Figure 8–4 Using Gateway Servers to Connect Topologies***



In this model, an ONS node within each isolated topology is configured as a gateway server, which serves as an entry point into the cluster. The gateway configuration is specified within a `<gateway>` subelement of the `<topology>` element.

Set the host and port for the source gateway node and each target node it will connect to as the value for the `list` attribute. The order in which the nodes are listed does not matter.

- For each node, specify the host name or IP address of the server and the OPMN `remote` port, which is defined in the `<port>` element within the `opmn.xml` file installed on the static server, as follows:

  ```
  <port local="6100" remote="6200" request="6003"/>
  ```

- Separate the data for each node with an ampersand (`&`).
- Include a `/` at the end of the list of nodes.

The following example shows the `opmn.xml` configuration for `node1`, which will connect with gateway nodes `node2` and `node3`. This same configuration can be set on each of these gateway nodes. Note the `/` at the end of the list:

```
<opmn>
 <notification-server>
  <port ...  />
  <ssl ... />
```

```
  <topology>
   <discover list="*224.0.0.37:8205"/>
     <gateway list="node1.com:6201&node2.com:6202&node3.com:6203/"/>
  </topology>
 </notification-server>
 ...
</opmn>
```

In addition to the <gateway> element, the <topology> element includes the <discover> element, which contains the multicast address and port used for dynamic discovery within the node's own cluster.

Alternatively, the entire <topology> element in the preceding example can be copied to the opmn.xml file on every node within the cluster topology. Only node1 will utilize the <gateway> configuration; it will be ignored by the other nodes.

To simplify configuration, you can set the connection data for all gateway nodes - sources and targets - in the <gateway> subelement and then copy this element to the opmn.xml file on each gateway node. Again, the order of the nodes does not matter; each node will simply ignore its own entry in the list.

---

**Note:** The opmn.xml file must be reloaded for changes to take effect in the OPMN runtime. Run the following command on the affected node to reload opmn.xml:

```
opmnctl reload
```

This command will not affect OPMN-managed components, including Oracle HTTP Server, OC4J, and deployed applications.

---

## Configuring a Machine to Work With and Without a Network Connection

When you work on a single machine using localhost, add the IP address in the <ipaddr> subelement of the <notification-server> element and explicitly set up a discover list in the <discover> element to refer to the localhost OPMN remote port, as defined in the cluster <*port*> element. An example of this configuration follows:

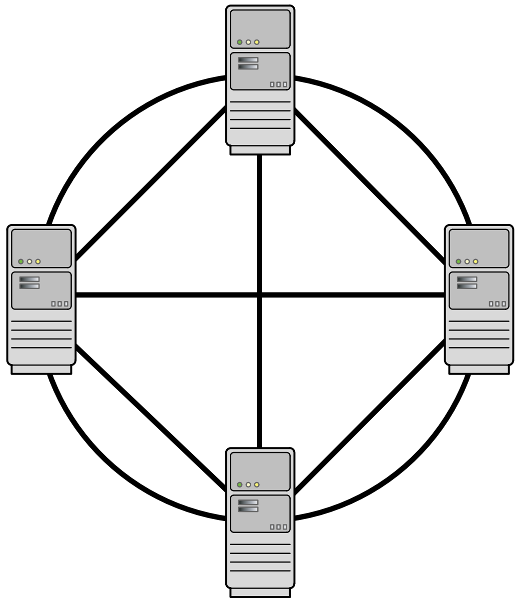
```
   <notification-server>
      <ipaddr remote="127.0.0.1" request="127.0.0.1"/>
      <port local="6101" remote="6201" request="6004"/>
      <ssl enabled="true"
wallet-file="$ORACLE_HOME\opmn\conf\ssl.wlt\default"/>      <topology>
         <discover list="localhost:6201"/>
      </topology>
   </notification-server>
```

If you supply the localhost IP address, 127.0.0.1, the machine can work with or without a network.

## Configuring Static Node-to-Node Communication

The static configuration model is essentially the same mechanism used in Oracle Application Server 10.1.2 and 9.0.4. It continues to be supported primarily to provide backward compatibility with these earlier releases.

***Figure 8–5 Static Node-to-Node Model***



In this configuration, a **node list** containing the host address and ONS remote listener port for each node in the cluster is supplied. Prior to Oracle Application Server Release 3 (10.1.3.0.0), when ONS configuration data was integrated into `opmn.xml`, this configuration would have been set in the `ons.conf` configuration file.

Define the host address and the ONS remote listener port, specified within the `<port>` subelement of `<notification-server>`, for each node in the cluster within the `<nodes>` subelement. Separate each node from the next with a comma.

For example:

```
<opmn>
 <notification-server>
  <port local="6101" remote="6202" request="6004"/>
  <ssl ... />
  <topology>
   <nodes list="node1-sun:6201,node2-sun:6202"/>
  </topology>
 </notification-server>
 ...
</opmn>
```

Supply the same list for each node in the cluster; each ONS instance will identify itself in the list and ignore that entry.

---

**Note:** The `opmn.xml` file must be reloaded for changes to take effect in the OPMN runtime. Run the following command on the affected node to reload `opmn.xml`:

```
opmnctl reload
```

This command will not affect OPMN-managed components, including Oracle HTTP Server, OC4J, and deployed applications.

---

# Viewing the Status of a Cluster

You can view the current status of the Oracle Application Server components within a cluster, using either `opmnctl` or Application Server Control.

- Viewing Cluster Status with opmnctl
- Viewing Cluster Status in Application Server Control

## Viewing Cluster Status with opmnctl

You can check the status of the cluster using `opmnctl` on any Oracle Application Server node within the cluster.

```
opmnctl @cluster status
```

The output shows the status of the components installed in each active Oracle Application Server instance within the cluster:

```
Processes in Instance: AppSrv1.comp1.yourcompany.com
-------------------+--------------------+---------+---------
ias-component      | process-type       |     pid | status
-------------------+--------------------+---------+---------
OC4JGroup:COLORS   | OC4J:home          |   26880 | Alive
OC4JGroup:COLORS   | OC4J:oc4j_soa      |   26256 | Alive
HTTP_Server        | HTTP_Server        |   26879 | Alive

Processes in Instance: AppSrv2.comp2.yourcompany.com
-------------------+--------------------+---------+---------
ias-component      | process-type       |     pid | status
-------------------+--------------------+---------+---------
OC4JGroup:COLORS   | OC4J:home          |   26094 | Alive
OC4JGroup:COLORS   | OC4J:oc4j_soa      |     N/A | Down
HTTP_Server        | HTTP_Server        |   26093 | Alive
```

## Viewing Cluster Status in Application Server Control

Click the **Cluster Topology** link in the upper left corner of the Application Server Control home page.

The resulting page displays each Oracle Application Server instance that is active within the cluster, as well as the active applications on each instance. You can access an instance or a deployed application within the cluster through this page.

# Configuring Routing and Load Balancing with Oracle HTTP Server

The term **load balancing** refers to the process of distributing incoming service requests over server instances within a cluster. Load balancing in an Oracle Application Server cluster is managed by the `mod_oc4j` module of Oracle HTTP Server. In this configuration, the Oracle HTTP Server instance acts as a front-end listener for incoming HTTP and HTTPS requests; `mod_oc4j` then routes each request to an OC4J instance serving the requested application.

In Oracle Application Server Release 3 (10.1.3), load balancing is completely dynamic for Oracle Application Server instances that belong to the same cluster. No additional Oracle HTTP Server or `mod_oc4j` configuration is required.

- Dynamic OC4J instance discovery

Oracle HTTP Server instances are dynamically updated with information on each OC4J instance in the cluster and the applications deployed to it, enabling Oracle HTTP Server to route requests to the appropriate instance.

See "Enabling Dynamic Configuration of Application Mount Points" on page 8-29 for details.

- Dynamic routing

  The new release supports a **routing ID** mechanism that enables you, optionally, to control which OC4J instances to which an Oracle HTTP Server instance forwards requests, essentially enabling you to control the set of OC4J instances that will service requests from specific Oracle HTTP Server instances. All Oracle HTTP Server and OC4J instances are configured to use a default routing ID upon installation; as such, no configuration is required.

  See "Using Web Server Routing IDs to Control OC4J Request Routing" on page 8-23 for details.

The only requirement is that the ONS servers within the various Oracle HTTP Server and OC4J instances within the cluster be connected using one of the clustering configuration mechanisms outlined in this chapter. See "Configuring a Cluster" on page 8-13 for details.

## Using Web Server Routing IDs to Control OC4J Request Routing

Every Oracle HTTP Server and OC4J instance in an OPMN-managed installation is assigned a *routing ID* that is defined in `opmn.xml`. An Oracle HTTP Server instance will route incoming Web requests only to OC4J instances that share its routing ID. This means that you can effectively define the set of OC4J instances to which a specific Oracle HTTP Server instance will route requests.

A default routing ID is assigned to all component instances, so that upon installation, every Oracle HTTP Server instance in a cluster can route requests to any OC4J instance within the cluster.

In a typical Oracle Application Server cluster, one or more Oracle HTTP Server instances receives requests from users and then routes those requests to the applications deployed within the cluster. The routing ID of each application server, each OC4J instance, each group, and each deployed application determines where the Oracle HTTP Server routes each request.

---

**Caution:** Changing the routing ID for an application server, component, or individual applications can prevent HTTP requests from being sent to your deployed applications. Unless other instances of the application are available in the cluster and have the same routing ID, this action can make the application unavailable to your users.

---

The rest of this section describes how to change routing IDs, in the following topics:

- Changing Routing IDs Through Application Server Control
- Changing Routing IDs in the opmn.xml file

For information on how Web sites are configured to listen for AJP requests, see Configuring Web Site Connection Data on page 13-2.

### Changing Routing IDs Through Application Server Control

To change or view the routing ID assigned to each application and component of your cluster through Application Server Control:

1. Navigate to the Cluster Topology page
2. Scroll to the Administration section of the page and click **Routing ID Configuration**.

The Routing ID Configuration page is designed to show the hierarchy of components and applications within your cluster topology. For example, if you click the **Expand** icon for an application server, then you see the groups within the application server instance. Within each group, you see the OC4J instances that are part of that group. And finally, if you expand a specific OC4J instance, you see the applications deployed to the OC4J instance.

By default, the application server instance is assigned a routing ID, and the groups, OC4J instances, and applications inherit the routing ID of the application server. If you enter a different routing ID for a specific group, OC4J instance, or application, then that new routing ID will override the routing ID inherited from the application server.

If you are managing multiple application server instances within a cluster, notice that the same group appears multiple times in the hierarchy, once for each application server that contains an OC4J instance that is a member of the group. This is because the hierarchy of the Routing ID Configuration page is based on the Oracle Process Management and Notification (OPMN) software configuration file (`opmn.xml`), which is stored in the Oracle home directory of each application server. As result, use caution when modifying the routing ID of a group. Be sure to assign the same routing ID to all instances of the group on the Routing ID Configuration page, unless you want specific Oracle HTTP Server requests to be routed to only some of the OC4J instances in the group.

### Changing Routing IDs in the opmn.xml file

The routing ID is defined in `opmn.xml` in a `<data>` element where the `id` attribute equals `routing-id`. The `<data>` element entry is a subelement of `<category id="start-parameters">`, which specifies parameters passed to the instance at startup. The default `routing-id` value set for each instance is `g_rt_id`.

```
<category id="start-parameters">
  <data id="routing-id" value="g_rt_id"/>
</category>
```

The `<data>` element containing the default routing ID is set within the `<ias-instance>` element, which contains the OPMN configuration data for the Oracle Application Server instance. Because the routing ID is set at this level, the `routing-id` value set in this `<data>` element is applied to all instances of the Oracle HTTP Server and OC4J components installed within the Oracle Application Server instance.

```
<opmn>
 <process-manager>
  ...
  <ias-instance id="instance1" name="instance1">
   ...
    <environment>
     ...
    </environment>
    <module-data>
     <category id="start-parameters">
      <data id="routing-id" value="g_rt_id"/>
```

```
      </category>
     </module-data>
    </environment>
    <ias-component id="HTTP_Server">
     ...
    </ias-component>
    <ias-component id="default_group">
     ...
    </ias-component>
  </ias-instance>
 </process-manager>
</opmn>
```

However, the routing ID can be set at the individual Oracle HTTP Server or OC4J instance level by adding a `<data>` element within the `<category id="start-parameters">` element for the component. This value overrides the routing ID assigned at the Oracle Application Server instance level.

You can specify any string as the value of the `routing-id` attribute. There is no required format for this identifier. The following entry in `opmn.xml` sets the routing ID for an Oracle HTTP Server instance:

```
<opmn>
 <process-manager>
  ...
  <ias-instance id="instance1" name="instance1">
   ...
   <ias-component id="HTTP_Server">
     <environment>
      ...
     </environment>
     <process-type id="HTTP_Server" module-id="OHS">
       <module-data>
        <category id="start-parameters">
          <data id="start-mode" value="ssl-enabled"/>
          <data id="routing-id" value="group_b_id"/>
        </category>
       </module-data>
       <process-set id="HTTP_Server" numprocs="1"/>
      </process-type>
     </ias-component>
  </ias-instance>
 </process-manager>
</opmn>
```

The following entry in `opmn.xml` sets the routing ID for the OC4J `home` instance:

```
<opmn>
 <process-manager>
  ...
  <ias-instance id="instance1" name="instance1">
   ...
    <ias-component id="default_group">
     <environment>
     </environment>
     <process-type id="home" module-id="OC4J" status="enabled">
       <module-data>
        <category id="start-parameters">
          <data id="java-options" ... />
          <data id="routing-id" value="group_b_id"/>
        </category>
       </module-data>
```

```
        <process-set id="HTTP_Server" numprocs="1"/>
        <port id="default-web-site" range="12501-12600" protocol="ajp" />
        <port id="rmi" range="12401-12500"/>
        <port id="jms" range="12601-12700"/
        <port id="rmis" range="12701-12800"/
        <process-set id="default_group" numprocs="1"/>
       </process-type>
      </ias-component>
   </ias-instance>
  </process-manager>
</opmn>
```

## Setting mod_oc4j Load Balancing Options

The mod_oc4j module within Oracle HTTP Server delegates requests to OC4J processes. Whenever Oracle HTTP Server receives a request for a URL that is intended for OC4J, Oracle HTTP Server routes the request to the mod_oc4j module, which then routes the request to an OC4J process. If an OC4J process fails, OPMN detects the failure and mod_oc4j does not send requests to the failed OC4J process until the OC4J process is restarted.

You can configure mod_oc4j to load balance requests to OC4J processes. Oracle HTTP Server, through mod_oc4j, supports different load balancing policies. Load balancing policies provide performance benefits along with failover and high availability, depending on the network topology and host machine capabilities.

You can specify different load balancing routing algorithms for mod_oc4j depending on the type and complexity of routing you need. Stateless requests are routed to any destination available based on the algorithm specified in `mod_oc4j.conf`. Stateful HTTP requests are forwarded to the OC4J process that served the previous request using session identifiers, unless mod_oc4j determines through communication with OPMN that the process is not available. In this case, mod_oc4j forwards the request to an available OC4J process following the specified load-balancing protocol.

By default, all OC4J instances have the same weight (all instances have a weight of 1), and mod_oc4j uses the round robin method to select an OC4J instance to forward a request to. An OC4J instance's weight is taken as a ratio compared to the weights of the other available OC4J instances in the topology to define the number of requests the instance should service. If the request belongs to an established session, mod_oc4j forwards the request to the same OC4J instance and the same OC4J process that started the session.

The mod_oc4j load balancing options do not take into account the number of OC4J processes running on an OC4J instance when determining which OC4J instance to send a request to. OC4J instance selection is based on the configured weight for the instance, and its availability.

To modify the mod_oc4j load balancing policy, set the `Oc4jSelectMethod` and the `Oc4jRoutingWeight` directives in the *`ORACLE_HOME`*`/Apache/Apache/conf/mod_oc4j.conf` file:

1. In the `mod_oc4j.conf` file on each Oracle Application Server instance, within the `<IfModule mod_oc4j.c>` section, set the `Oc4jSelectMethod` directive to one of the values shown in Table 8–3.

   If you set the `Oc4jSelectMethod` directive to either `roundrobin:weighted` or `random:weighted`, you may also need to set the `Oc4jRoutingWeight` directive to specify the weight (see the next step).

See "Choosing a mod_oc4j Load Balancing Algorithm" on page 8-28 for tips on choosing a routing algorithm.

***Table 8–3 Values for Oc4jSelectMethod***

| Value | Description |
|---|---|
| `roundrobin` (default) | mod_oc4j places all the OC4J processes in the topology in a list, and it selects processes in order from the list. |
| `roundrobin:local` | Similar to `roundrobin`, but the list includes only local OC4J processes. If no local OC4J processes are available, then it selects a remote OC4J process. |
| `roundrobin:weighted` | mod_oc4j distributes the total request load to each OC4J instance based on routing weight configured on each instance. It then selects OC4J processes from the local instance in a round robin manner.<br><br>You configure the weight using the `Oc4jRoutingWeight` directive. |
| `random` | mod_oc4j randomly selects an OC4J process from a list of all OC4J processes in the topology. |
| `random:local` | Similar to `random`, but mod_oc4j gives preference to local OC4J processes. If no local OC4J processes are available, then it selects a remote OC4J process. |
| `random:weighted` | mod_oc4j selects an OC4J process based on the weight configured for each instance in the topology.<br><br>You configure the weight using the `Oc4jRoutingWeight` directive. |
| `metric` | mod_oc4j routes requests based on runtime metrics that indicate how busy a process is. |
| `metric:local` | Similar to metric, but mod_oc4j gives preference to local OC4J processes. If no local OC4J processes are available, then it routes to a remote OC4J process. |

Example:

```
Oc4jSelectMethod random:local
```

For information on how to set up metric-based load balancing, see *Oracle HTTP Server Administrator's Guide*.

2. If you set the `Oc4jSelectMethod` directive to a weight-based method (that is, `roundrobin:weighted` or `random:weighted`), you may also need to set the `Oc4jRoutingWeight` directive to specify the weight.

   `Oc4jRoutingWeight` has the following syntax:

   `Oc4jRoutingWeight` *`hostname weight`*

   If you do not set the `Oc4jRoutingWeight` directive, it uses a default weight of 1.

   Example: If you have a topology that consists of three instances (A, B, and C), and you want B and C to get twice as many requests as A, set the following directives for B and C:

   ```
   Oc4jSelectMethod roundrobin:weighted
   Oc4jRoutingMethod hostB 2
   Oc4jRoutingMethod hostC 2
   ```

   Setting `Oc4jRoutingMethod` for hostA is optional because the default value is 1.

3. Restart Oracle HTTP Server on all instances in the topology for the changes to take effect.

```
> opmnctl @cluster restartproc ias-component=HTTP_Server
```

### Choosing a mod_oc4j Load Balancing Algorithm

Use the following guidelines to help determine which mod_oc4j load balancing option to use:

- In a topology with identical machines running Oracle HTTP Server and OC4J in the same Oracle home, the round robin with local affinity algorithm is preferred. In this case Oracle HTTP Server gains little by using mod_oc4j to route requests to other machines, except in the extreme case that all OC4J processes on the same machine are not available.
- For a distributed deployment, where one set of machines runs Oracle HTTP Server and another set runs OC4J instances that handle requests, the preferred algorithms are simple round robin and simple metric-based. To determine which of these two works better in a specific setup, you may need to experiment with each and compare the results. This is required because the results are dependent on system behavior and incoming request distribution.
- For a heterogeneous deployment, where the different Oracle Application Server instances run on nodes that have different characteristics, the weighted round robin algorithm is preferred. In addition to setting the weight for each instance, remember to tune the number of OC4J processes running on each Oracle Application Server instance to achieve the maximum benefit. For example, a machine with a weight of 4 gets four times as many requests as a machine with a weight of 1, but you need to ensure that the system with a weight of 4 is running four times as many OC4J processes.
- Metric-based load balancing is useful when there are only a few metrics that dominate the performance of an application, for example, CPU or number of database connections.

# Configuring Application Mount Points

To route incoming requests, Oracle HTTP Server utilizes a list of application-specific *mount points*, which map the URLs supplied in requests with the OC4J instances that will service the requests. This section includes the following topics on mount point creation:

- Enabling Dynamic Configuration of Application Mount Points
- Changing the Mount Point Configuration Algorithm
- Viewing Mount Point Configuration Data

See the *Oracle HTTP Server Administrator's Guide* for additional details on mount point configuration.

## Enabling Dynamic Configuration of Application Mount Points

In previous releases of Oracle Application Server the list of application mount points had to be managed manually in the `mod_oc4j` configuration file, `mod_oc4j.conf`.

In the current release the list of mount points is dynamically updated as new nodes and applications are added to, or removed from, the cluster. Using ONS notifications, every OC4J instance within the cluster sends mount point data for each of its deployed applications to mod_oc4j, which adds this information to its internal routing table.

This dynamic discovery mechanism is enabled by default for clustered Oracle Application Server instances and requires no additional configuration.

The mount point information sent by each OC4J instance to Oracle HTTP Server includes these items:

- The OC4J host address
- OC4J port information, including the Apache JServ Protocol (AJP) listener port

  This value is the lowest available port assigned to AJP in the `opmn.xml` file on the node.
- The Web module name

  This value is defined as the value of the `name` attribute in the `<web-app>` element defined for the module in the `*-web-site.xml` configuration file the module is bound to.
- The Web context, or root context, defined for the application

  This value is set in the `root` attribute of the `<web-app>` element defined for the module `*-web-site.xml` configuration file.

> **Note:** Dynamically configured mount points are not written to the the `mod_oc4j` configuration file, `mod_oc4j.conf`.

When a new application is deployed to an OC4J instance, its mount point information is transmitted to Oracle HTTP Server, enabling `mod_oc4j` to dynamically *discover* the application and begin routing requests to it.

Conversely, when an application is stopped or removed from an OC4J instance, the `mod_oc4j` routing table is updated to reflect the application's absence, causing `mod_oc4j` to stop routing requests to the application instance.

You can still configure application mount points manually, as "Changing the Mount Point Configuration Algorithm" on page 8-29 describes. For information about viewing the mount point list, see "Viewing Mount Point Configuration Data" on page 8-31. For additional information about configuring mount points, see *Oracle HTTP Server Administrator's Guide*.

## Changing the Mount Point Configuration Algorithm

Although dynamic mount point creation is enabled by default, you do have the option of continuing to use manually configured mount points, which is the default mechanism supported in previous releases of Oracle Application Server.

Static mount points are defined in the `mod_oc4j` configuration file, `mod_oc4j.conf`, which is installed in the *`ORACLE_HOME`*`/Apache/Apache/conf` directory. By default, Oracle HTTP Server will create dynamic mount points as applications are deployed; however, static mount points defined in `mod_oc4j.conf` will also be honored.

The mount point configuration mechanism to use is specified in the `Oc4jRoutingMode` parameter in `mod_oc4j.conf`. Table 8–4 lists the values for this variable. See the *Oracle HTTP Server Administrator's Guide* for details on mount point configuration and using `mod_oc4j.conf`.

---

**Note:** If you change `Oc4jRoutingMode` to `Static` in the `mod_oc4j` configuration file, you also need to add the following configuration to `mod_oc4j.conf` to prevent losing access to Application Server Control:

```
Oc4jMount /em/* home
Oc4jMount /em home
```

---

***Table 8–4 Oc4jRoutingMode Values***

| Value | Description |
|---|---|
| `Dynamic` | Dynamically configured mount points are used exclusively. Static mount points will be ignored. |
| `Static` | Static, manually configured mount points defined in `mod_oc4j.conf` are used exclusively. Dynamic mount points will not be created for new applications. |
| `DynamicOverride` | Both dynamic and static mount points are used. In the event of a conflict, the dynamically configured mount point will be used. |
| `StaticOverride` | Both dynamic and static mount points are used; however, in the event of a conflict, the static, manually configured mount point will be used.<br>This is the default mode used, although it is not defined in `mod_oc4j.conf` by default. |

The following `mod_oc4j.conf` example enables the `DynamicOverride` mode, in which the dynamic mount points specified will take precedence over static mount points in the event of a conflict:

```
#########################################################
# Oracle iAS mod_oc4j configuration file: mod_oc4j.conf #
#########################################################

LoadModule oc4j_module libexec/mod_oc4j.so
Oc4jRoutingMode DynamicOverride
<IfModule mod_oc4j.c>
  <Location /oc4j-service>
    SetHandler oc4j-service-handler
  </Location>
    Oc4jMount /j2ee/*
    Oc4jMount /webapp home
    Oc4jMount /webapp/* home
    Oc4jMount /cabo home
    Oc4jMount /cabo/* home
    Oc4jMount /stressH home
    Oc4jMount /stressH/* home
</IfModule>
```

## Viewing Mount Point Configuration Data

You can configure Oracle HTTP Server to output mount point configuration data to a Web page generated on the Oracle HTTP Server host.

Add the following entry to the Oracle HTTP Server configuration file, `httpd.conf`, on the Oracle HTTP Server host machine. This file is installed in *`ORACLE_HOME`*`/Apache/Apache/conf`.

```
<IfModule mod_oc4j.c>
   Oc4jSet StatusUri /oc4j-status
</IfModule>
```

You will now be able to view mount point data by appending the `/oc4j-status` context URI to the Oracle HTTP Server server URL:

```
http://ohsHost:ajpPort/oc4j-status
```

For example:

```
http://node1.company.com:7777/oc4j-status
```

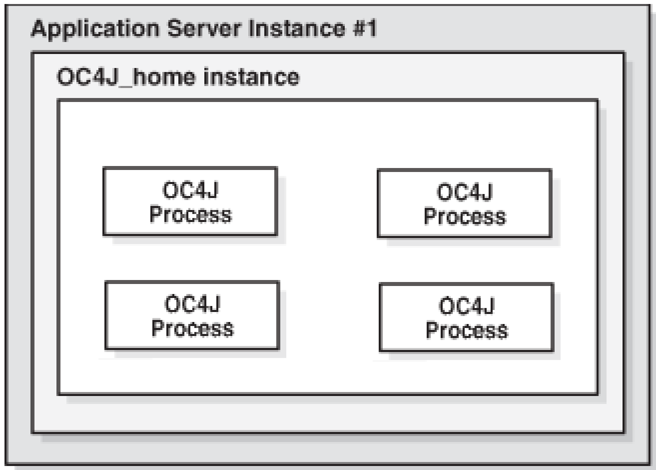
The following is sample output displayed in the resulting Web page, with comments:

```
hostname          : node1.company.com
local instance    : node1.company.com
select method     : Round-Robin
select affinity   : None
# OHS routing configuration
routing mode      : Static-Dynamic
routing ID        : g_rt_id

OC4J Dynamic routing
# Applications using dynamic routing

# 'ascontrol' application
application       : ascontrol
  context         : /em
  process (Jgroup): 0

# 'demos' application
application       : demos
  context         : /ojspdemos/jstl, /ojspdemos
  process (Jgroup): 0 (demos)

OC4J Process List

  process,ias instance,host,port,status
  0 : home.node1.company.com, node1.company.com, 12502, ALIVE
    1 : home.node1.company.com, node1.company.com, 12501, ALIVE
    2 : home.node1.company.com, node1.company.com, 12503, ALIVE
```

## Running an OC4J Instance on Multiple JVMs

OC4J executes on the Java Virtual Machine (JVM) of the standard Java Development Kit (JDK). By default, each OC4J instance uses one JVM; however, you can configure an OC4J instance so it runs on multiple JVMs, as Figure 8–6 shows. When you configure an OC4J instance to run on multiple JVMs, the instance essentially runs on multiple processes, which can improve performance and provide a level of fault tolerance for your deployed applications.

***Figure 8–6 OC4J Instance Running on Multiple JVMs***




This figure shows four processes that are configured to run from an OC4J instance, named `OC4J_home`, in one of the Oracle Application Server instances within a cluster.

---

**Notes:** The OC4J instance named `home` typically cannot be configured to run with multiple processes because it hosts the Application Server Control application, `ascontrol`, which is not suitable for running in the multiple-process model.

You cannot run an application that uses an EJB timer in an OC4J instance that runs on multiple processes. EJB timers are supported only in an OC4J instance that runs on a single JVM (where `numprocs=1` in the `<process-set>` element of the `opmn.xml` configuration file).

---

Multiple JVMs, however, require additional hardware resources to run efficiently. Also, if multiple processes run on the same host and the host goes down, all the JVM processes will go down.

If you install and manage multiple application server instances, you can install those application server instances on multiple hosts. By clustering the application servers and creating OC4J groups from the Cluster Topology page (or from the command line or API), you can also take advantage of application clustering and load balancing. Application clustering, described in Chapter 9, "Application Clustering in OC4J", ensures that state information is replicated to the different instances of your application running in each JVM.

In addition, Oracle Application Server clusters and OC4J groups provide added protection against hardware or network outages. If one host goes down, the applications deployed on the other hosts are still available.

---

**Note:** Application Server Control (represented by the `ascontrol` application) cannot run on an OC4J instance that is running on multiple JVMS. Make sure that you do not configure multiple JVMs for the OC4J instance that is hosting the active Application Server Control instance.

---

## Creating Additional JVMs for an OC4J Instance

By default, each OC4J instance uses one JVM. However, you can configure the OC4J instance so it runs on multiple JVMs, with a copy of the instance on each JVM. You can create additional JVMs for an OC4J instance on the Server Properties page in Application Server Control or by setting the `numprocs` attribute for an OC4J instance directly in the `opmn.xml` file.

---

**Note:** When the `numprocs` attribute for an OC4J instance is greater than `1` (*n*) in the `opmn.xml` file, whether you set it with Application Server Control or by editing the file, Oracle Application Server starts *n* separate, physical JVM processes, each dedicated to run a copy of the related OC4J instance (including any deployed applications). Be sure to take the physical hardware resources into account when you set this value.

---

### How to Create Additional JVMs for an OC4J instance with Application Server Control

You can add one or more JVMs for an OC4J instance on the Server Properties page in Application Server Control.

#### To create additional JVMs for an OC4J instance with Application Server Control:

1. Navigate to the OC4J Home page and then click **Administration** to display the OC4J Administration page, which contains a table listing the various administration tasks you can perform for the OC4J instance.
2. On the Administration page, select **Server Properties** to display the OC4J Server Properties page.
3. Enter the number of JVMs to configure for the OC4J instance in the Number of VM Processes field.
4. Click **Apply**.
5. Restart the OC4J instance from the Cluster Topology page or the OC4J Home page.

### How to Create Additional JVMs for an OC4J instance in the opmn.xml File

By default, each OC4J instance uses one JVM. However, you can configure an OC4J instance so that it runs on multiple JVMs. You can add one or more JVMs by setting the `numprocs` attribute of the <process-set> element> in the configuration for an OC4J instance in the `opmn.xml` file.

#### To create additional JVMs for an OC4J instance in the opmn.xml file:

1. Edit the `opmn.xml` file.
2. In the `<process-set>` element of the OC4J configuration, change the value of the `numprocs` attribute to the number of JVMs on which you want OC4J to run.

   Example 8–1 shows the `numprocs` setting in `opmn.xml`.

***Example 8–1 numprocs Attribute for OC4J in opmn.xml***

```
<opmn>
 . . .
 <ias-component id="default_group">
  <process-type id="home" module-id="OC4J" status="enabled">
    <module-data>
```

```
      <category id="start-parameters">
        <data id="java-options" value=" -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"/>
    <port id="rmis" range="12701-12800"/>
    <process-set id="default_group" numprocs="2"/>
  </process-type>
 </ias-component>
</opmn>
```

3. Save the `opmn.xml` file.
4. Restart the OC4J instance.

## Monitoring Multiple JVMs

When you use multiple JVMs, it is important to monitor the performance of the JVMs to be sure the current hardware resources can handle the configuration. From Application Server Control, you can monitor and compare the performance of JVMs associated with the OC4J instance.

The following topics describe how to monitor JVM metrics with Application Server Control:

- Monitoring Dynamic Monitoring Service JVM Metrics
- Setting the jmxremote System Property for Monitoring J2SE JVM 5.0 Metrics
- Monitoring J2SE 5.0 JVM Metrics in an Oracle Application Server Environment
- Monitoring J2SE 5.0 JVM Metrics in a Standalone OC4J Environment

Before you can monitor the J2SE 5.0 JVM metrics with Application Server Control, you must be running OC4J on JDK 5.0 (1.5) and set the `jmxremote` system property for each OC4J instance to enable this monitoring.

### Monitoring Dynamic Monitoring Service JVM Metrics

If you are running OC4J in an Oracle Application Server environment, then you can monitor a set of Dynamic Monitoring Service (DMS) metrics for each JVM. These metrics are unavailable in the standalone OC4J environment.

To view the DMS JVM Metrics in an Oracle Application Server environment with Application Server Control:

1. Navigate to the OC4J Home page.
2. Locate the **Virtual Machines** field in the General section of the OC4J Home page.
3. Click the number that indicates how many JVMs are configured for the OC4J instance.

The JVM Metrics page displays a summary of key metrics for all the JVMs configured for the selected OC4J instance. You can use this table to compare the performance of multiple JVMs.

4. For more detailed information, click the name of a JVM.

   The OC4J JVM page displays a set of charts and numeric metrics that give you a detailed picture of how the JVM is performing. Select a refresh interval from the **View Data** list. You can then view the changes in the performance charts over a period of time.

### Setting the jmxremote System Property for Monitoring J2SE JVM 5.0 Metrics

You can set the jmxremote System Property for monitoring J2SE JVM 5.0 metrics with Application Server Control, with an OC4J startup option, or for an OPMN-managed environment, in the `opmn.xml` file.

#### Using Application Server Control to Set the jmxremote System Property

To enable the monitoring of JVM J2SE 5.0 metrics for each OC4J instance with Application Server Control:

1. Navigate to the OC4J Home page and then click **Administration** to display the OC4J Administration page, which contains a table listing the various administration tasks you can perform for the OC4J instance.
2. On the OC4J Administration page, select **Server Properties** to display the OC4J Server Properties page.
3. Scroll down to the Command Line Options section of the page and select **Enable J2SE 5.0 Platform MBeans**.
4. Click `Apply` to apply the changes.
5. Navigate to the Cluster Topology page, select the OC4J instance, and then click **Restart**.

#### Setting the jmxremote System Property in an OC4J Startup Option

You can also enable monitoring of JVM J2SE 5.0 metrics by including the following string as an OC4J runtime startup option:

```
com.sun.management.jmxremote
```

For information about how to specify OC4J runtime startup options, see Setting OC4J Runtime Options at Startup on page 4-2.

If you are running OC4J in a standalone environment, include the following argument to the OC4J `java` command:

```
java -Dcom.sun.management.jmxremote -jar oc4j.jar
```

#### Setting the jmxremote System Property in the opmn.xml File

If you are running OC4J in an OPMN-managed, Oracle Application Server environment, include `-Dcom.sun.management.jmxremote` in the in the `opmn.xml` file, as follows:

```
<ias-component id="default_group">
  <process-type id="home" module-id="OC4J" status="enabled">
     <module-data>
        <category id="start-parameters">
           <data id="java-options" value="-Dcom.sun.management.jmxremote"/>
           ...
```

```
        </category>
        ...
      </module-data>
      ...
    </process-type>
    ...
</ias-component>
```

Using Application Server Control to enable the J2SE 5.0 Platform MBeans results in the `jmxremote` system property being set in the `opmn.xml` file. If you use this approach, then there is no need to set the property manually in the `opmn.xml` file.

### Monitoring J2SE 5.0 JVM Metrics in an Oracle Application Server Environment

To view the J2SE 5.0 JVM Metrics in an Oracle Application Server environment with Application Server Control:

1. On the OC4J Home page, locate the **Virtual Machines** field in the General section.
2. Click the number that indicates how many JVMs are configured for the OC4J instance.

   Enterprise Manager displays the JVM Metrics page.
3. Click the name of a JVM.

   Enterprise Manager displays the OC4J JVM page.
4. Scroll to the Related Links section of the page and click **J2SE 5.0 Metrics**.

### Monitoring J2SE 5.0 JVM Metrics in a Standalone OC4J Environment

To view the J2SE 5.0 JVM Metrics in a standalone OC4J environment with Application Server Control:

1. On the OC4J Home page, click **Performance** to display the OC4J Performance page.
2. Scroll to the Related Links section of the page and click **J2SE 5.0 Metrics**.

# 9

# Application Clustering in OC4J

This chapter discusses the application clustering framework provided in OC4J 10*g* (10.1.3.5.0). It includes these topics:

- Overview of Application Clustering in OC4J
- How Application Clustering Differs from Previous OC4J Releases
- Configuring Application Clustering

## Overview of Application Clustering in OC4J

OC4J provides a flexible framework for creating a clustered environment for development and production purposes. An **application cluster** is the same set of applications hosted by two or more OC4J instances. The OC4J application clustering framework supports:

- Replication of objects and values contained in an HTTP session or a stateful session Enterprise JavaBeans (SFSB) 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 an application cluster.
- Configuration within an OC4J instance at either the global server or application level.

A new `<cluster>` element, which contains a number of new subelements, has been added to the XML schema definition for these files to provide a single mechanism for management of application clustering. See "Specifying the <cluster> Element" on page 9-11 for descriptions of this element and its subelements.

## How Application Clustering Differs from Previous OC4J Releases

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

### Islands No Longer Supported

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 an Oracle Application Server 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 applicable only to Web applications; EJB applications could not utilize the island configuration.

In OC4J 10*g* (10.1.3), you can still effectively limit the number of nodes to which to replicate data by using the `write-quota` attribute of the `<cluster>` element. This attribute makes it possible to control the extent of state replication.

See "Managing the Number of JVMs to Which Application State Data Is Replicated" on page 9-5 and "Specifying the <cluster> Element" on page 9-11 for details on the `write-quota` attribute.

### loadbalancer.jar No Longer Used

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

### Application-Clustering-Specific XML Elements Deprecated

The following XML elements are deprecated and should no longer be used to configure clustering:

- 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

The new `<cluster>` element is now used for all application cluster management.

## Configuring Application Clustering

Application clustering is enabled by adding the `<cluster>` element to the `orion-application.xml` file of each application to be clustered in an OC4J instance. For deployed applications, this file is located in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application-deployments/`*`applicationName`* directory. See "Specifying the <cluster> Element" on page 9-11 for descriptions of this element and its subelements.

This section includes the following topics:

## Enabling Application Clustering

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

- **Enabling clustering for all applications**

  Application clustering can be enabled by default for all applications deployed to an OC4J instance, through `ORACLE_HOME/j2ee/instance/config/application.xml`, the configuration file for the `default` application. All other applications deployed into the OC4J instance inherit default properties from this application, including the application clustering configuration.

- **Enabling clustering for a specific application**

  Application clustering is defined in the application-specific `ORACLE_HOME/j2ee/instance/application-deployments/app_name/orion-application.xml` file. Settings in this file override the global configuration, as well as the configuration inherited from a parent application.

> **Note:** Application clustering can also be configured at the time the application is deployed by using Oracle Enterprise Manager 10*g* Application Server Control, through either the deployment tasks or the deployment plan editor.
>
> See the *Oracle Containers for J2EE Deployment Guide* for details.

Any changes made to a particular application's `orion-application.xml` file in one OC4J instance must be replicated to the corresponding XML files in other OC4J instances for all applications within an Oracle Application Server cluster. For more information, see "Replicating Changes Across a Cluster" on page 8-11.

At the application level, application clustering can be configured at the time the application is deployed into an OC4J instance by using the deployment plan editor, which sets values in each application's `orion-application.xml` file. See the *Oracle Containers for J2EE Deployment Guide* for details on using the deployment plan editor.

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

## Setting Replication Policies

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

The replication policy applied to all Web modules and EJB modules 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 9–1 for an overview of `trigger` attribute values.

- The `scope` attribute defines what data is replicated: Either all attribute or variable values, or only changed values. By default, only modified HTTP session attributes are replicated; for stateful session beans, all member variables are replicated.

  See Table 9–2 for an overview of `scope` attribute values.

***Table 9–1 <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.<br><br>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.<br><br>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.<br><br>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 9–2 <replication-policy> scope Attribute Values*

| scope Value | HttpSession | Stateful Session Bean |
|---|---|---|
| `modifedAttributes` (default) | Replicate only modified HTTP session attributes; that is, values changed by calling `setAttribute()` on the `HttpSession` object. | Not applicable. |
| `allAttributes` | Replicate all attribute values set on the HTTP session. | Replicate all member variable values set on the stateful session bean. |

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

See Table 9–3 for valid values for the `replication` attribute. For example:

```
<session-deployment name="MyStatefulVM" replication="onShutdown" />
<session-deployment name="MyEntity2" replication="onRequestEnd" />
```

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

*Table 9–3 Stateful Session EJB Replication Policy Configuration*

| replication Value | Description |
|---|---|
| `onRequestEnd` (default) | 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 a Linux or UNIX environment. |
| `none` | Do not replicate data. |

## Managing the Number of JVMs to Which Application State Data Is Replicated

You can effectively limit the number of JVMs to which state data is replicated by using the `write-quota` attribute of the `<cluster>` element. This functionality makes it possible to reduce network traffic and related overhead by controlling the extent of state replication.

The default value for `write-quota` is `1`, indicating that state will be replicated to one other JVM within an Oracle Application Server cluster.

An application group member actually runs on a JVM, not an Oracle Application Server node. It is possible to construct architectures and configurations in which multiple JVMs are running per node as components of the cluster.

To force state replicas to be stored on separate physical nodes, which provides failover protection for hardware outages, set the `allow-colocation` attribute to `false`. This will require the state replication manager to select a peer (or peers if `write-quota` is greater than 1) running on a separate physical node (or nodes) to store its state replicas.

To replicate state to all JVMs within the Oracle Application Server cluster, you must specify the total number of JVMs within the cluster as the value of write-quota.

## Using Synchronous or Asynchnronous Replication

By default, OC4J instances will replicate data to other instances asynchronously. However, you can enable synchronous replication by including the <synchronous-replication> subelement within the <cluster> element. This will force a replicating OC4J instance to wait for an acknowledgement that the data was received from at least one other peer instance before continuing with replication.

## Configuring Multicast Replication

Multicast IP replication is the default replication protocol used in a standalone OC4J installation. 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 to be picked up by other OC4J processes using the same multicast address and port. Lost messages are identified and retransmitted, providing a reliable transmission service.

The configuration must specify the same multicast address and port on all OC4J instances. The default values used by OC4J multicast are 230.230.0.1 for the address and 45566 for the port. These values can be changed in the appropriate XML configuration file, if necessary.

Multicast replication can be enabled between multiple application instances simply by adding an empty <cluster> element to orion-application.xml file for each instance:

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

The next example specifies a new multicast address and port, using the ip and port attributes. The optional bind_addr attribute can 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 want 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="225.130.0.0" port="45577" bind_addr="226.83.24.10" />
    </protocol>
   </cluster>
</orion-application>
```

### Using an Existing JavaGroups Configuration for Multicast Replication

The multicast-based 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 items in the `<property-config>` subelement within the `<cluster>` element:

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

See "Specifying the <cluster> Element" on page 9-11 for details.

## Configuring Peer-to-Peer Replication

OC4J supports replication in a peer-to-peer (P2P) topology, using TCP to establish connections between instances within an Oracle Application Server cluster. The state data held in each application instance is then unicast to each OC4J instance.

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 the default used in an Oracle Application Server environment where OPMN is used to manage the various components, including OC4J.

  See "Configuring Dynamic OPMN-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 supported only 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 OPMN-Managed Peer-to-Peer Replication

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

**Note:** To use this feature, all nodes hosting the application must first be members of a cluster utilizing either the OPMN dynamic multicast discovery or the static discovery server mechanism.

See "Supported Clustering Models" on page 8-2 for details.

Each node 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. In the event that a node is lost, another node is able to service its requests.

```
<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 of at least one other peer node are supplied to enable peer-to-peer communication. 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. The following elements and attributes affect the configuration:

- 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 communication. 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 element define the name of the node address and the port 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. The `range` attribute 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 the `orion-application.xml` application deployment descriptor deployed with the `sample` application to three cluster nodes.

In this configuration, each node specifies one other node as its peer. The result is that all of the nodes within the cluster are able to establish connections with one another. This scenario will work only if each node is started in succession; that is, `www3.company.com` must be started before `www2.company.com`. Otherwise, `www2.company.com` will not be able to "see" `www3.company.com`.

1. 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>
```

2. 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>
```

3. 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 `www2.company.com` as a peer. In this configuration, `www2.company.com` would have to be the first node started; the other nodes would then connect to this node, and establish connections with one another.

## Specifying Ports for State Replication in OPMN

When you deploy an application utilizing state replication in an OPMN-managed Oracle Application Server environment, OPMN dynamically allocates the ports that are used to propagate state across the cluster. You can restrict this allocation to a range of ports for an application that has peer-to-peer replication enabled. Specifying ports for state replication might be necessary in an installation with a firewall or network that uses a well-defined port range.

**To specify a range of ports for peer-to-peer state replication:**

1. Add a `<port>` element to an OC4J instance configuration in the `opmn.xml` file.
2. Specify the name of an application that has peer-to-peer replication enabled as the value of the `id` attribute of the `<port>` element.
3. Specify a range of ports in the `range` attribute of the `<port>` element.

For example, for deployment of an application named `rac-web` that is set up for peer-to-peer replication, the line labeled `<port id=rac-web .../>` in the following OC4J instance configuration tells OPMN to use ports 15213 to 15214 for state replication:

```
<port id="default-web-site" range="80-100" protocol="http"/>
<port id="rmi" range="12401-12500"/>
<port id="rmis" range="12701-12800"/>
<port id="jms" range="12601-12700"/>
<port id="rac-web" range="15213-15214"/>
```

## Configuring Database Replication

The new clustering framework provides the ability to replicate an HTTP session and stateful session bean state to a database. Data is persisted outside of the clustered OC4J framework, enabling the entire session to be recovered in the event of a catastrophic failure of all of the OC4J instances within the cluster. The full HTTP session or stateful session bean object is replicated to the database.

The connection to the database is created using a *data source*, which is specified in the `data-source` attribute of the `<database>` subelement of `<protocol>`. Set the value of the `data-source` attribute to the data source's `jndi-name` as specified in `data-sources.xml`.

The data source specified must already exist within the OC4J instance. See the *Oracle Containers for J2EE Services Guide* for details on creating and using data sources.

The following example configures the application to replicate data to the database accessed through the MyOracleDS data source.

```
<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) value. 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 "Configuring the OC4J Task Manager" on page 10-1 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.

## Determining an Application's JVM, OC4J Instance, and Application Server Instance

You can use system properties to determine the JVM, OC4J instance, and application server instance in which an application instance runs within a cluster. This information is useful for debugging as well as for building management utilities on top of Oracle Application Server.

To obtain information about the environment of an application instance running in a cluster, you can access several properties through `System.getProperty()` calls. In a Java process running on OC4J, you can use system properties to print information about the JVM, OC4J instance, and application server instance to the system console, as Example 9–1 shows.

***Example 9–1 System.getProperty() Calls to Print System Property Values***

```
System.out.println("Oracle home name: " + System.getProperty("oracle.home"));
System.out.println("OC4J instance name: " +
                    System.getProperty("oracle.oc4j.instancename"));
System.out.println("AS instance name: " +
                    System.getProperty("oracle.ons.instancename"));
System.out.println("Instance:Group:JVM PID: " +
                    System.getProperty("oracle.ons.indexid"));
```

Table 9–4 describes the system properties that specify the JVM, group, OC4J instance, Oracle home, and Oracle Application Server instance for an application.

***Table 9–4 System Properties for JVM, OC4J Instance, and Application Server Instance***

| Property | Value |
|---|---|
| `oracle.home` | A string containing the name of the physical directory in which Oracle Application Server is installed |
| `oracle.ons.instancename` | A string containing the name of the Oracle Application Server instance |
| `oracle.oc4j.instancename` | A string containing the name of the OC4J instance |
| `oracle.ons.indexid` | A string containing a combination of the OC4J instance name, the group to which the instance belongs, and the JVM executing the application instance, in the following format:<br>*`OC4J_INSTANCE.oc4j_groupname.jvm_number`*<br>For example: `java_ee1.javaee_group.2` |

## Disabling Clustering

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

## Specifying the <cluster> Element

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

### <cluster>

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

Subelements of `<cluster>`:

```
<property-config>
<flow-control-policy>
<replication-policy>
<protocol>
<synchronous-replication>
```

Attributes:

- enabled: Whether clustering is enabled for the application. The default is true. Setting this value at the application level overrides the value inherited from the parent application, including the default 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.

  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 default-web-site.xml, jms.xml, and rmi.xml configuration files.
- write-quota: The number of other application group members (JVMs) to which the application state should be replicated. This attribute makes it possible to reduce overhead by limiting the number of JVMs to which state is written, similar to the islands concept used in previous OC4J releases.

  The default is 1 JVM.

  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. In installations where heavy request loads are expected, this value should be increased; for example, to 5000. Setting this value higher also prevents the OC4J instance from creating a replica of session data within itself if allow-colocation is set to true.

  This attribute is ignored if the <database> tag is included.

### <property-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 9–1 on page 9-4 for the values for this attribute.

- `scope`: What data is replicated. See Table 9–2 on page 9-5 for the values for this attribute.

### <protocol>

Defines the mechanism to use for data replication. Only one mechanism 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 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`. Valid only for configuring static peer-to-peer replication in a standalone OC4J installation.
- `range`: The number of times to increment the port value specified in each <node> subelement while looking for a potential peer node. The default is `5` increments. Valid only for configuring static peer-to-peer replication in a standalone OC4J installation.
- `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 `3000` milliseconds. Valid only for configuring static peer-to-peer replication in a standalone OC4J installation.
- `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 host name 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 host name 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.

Attribute:

- `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` attribute, as specified in `data-sources.xml`.

### <flow-control-policy>

Controls the amount of memory to allocate to the handling of clustering messages during replication. This element is intended to prevent out-of-memory errors by gating the amount of data (bytes) sent from one node to another during replication.

Attributes:

- `enabled`: Whether flow control is enabled. The default is `true`.
- `max-bytes`: The maximum number of bytes the receiving node can accept. After this value is reached, the sending node must wait for an acknowledgement from the receiver before additional messages can be sent. The default value is `500000`.
- `min-bytes`: The minimum number of bytes the receiving node can accept without triggering an acknowledgement that more bytes should be sent. If the number of bytes received is less than this value, the receiver will acknowledge that it can accept more bytes from the sender. The default is `0`.
- `threshold`: If `min-bytes` is not specified, this factor value is applied to incoming requests to determine the value of that attribute. The default value is `0.25`.

### <synchronous-replication>

If included, a node that is replicating application data will wait for an acknowledgement that the data update was received from at least one other peer node before continuing with replication. This element is optional; the default behavior is for nodes to continue replicating data to other nodes asynchronously.

Attributes:

- `timeout`: The length of time, in milliseconds, to wait for a response from a peer node. If this value is exceeded, replication should continue, although no acknowledgement was received. The default value is 10000 milliseconds (10 seconds).

# 10

# Task Manager and Thread Pool Configuration

This chapter provides guidelines for configuring the task manager for an OC4J instance and configuring thread pools for OC4J instances and Web site applications. It contains the following sections:

- Configuring the OC4J Task Manager
- Configuring OC4J Thread Pools

## Configuring the OC4J Task Manager

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 executes every second (1000 milliseconds).

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 execute every minute (60000 milliseconds):

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

You must restart OC4J after making modifications to `server.xml`.

> **Note:** You can also set this parameter through the `granularity` attribute in the `TaskManager` MBean, which is accessible through the JMX Browser in Application Server Control.
>
> See Chapter 12, "Using MBeans in OC4J" for details on accessing and using MBeans to manage OC4J processes.

## Configuring OC4J Thread Pools

*Thread pools* create and store threads for use and reuse by an OC4J process and applications deployed to the OC4J instance. Reusing existing threads rather than creating new threads on demand improves performance and reduces the burden on the JVM and underlying operating system.

Table 10–1 lists the thread pools available in OC4J.

*Table 10–1 OC4J Thread Pools*

| Thread Pool | Description |
|---|---|
| `system` | For the OC4J runtime to use<br>The threads from this pool do not run any applications deployed on an OC4J instance.<br>You should not change the configuration of this thread pool. |
| `http` | Serves HTTP and AJP requests<br>If no `rmi request` thread pool exists, the `http` thread pool serves RMI requests.<br>If no `rmi connection` thread pool exists, the `http` thread pool handles RMI connections. |
| `jca` | Serves work management requests from resource adapters<br>If needed by a resource adapter deployed to an OC4J instance, a work management thread pool containing worker threads used by resource adapters, such as the JMS connector, is created within the OC4J process. |
| `rmi request` | Serves RMI requests<br>This optional thread pool provides more control over allocation of thread resources. |
| `rmi connection` | Handles RMI connections<br>This optional thread pool provides threads that block-read on the RMI connection.<br>The `rmi connection` pool is used not only for RMI connections but also for RMI listener threads and a JMS server thread. |
| Custom | For use by one or more applications<br>Separate, custom thread pools for applications reduce contention for thread resources. A set of applications in a cluster can share a custom thread pool. |

By default, three of these thread pools are created at OC4J startup:

- `system`
- `http`
- `jca`

In each thread pool, idle threads are reused before a new thread is spawned, unless the number of requests exceeds the number of available threads. After 10 minutes of inactivity, an idle thread is automatically destroyed.

A <thread-pool> or <custom-thread-pool> element in the `server.xml` file defines each thread pool. Table 10–2 summarizes the attributes of these elements and gives the default attribute values.

You can use the default thread pool configuration or change it. For each OC4J instance, you can change the attribute values for any of the thread pools except `system`, and you can and add `rmi request`, `rmi connection`, and one or more custom thread pools. The following topics describe how to configure thread pools:

- Changing the Thread Pool Configuration
- Configuring Custom Thread Pools for Applications

*Table 10–2 Attributes of <thread-pool> and <custom-thread-pool>*

| Attribute | Description |
|---|---|
| name | The name attribute specifies the thread pool name and has no default value.<br>For a custom thread pool, the name can be any string value.<br>In the <thread-pool> element, the name must be one of these values:<br>■ system<br>A thread pool for the OC4J runtime to use<br>■ http<br>A thread pool that serves HTTP and AJP requests and possibly RMI requests (if an rmi request thread pool is not configured) and RMI connections (if an rmi connection thread pool is not configured).<br>■ jca<br>The work management thread pool, for the J2CA work manager to serve resource adapter requests<br>■ rmi request<br>A thread pool that serves RMI requests<br>■ rmi connection<br>A thread pool whose threads block-read on the RMI connection<br>The names of the threads in these named pools are prefixed with SystemThreadGroup_, HTTPThreadGroup_, WorkManager_, RMIRequestThreadGroup_, and RMIConnectionThreadGroup_, respectively, and suffixed with an incrementing counter. |
| min | The minimum number of threads to create in the pool. The default value is 0.<br>The minimum number of threads for a jca thread pool 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.<br>The value of min for an rmi connection thread pool is relative to the number of physical connections you have at any point in time. The queue value handles bursts in connection traffic. |

*Table 10–2 (Cont.) Attributes of <thread-pool> and <custom-thread-pool>*

| Attribute | Description |
|---|---|
| `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 value is `1024`.<br><br>The `rmi connection` thread pool usually creates three threads for internal use as RMI and JMS listeners, so you need to set the value of `max` to your required maximum number of threads plus 3. For example, if you specify `max="16"`, then only 13 threads are available to service requests. Similarly, if the `max` value is `20`, then only 17 threads are available.<br><br>The value of `max` for an `rmi connection` thread pool is also relative to the number of the physical connections you have at any point in time. The `queue` value handles bursts in connection traffic.<br><br>The maximum number of threads for a `jca` thread pool 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. |
| `queue` | The maximum number of requests that can be kept in the queue. The default value is `0`.<br><br>The `queue` value should be at least twice the size of the maximum number of threads. |
| `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. The default value is `600000`.<br><br>To never destroy threads, set to `-1`.<br><br>The default value, `600000` milliseconds (10 minutes), is also the minimum value allowed if the value is not `-1`.<br><br>Setting `keepAlive` to `0` (zero) will cause high CPU usage due to active polling. |
| `stackSize` | The size of the thread pool stack. The default value is `0`. |
| `debug` | A value of `true` specifies printing the thread pool information for the application server to the console at startup. The default is `false`. If `debug` is `false`, the thread pool information is not printed. |

## Changing the Thread Pool Configuration

You can change the thread pool configuration for an OC4J instance with Application Server Control or by editing the `server.xml` file, in the following ways:

- Change attribute values for thread pools on the Thread Pool Configuration page in Application Server Control.
- Change the attributes of thread pool MBeans through the System MBean Browser in Application Server Control.

  See "Using the System MBean Browser" on page 12-5 for details on accessing and using MBeans to manage OC4J.
- Configure an `rmi request` or `rmi connection` thread pool, or both, by adding a `<thread-pool>` element for each to `server.xml`.

  You must restart OC4J after making modifications to `server.xml`.

---

**Note:** Configuring thread pools or modifying the default configuration are expert-mode tasks. Oracle recommends that you use the default thread pool configuration unless you need to change it.

---

### Changing the Thread Pool Configuration with Application Server Control

To change the thread pool configuration for an OC4J instance with Application Server Control:

1. Navigate to the OC4J Home page and then click **Administration** to display the OC4J Administration page, which contains a table listing the various administration tasks you can perform for the OC4J instance.
2. Under **Properties** on the Administration page, select **Thread Pool Configuration** to display the OC4J Thread Pool Configuration page.
3. Change the value of one or more attributes for any thread pool displayed on this page.

   For information about attribute values, see Table 10–2 on page 10-3 or "<thread-pool>" on page B-18.
4. Click **Apply**.
5. Restart the OC4J instance from the Cluster Topology page or the OC4J Home page.

You can also configure thread pools in Application Server Control through MBeans, as "Changing the Thread Pool Configuration Through MBeans" describes in the following text.

### Changing the Thread Pool Configuration Through MBeans

To change the attributes of thread pool MBeans with Application Server Control:

1. Navigate to the OC4J Home page and then click **Administration** to display the OC4J Administration page, which contains a table listing the various administration tasks you can perform for the OC4J instance.
2. Under **JMX** on the Administration page, select **System MBean Browser** to display the OC4J System MBean Browser page., which displays the system MBeans exposed by the OC4J instance.
3. Expand the navigation tree on the left of the page and select a thread pool for the OC4J instance under ThreadPool.
4. Change any attributes of the thread pool that have edit boxes.

   For information about attribute values, see Table 10–2 on page 10-3 or "<thread-pool>" on page B-18.
5. Click **Apply**.

### Adding <thread-pool> Elements to server.xml

The following example uses the <thread-pool> element to configure an `rmi request` thread pool in the `server.xml` file:

```
<thread-pool
   name="rmi request"
   min="50"
   max="50"
   queue="2560"
   keepAlive="-1"
```

```
   stackSize="0"
   debug="true"/>
```

With this configuration, OC4J will create a separate thread pool to serve RMI requests. The thread pool will have these attributes:

- A minimum of 5 threads
- A maximum of 50 threads
- A maximum of 2560 requests in the queue
- A `keepAlive` value of -1 (no timeout)
- A `stackSize` value of 0 (let the JVM decide)
- The `debug` attribute set to true

The following example shows `<thread-pool>` elements that configure separate thread pools in `server.xml`, one to serve RMI requests, one to handle RMI connections, and one to serve HTTP and AJP requests:

```
<thread-pool
   name="rmi request"
   min="50"
   max="50"
   queue="2560"
   keepAlive="-1"
   stackSize="0"/>

<thread-pool
   name="rmi connection"
   min="44"
   max="44"
   queue="2560"
   keepAlive="-1"
   stackSize="0"/>

<thread-pool
   name="http"
   min="40"
   max="40"
   queue="2560"
   keepAlive="-1"
   stackSize="0"/>
```

The `http` thread pool is created by default when an OC4J instance starts up, with the default attribute values in Table 10–2 on page 10-3. In addition to serving HTTP and AJP requests, this tread pool can serve RMI requests and handle RMI connections in the absence of separate `rmi-*` thread pools.

> **Note:** You must restart OC4J after making modifications to `server.xml`.

## Configuring Custom Thread Pools for Applications

You can create a separate, custom thread pool for your applications to use in an OC4J instance by adding a `<custom-thread-pool>` element to the `server.xml` file. Then you can make the custom thread pool available to an application by referring to the thread pool in the `custom-thread-pool` attribute of the `<web-site>` element in the `*-web-site.xml` file for the application. A `server-xml` file can include more than one <custom-thread-pool> element, and you can configure more than one application to use each custom thread pool.

In server.xml, the <custom-thread-pool> element is a subelement of the <application-server> element and has the same attributes as the <thread-pool> element, except that the value of name is not restricted. For example:

```
<custom-thread-pool name="mypool" min="3" />
```

The name attribute is required, and all other attributes are optional. For a complete description of this element, see "<custom-thread-pool>" on page B-9.

For information about the <thread-pool> element, see "Changing the Thread Pool Configuration" on page 10-4 and "<thread-pool>" on page B-18.

For information about the *-web-site.xml file, see "Overview of the Web Site Configuration File (*-web-site.xml)" on page B-21. The custom-thread-pool attribute is described in Table B–25 on page B-21.

The following example configures an HTTP Web site to use a nondefault thread pool by adding the custom-thread-pool attribute to the <web-site> element in the default-web-site.xml file for the HTTP Web site:

```
<web-site xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/web-site-10
_0.xsd"
       protocol="http"
       port="8888"
       custom-thread-pool="mypool1"
       display-name="OC4J 10g (10.1.3) Default Web Site"
       schema-major-version="10"
       schema-minor-version="0"
     />
```

## Converting from the Older Thread Pool Format

The <global-thread-pool> and <work-manager-thread-pool> elements in server.xml configure thread pools in an older format. These elements are deprecated. Table B–7 on page B-10 and Table B–24 on page B-20 describe the attributes of these elements.

If a server.xml file contains a <global-thread-pool> or <work-manager-thread-pool> element, OC4J updates the older element format to the new format in server.xml. For example, suppose a server.xml file contains the following elements:

```
<global-thread-pool
   min="60"
   max="60"
   queue="20000"
   keepAlive="-1" />

   <work-manager-thread-pool
   min="23"
   max="24"
   queue="5000"
   keepAlive="-1" />
```

After OC4J startup, instead of the <global-thread-pool> and <work-manager-thread-pool> elements, the server.xml file will contain the following <thread-pool> elements:

```
<thread-pool
   name="http"
   min="60"
   max="60"
   queue="20000"
   keepAlive="-1"
   stackSize="0" />

<thread-pool
   name="jca"
   min="23"
   max="24"
   queue="5000"
   keepAlive="-1" />
```

Table 10–3 shows how the attributes of `<global-thread-pool>` and `<work-manager-thread-pool>` map to the new thread pools introduced in OC4J 10*g* (10.1.3.1.0).

***Table 10–3 Mapping of Old Thread Pool Configuration to New Thread Pools***

| Old Thread Pool Attributes | Value of name Attribute in <thread-pool> | New Thread Pool Attributes |
|---|---|---|
| `min`, `max`, `queue`, `keepAlive`, and `debug` attributes of `<global-thread-pool>` | `http` | `min`, `max`, `queue`, `keepAlive`, and `debug` attributes of `<thread-pool>` |
| `min`, `max`, `queue`, `keepAlive`, and `debug` attributes of `<work-manager-thread-pool>` | `jca` | `min`, `max`, `queue`, `keepAlive`, and `debug` attributes of `<thread-pool>` |
| `cx-min`, `cx-max`, `cx-queue`, `cx-keepAlive`, and `cx-debug` attributes of `<global-thread-pool>` | `rmi request` | `min`, `max`, `queue`, `keepAlive`, and `debug` attributes of `<thread-pool>` |
| `rmiRequest-min`, `rmiRequest-max`, `rmiRequest-queue`, `rmiRequest-keepAlive`, and `rmiRequest-debug` attributes of `<global-thread-pool>` | `rmi connection` | `min`, `max`, `queue`, `keepAlive`, and `debug` attributes of `<thread-pool>` |

For example, OC4J would generate new `<thread-pool>` elements from the following `<global-thread-pool>` element:

```
<global-thread-pool
   keepAlive="-1"
   debug="false"
   cx-keepAlive="-1"
   cx-debug="false"
   rmiRequest-keepAlive="-1"
   rmiRequest-debug="false"
   min="40"
   max="40"
   queue="2560"
   cx-min="44"
   cx-max="44"
```

```
   cx-queue="2560"
   rmiRequest-min="50"
   rmiRequest-max="50"
   rmiRequest-queue="2560"/>
```

The equivalent `<thread-pool>` elements follow:

```
<thread-pool
   name="rmi request"
   min="50"
   max="50"
   queue="2560"
   keepAlive="-1"
   stackSize="0"/>

<thread-pool
   name="rmi connection"
   min="44"
   max="44"
   queue="2560"
   keepAlive="-1"
   stackSize="0"/>

<thread-pool
   name="http"
   min="40"
   max="40"
   queue="2560"
   keepAlive="-1"
   stackSize="0"/>
```

# 11

# Logging in OC4J

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

- Overview of Log Files Generated by OC4J
- Using Plain Text File Logging
- Using Oracle Diagnostic Logging (ODL)
- Configuring OC4J Logging
- Viewing Application Messages in the OC4J Log with LogViewer
- Configuring Application Loggers with Application Server Control
- Redirecting log4j Messages for an Application to the OC4J Log

## Overview of Log Files Generated by OC4J

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

OC4J can generate two types of log files:

- **Plain text log files**

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

- **Oracle Diagnostic Logging (ODL) log files**

  The messages logged in these files use an XML format that is viewable with Application Server Control. ODL supports log file rotation.

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

Table 11–1 lists the names and locations of the various log files generated, as well as the XML configuration file containing the logging configuration for each component. Unless otherwise indicated, all paths indicated are within `ORACLE_HOME/j2ee/home` for standalone OC4J or `ORACLE_HOME/j2ee/instance` for OPMN-managed OC4J instances.

***Table 11–1 List of Log Files Generated for OC4J***

| Component | Configuration File | Default Log File Name and Location |
|---|---|---|
| OC4J component using Java logging | See "Configuring OC4J Logging" on page 11-8 for details on configuring this ODL-formatted log file. | Standalone OC4J:<br>`/log/oc4j/log.xml`<br><br>OPMN-managed OC4J:<br>`/log/`*`instance`*`_default_group_1/oc4j/log.xml` |
| Application Server Control | `/application-deployments/ascontrol/orion-application.xml` | Standalone OC4J:<br>`/log/ascontrol-application.log`<br><br>OPMN-managed OC4J:<br>`/log/`*`instance`*`_default_group_1/ascontrol-application.log` |
| Application deployed to OC4J | `/application-deployments/`*`app_name`*`/orion-application.xml` | Standalone OC4J:<br>`/application-deployments/`*`app_name`*`/application.log`<br><br>OPMN-managed OC4J:<br>`/application-deployments/`*`app_name`*`/`*`instance`*`_default_group_1/application.log` |
| Global (`default`) application | `/config/application.xml` | Standalone OC4J:<br>`/log/global-application.log`<br><br>OPMN-managed OC4J:<br>`/log/`*`instance`*`_default_group_1/global-application.log` |
| Default Web site access logging | `/config/default-web-site.xml` | Standalone OC4J:<br>`/log/default-web-access.log`<br><br>OPMN-managed OC4J:<br>`/log/`*`instance`*`_default_group_1/default-web-access.log` |
| OC4J server | `/config/server.xml` | Standalone OC4J:<br>`/log/server.log`<br><br>OPMN-managed OC4J:<br>`/log/`*`instance`*`_default_group_1/server.log` |

*Table 11–1 (Cont.) List of Log Files Generated for OC4J*

| Component | Configuration File | Default Log File Name and Location |
|---|---|---|
| JMS | `/config/jms.xml` | Standalone OC4J:<br>`/log/jms.log`<br><br>OPMN-managed OC4J:<br>`/log/`*`instance`*`_default_group_1/jms.log` |
| RMI | `/config/rmi.xml` | Standalone OC4J:<br>`/log/rmi.log`<br><br>OPMN-managed OC4J:<br>`/log/`*`instance`*`_default_group_1/rmi.log` |
| OPMN | *`ORACLE_HOME`*`/opmn/conf/opmn.xml` | *`ORACLE_HOME`*`/opmn/logs` |

Through the `oc4j.properties` file, you can configure an OC4J instance to generate trace files to a specific debug destination instead of the default destination. Oracle Application Server does not support the configuration of two different OC4J instances to generate trace output to the same destination, even if the instances are in the same group. Each OC4J instance manages its own trace files.

# 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. The following topics describe how to use text logging:

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

## Enabling or Disabling Text File Logging

Text logging is enabled or disabled through elements in the XML configuration files listed in Table 11–1, except for the `default-web-site.xml` file. (See "Configuring Web Site Access Logging" on page 13-13 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 single `path` attribute which specifies the name and optionally the location of the log file generated:

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

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 you enable ODL, both logging options will be enabled.

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

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

Although both 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 files, and 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 11–1 on page 11-2. Text log files are identified by the `log` extension.

Text log files generated for OC4J components can be viewed with Application Server Control, as follows:

1. Click the **Logs** link at the bottom of any Application Server Control page.
2. Expand **OC4J**.
3. Expand **<instanceName>**. The default instance name is `home`.

Text log files for deployed J2EE applications cannot be viewed with Application Server Control.

## 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. In ODL, unlike in text-based logging, log file rotation is supported.

- Enabling or Disabling ODL
- Managing ODL Log Files
- Viewing ODL Log Files

### Enabling or Disabling ODL

ODL is enabled by adding the `<odl>` element within the `<log>` element in any of the XML files listed in Table 11–1.

**Notes:**

- You can enable ODL 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.

- ODL for Web sites uses a different configuration. For more information about this configuration, see "Configuring Web Site Access Logging" on page 13-13.

- Both ODL and text file logging can be enabled simultaneously. However, you should disable one of these options to save disk space.

The `<odl>` element has the following attributes. All are required.

- `path`: The path to the directory where the `log.xml` files for this component will be generated.

**Important:**

Specify the path as `../log/`*`appName`*, as the next example shows. This path is required for viewing log files with Application Server Control.

- `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.

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. It will also set log files to a maximum of 1,000 KB and the directory maximum to 10,000 KB.

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

Using this configuration, `petstore` log files will be generated in the following locations, depending on your OC4J installation.

- Standalone OC4J:

  Log files will be generated in *`ORACLE_HOME`*`/j2ee/home/application-deployments/log/petstore`.

- OPMN-managed OC4J:

  Files will be generated in an OC4J instance- specific directory named *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application-deployments/log/`*`instance_`*`default_group_1/petstore`.

## Managing ODL Log Files

The ODL framework 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. In addition, using ODL provides these benefits:

- You can limit the total amount of diagnostic information saved.
- Older segment files are removed and newer segment files are saved in chronological fashion.
- Components can remain active and do not need to be shut down when diagnostic logging files are cleaned.

An **ODL log** is a set of log files that includes the current log file, `log.xml`, and zero or more **ODL Archives (segment files)**, which contain older messages. After you enable ODL, each new message is added to the end of `log.xml`. When this log file reaches the rotation point, it is renamed and a new `log.xml` file is created.

Segment files are created when the current log file reaches the rotation point, specified by the maximum ODL segment size, and for some OC4J logs, the rotation time and rotation frequency. The`log.xml` file is renamed to `log`*n*`.xml`, in which *n* is an integer, starting at 1. The new `log.xml` file is created when the component generates new diagnostic messages.

When the last log file is full, the following procedure 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, in which *n* increments by one over the most recent log file.

### Size-Based Log Rotation

To limit the size of an ODL log for an application or component, you can use a configuration option specifying the maximum size of the logging directory. Whenever the sum of the sizes of all of the files in the directory reaches the maximum, the oldest archive is deleted to keep the total size under the specified limit.

> **Note:** The most recent segment file is never deleted.

For example, when the maximum directory size is reached, with the starting segment file named `log9872`, the following files could be present in the log file directory:

```
File                    Size

log.xml                 10002
log9872.xml             15000
log9873.xml             15000
log9874.xml             15000
log9875.xml             15000
log9876.xml             15000
```

In this case, when `log.xml` fills up, `log9872.xml` is removed and `log.xml` is moved to the new file `log9877.xml`. New diagnostic messages are then written to a new `log.xml` file.

For example, to specify the maximum ODL segment size and maximum directory size for an OC4J application named `petstore`, you would add the following entry to the file *`ORACLE_HOME`*`/j2ee/`*`instance_name`*`/application-deployments/petstore/orion-application.xml`:

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

For OC4J components that are configured in the `j2ee-logging.xml` file, you can specify a rotation time and rotation frequency, in addition to a maximum segment size and directory size.

### Time-Based Log Rotation

For time-based log rotation, you can specify the following properties in a <log_handler> subelement of the <log-handlers> element of the <logging-configuration> root element:

- `baseRotationTime`: (Optional.) The base time for the rotation. The format for the base time can be any of the following:
  - `hh:mm`, for example, 04:20. This format uses the local time zone.
  - `yyyy-MM-dd`, for example, 2006-08-01. This format uses the local time zone.
  - `yyyy-MM-ddThh:mm`, for example 2006-08-01T04:20. This format uses the local time zone.
  - `yyyy-MM-ddThh:mm:ss.sTZD`, where TZD is the timezone indicator. TZD can be `Z`, indicating UTC, or `{+|-}hh:mm`. For example, 2006-03-01T04:20:00-08:00 represents March 1, 2006 4:20:00 in US Pacific Standard Time time zone.

  If you do not specify `baseRotationTime`, the default value is Jan. 1, 1970, 00:00 UTC.
- `rotationFrequency`: The frequency of the rotation, in minutes. In addition, you can specify one of the following values: `hourly`, `daily`, or `weekly`.

You specify these properties in the following file:

```
ORACLE_HOME/j2ee/instance_name/config/j2ee-logging.xml
```

For example, to specify that the log files are rotated every day at 4:00AM local time, or when they reach 2000000 bytes in size, use the following:

```
<log_handler name="h1" class="oracle.core.ojdl.logging.ODLHandlerFactory">
     <property name="path" value="log"/>
     <property name="baseRotationTime" value="04:00"/>
     <property name="rotationFrequency" value="daily"/>
     <property name="maxFileSize" value=" 2000000"/>
</log_handler>
```

## Viewing ODL Log Files

ODL-formatted log files can be viewed by clicking the **Logs** link in Application Server Control, enabling 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.

1. Click the **Logs** link at the bottom of any Application Server Control page.
2. Expand **OC4J**.
3. Expand **<instanceName>**. In both standalone OC4J and OAS, the default instance name is `home`.

- To view the OC4J log files, expand **Diagnostic Message Logs**, then open `log.xml`.
- To view ODL logs for a specific J2EE application:
  - Expand **Application <applicationName>** .
  - Expand **Diagnostic Message Logs**. Open and view the `log.xml` file generated within this director.

# Configuring OC4J Logging

The various components of OC4J utilize Java loggers that write to the OC4J log file. The OC4J log file is generated in XML format using the Oracle Diagnostic Logging framework and can be viewed with Application Server Control.

The section covers the following topics:

- Using and Configuring the OC4J Component Loggers
- Viewing the OC4J Log File
- Configuring the oracle Logger

## Using and Configuring the OC4J Component Loggers

OC4J provides a number of component loggers that write to the OC4J log file (`log.xml`). You can view and configure the available component loggers on the Logger Configuration page of Application Server Control.

The Java log level can be set for each individual component logger. If set to `NULL`, a logger inherits the log level set for its parent.

Therefore, the default level for all loggers is `INFO`, which maps to the `NOTIFICATION` Java log level, as that is the default value inherited from the `oracle` logger. See "Configuring the oracle Logger" on page 11-9 for details on changing this default value.

The log level set on a logger through the Logger Configuration page is not persisted, but is applied to the OC4J runtime only. When OC4J is restarted, the log level reverts back to the default setting inherited from the parent logger.

Table 11–2 illustrates the log levels that you can set with Application Server Control as well as the ODL *`message type:log level`* that each Java log level maps to.

***Table 11–2 Log Levels for OC4J Component Loggers***

| Java Log Level | ODL Message Type:Log Level | ODL Description |
|---|---|---|
| `NULL` | | The logger will inherit the log level set for its parent. |
| `SEVERE` | `ERROR:1` | Log system errors requiring attention from the system administrator. |
| `WARNING` | `WARNING:1` | Log actions or conditions discovered that should be reviewed and may require action before an error occurs. |

*Table 11–2 (Cont.) Log Levels for OC4J Component Loggers*

| Java Log Level | ODL Message Type:Log Level | ODL Description |
|---|---|---|
| INFO | NOTIFICATION:1 | Log normal actions or events. This could be a user operation, such as *login completed*, or an automatic operation, such as a *log file rotation*. |
| CONFIG | NOTIFICATION:16 | Log configuration-related messages or problems. |
| FINE | TRACE:1 | Log trace or debug messages used for debugging or performance monitoring. Typically contains detailed event data. |
| FINER | TRACE:16 | Log fairly detailed trace or debug messages. |
| FINEST | TRACE:32 | Log highly detailed trace or debug messages. |

#### To configure OC4J component loggers with Application Server Control:

1. Click the **Administration** link.
2. Click **Logger Configuration**.
3. Set Log Level to a value listed in the left-hand column of Table 11–2.
4. Click **Apply** to apply your changes to the OC4J runtime.

## Viewing the OC4J Log File

The OC4J log file can be viewed with Application Server Control. To view the file:

1. Click the **Logs** link at the bottom of any Application Server Control page.
2. Expand **OC4J**.
3. Expand **<instanceName>**.
4. Expand **Diagnostic Message Logs**.

As with all ODL log files, each new message goes into the current log file, named `log.xml`. Once the maximum size is reached, the log is copied to an archival log file, named `log`*n*`.xml`, in which *n* is an integer starting at 1.

## Configuring the oracle Logger

The configuration for the `oracle` logger is defined in `j2ee-logging.xml`, which is installed in the *ORACLE_HOME*`/j2ee/`*instance*`/config` directory.

#### To configure the oracle logger with Application Server Control:

1. On the OC4J Home page, click **`Administration`**.
2. From the administration tasks, select **`Logger Configuration`** to display the Logger Configuration page.
3. Click **`Expand All`** to view the entire list of loggers currently loaded for the OC4J instance.
4. Select a log level for any of the loggers shown on the page.

**To configure the oracle logger in j2ee-logging.xml:**

You can also edit the `j2ee-logging.xml` configuration file by hand. Restart OC4J after making any changes to this file.

The configuration file contains two elements within the `<logging-configuration>` root element:

- `<log_handlers>`

  This element includes `<log_handler>` elements defining three different log handlers:

  - `oc4j-handler`

    This is the log handler for the `oracle` logger.

  - `oracle-webservices-management-auditing-handler`

    This is the log handler for the oracle.webservices.management.auditing logger.

  - `oracle-webservices-management-logging-handler`

    This is the log handler for the oracle.webservices.management.logging logger.

  The following properties are specified in `<property>` subelements for each log handler:

  - `path`: Specifies the directory in which the handler will generate log files. Do not modify this value.
  - `maxFileSize`: Sets the maximum size, in bytes, that any log file in the directory will be allowed to grow to. When a file exceeds this limit, a new file is generated.
  - `maxLogSize`: Sets the maximum size, in bytes, allowed for the log file directory. When this limit is exceeded, log files are purged, beginning with the oldest files.

- `<loggers>`

  This element includes a `<logger>` element defining the following:

  - `name`: The logger name. Do not modify this value.
  - `level`: The minimum log level that this logger acts upon. This level is set by default to the ODL `NOTIFICATION:1` value, which maps to the `INFO` Java log level displayed on the Logger Configuration page in Application Server Control.

    You can set this value to either a Java logging level (`FINE`) or an ODL Message Type:Log Level (`TRACE:1`).

  - `useParentHandlers`: Indicates whether or not the logger should use its parent handlers. Because this value is set to `false` by default, the `oracle` logger does not inherit the log level set for its parent, the `root` logger.
  - `<handler>`: The name of the handler to use. Do not modify this value.

The following example sets the default log level to `FINEST` by specifying `TRACE:32` as the ODL Message Type:Log Level.

```
<logging_configuration>
  <log_handlers>
    <log_handler name='oc4j-handler'
        class='oracle.core.ojdl.logging.ODLHandlerFactory'>
      <property name='path' value='%ORACLE_HOME%/j2ee/%OPMN_PROC_TYPE%/log/
          %OPMN_PROC_TYPE%_%OPMN_PROC_SET%_%OPMN_PROC_INDEX%/oc4j'/>
```

```
      <property name='maxFileSize' value='10485760'/>
      <property name='maxLogSize' value='104857600'/>
    </log_handler>
  </log_handlers>
  <loggers>
    <logger name='oracle' level='TRACE:32' useParentHandlers='false'>
      <handler name='oc4j-handler'/>
    </logger>
  </loggers>
</logging_configuration>
```

## Viewing Application Messages in the OC4J Log with LogViewer

You can direct application log messages into the OC4J logging system and view them in Application Server Control with LogViewer. If you use the standard JDK logging API, you can configure logging handlers and log levels to direct the log entries from an application into `ORACLE_HOME/j2ee/instance/log/oc4j/log.xml`, the main OC4J log. LogViewer lists this log in Application Server Control as **Diagnostic Logs**.

**To view application messages in the OC4J log with LogViewer:**

1. In an application, set up a logger to issue log messages at different levels, using a logger naming hierarchy, as follows.

***Example 11–1 Application Logger to Set Up Different Log Levels***

```
Logger logger = Logger.get("emp.app.web.EmployeeFrontEnd");
public void doGet(HttpServletRequest request,
          HttpServletResponse response) throws ServletException,
          IOException {response.setContentType(CONTENT_TYPE);
logger.fine(
String.format("Handling web request for %s", request.getRequestURL()));
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head><title>EmployeeFrontEnd</title></head>");
out.println("<body>");
Employee test = Employee.getTestInstance();
logger.finest(
String.format("Test Employee Instance: %s", test));
logger.finest(
String.format("Calling %s to locate office for %s",
          employeeManager.toString(),
          test.identifier(test.ID_SHORT)));
String location = employeeManager.locateEmployeeOffice(test);
logger.finest(
String.format("bean returned %s for %s ",
          location, test.identifier(test.ID_SHORT)));
out.printf("<p>Employee: %s</br>Office: %s</p>", test.identifier(test.ID_SHORT),
location);
logger.fine(String.format("Employee currently earns $%s", test.getSalary()));
test.raiseSalary(15D);
out.printf("<p>Give employee 15percent raise, now earns %s", test.getSalary());
out.println("</body></html>");
out.close();
}
```

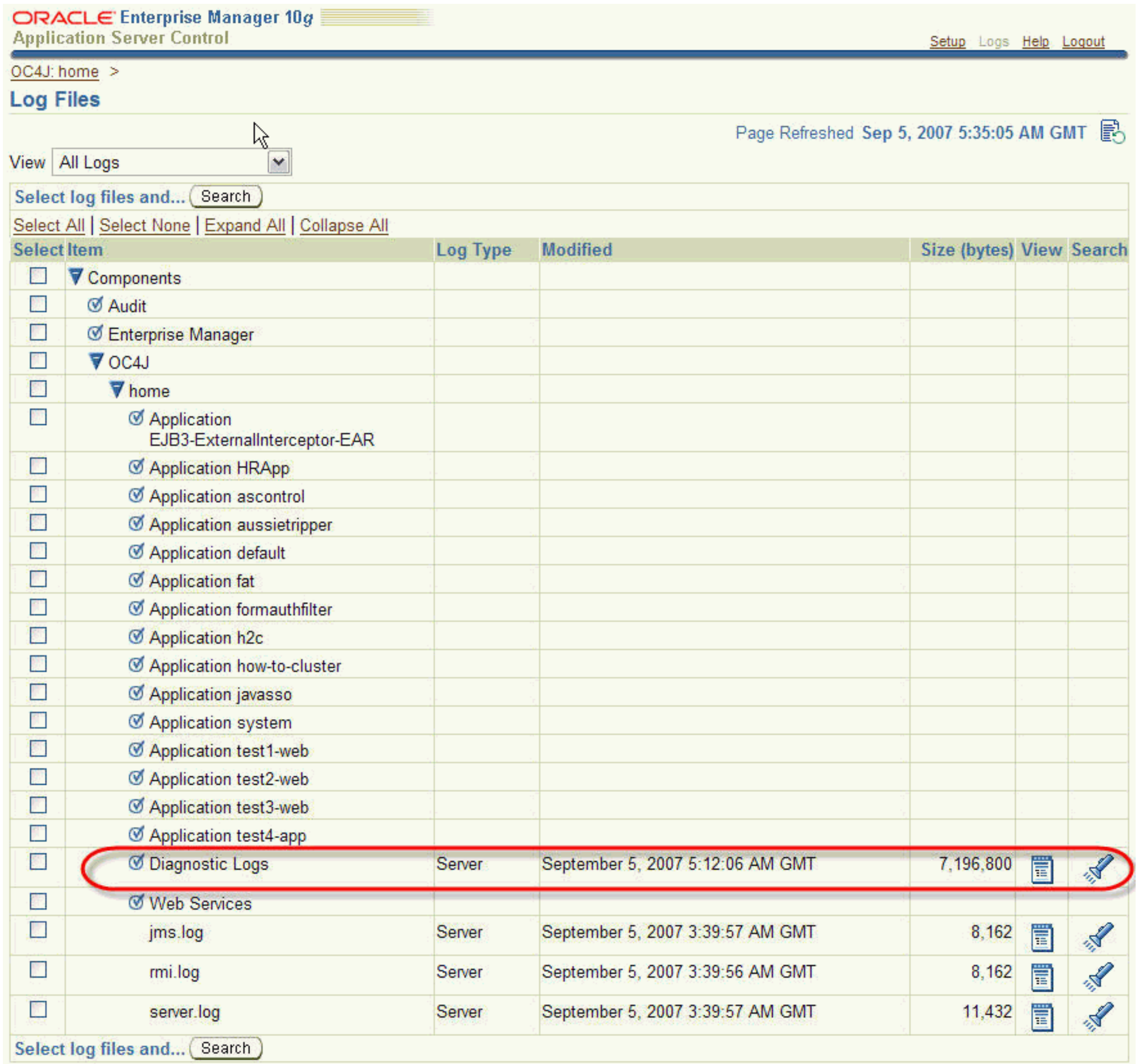
2. In the `<logging_configuration>` element of the `j2ee-config.xml` file, add a new `<logger>` element that specifies the application logger name and logging level and declares the logger to use `oc4j-handler`, as in the following example:

```
<logging_configuration>
  <log_handlers
    <log_handler name="oc4j-handler"
           class="oracle.core.ojdl.logging.ODLHandlerFactory">
      <property name="path" value="../log/oc4j"/>
      <property name="maxFileSize" value="10485760"/>
      <property name="maxLogSize" value="104857600"/>
      <property name="encoding" value="UTF-8"/>
      <property name="supplementalAttributes" value="J2EE_APP.name,
           J2EE_MODULE.name,WEBSERVICE.name,WEBSERVICE_PORT.name"/>
    </log_handler>
  </log_handlers>
  <loggers>
    <logger name="oracle" level="NOTIFICATION:1" useParentHandlers="false">
      <handler name="oc4j-handler"/>
      <handler name="console-handler"/>
    </logger>
    <logger name="emp" level="FINEST
      <handler name="oc4j-handler"/>
    </logger>
  </loggers>
</logging_configuration>
```

Any messages written to the `oracle` root logger will be directed to `oc4j-handler`, which writes messages out in XML format to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/log/oc4j/log.xml` file.

3. Click the **Logs** link at the bottom of any Application Server Control page to list all of the available logs, as Figure 11–1 shows.

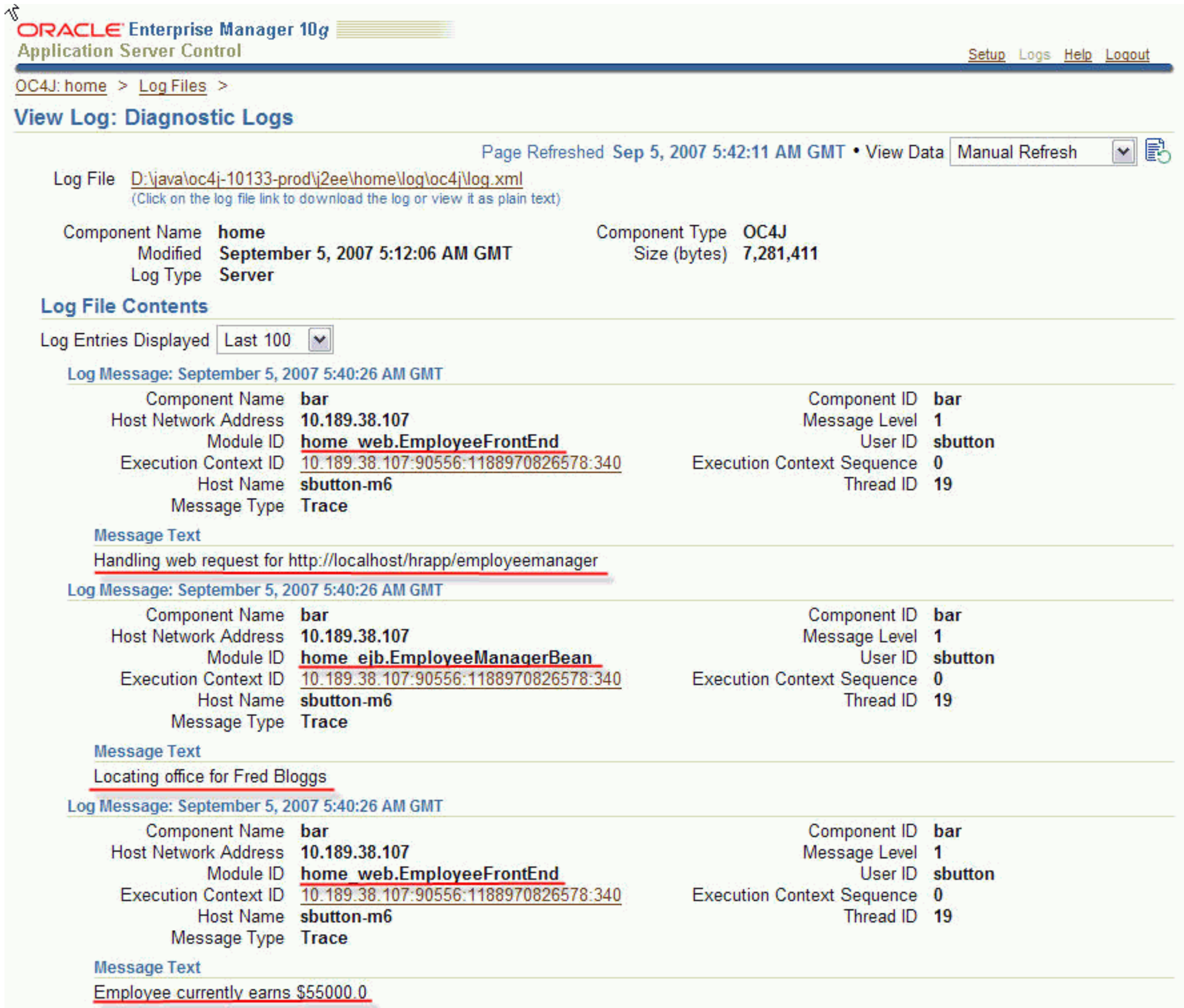
***Figure 11–1 Logger List in Application Server Control***




4. On the Log Files page, click **Diagnostic Logs** to have LogViewer display messages from the OC4J log.

   As Figure 11–2 shows, the Diagnostic Logs page displays log entries for the OC4J server plus log entries from the application. Each log message shows the component that generated the log entry, such as `web_EmployeeFrontEnd` or `ejb_EmployeeManagerBean`.

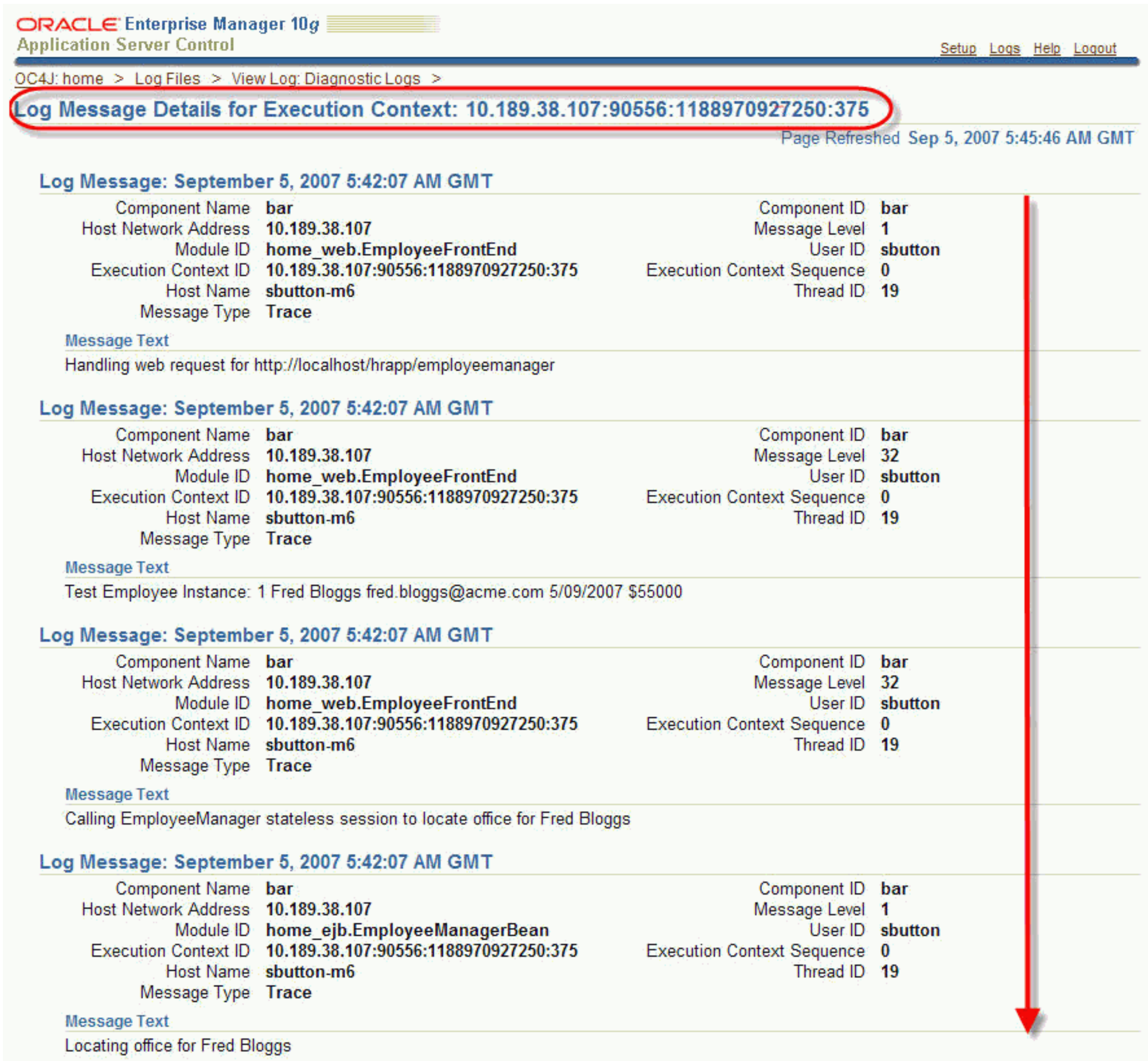
***Figure 11–2 LogViewer Diagnostic Log Page***




5. To view all the log entries for an individual request in sequence, click the Execution Context ID (ECID) link for a log entry.

   Based on an ECID assigned to every log message, the Oracle logging framework can correlate various log entries from different components in the same execution path. As Figure 11–3 shows, LogViewer displays all the log entries for the selected ECID in timestamp order.

*Figure 11–3 Log Messages for an Execution Context Displayed in Sequence*




6. Use the LogViewer search facilities to search for items of interest in the OC4J log.

## Configuring Application Loggers with Application Server Control

After an application has run and its loggers have registered themselves, or after you have configured them in the `j2ee-config.xml` file, you can use Application Server Control to configure the application loggers. The Logger Configuration page enables you to customize application logging quickly.

**To configure application loggers with Application Server Control:**

1. On the OC4J Home page, click **`Administration`**.
2. From the administration tasks, select **`Logger Configuration`** to display the Logger Configuration page.
3. Click **`Expand All`** to view the entire list of loggers currently loaded for the OC4J instance.

The OC4J loggers and application loggers are listed, along with a select list that enables you to specify the level for each logger.

4. Select a log level for any of the loggers listed on the page.

## Redirecting log4j Messages for an Application to the OC4J Log

The Oracle Application Server distribution includes a JAR file, *ORACLE_HOME*/diagnostics/lib/ojdl-log4j.jar, that contains an OracleAppender class. This class is the Oracle log4j appender, which transforms log4j messages into the Oracle Diagnostic Logging (ODL) XML format. Before an application can use this appender, you need to configure it and make the log4j class libraries available to the application.

A log4j.properties file can configure the appropriate log4j settings and the Java log level you want to enable. You can capture and route your log4j entries into the OC4J log system with OracleAppender, as Example 11–2 shows:

***Example 11–2 Configuration File for the Oracle log4j Appender***

```
log4j.rootLogger=TRACE,OJDL
log4j.appender.OJDL=oracle.core.ojdl.log4j.OracleAppender
log4j.appender.OJDL.LogDirectory=${oracle.j2ee.home}/log/oc4j
#log4j.appender.APP1.MaxSize=1000000
#log4j.appender.APP1.MaxSegmentSize=200000
#log4j.appender.APP1.Encoding=iso-8859-1
log4j.appender.OJDL.ComponentId=OracleProd
```

This configuration directs log4j messages into the *ORACLE_HOME*/j2ee/home/log/oc4j/log.xml file, which LogViewer reads and then displays in Application Server Control as **Diagnostics Logs**.

After you configure the Oracle log4j appender, you can use the OC4J shared-library mechanism to publish the log4j class libraries to OC4J and then import the libraries into the application.

### Configuring the Oracle log4j Appender for an Application

You can configure the Oracle log4j appender for an application by using the OracleAppender class in the log4j.properties file. Then you can import the configuration into the application during deployment.

To inject the log4j.properties file into an application when you deploy it, you can use the OC4J shared-library mechanism. OC4J reads the properties file from the class path. Importing a shared library that contains the log4j.properties file into an application makes the file accessible to configure log4j for the application.

You can even publish a set of shared libraries that contain different log4j.properties files to OC4J, such as for enabling different log levels, and then choose between the files when the application is deployed. After deployment, you can change the log4j.properties file that the application uses to switch between different logging settings.

**To configure the Oracle log4j appender for an application:**

1. Put the log4j.properties file into a JAR file, using a file name that indicates the configured log level, such as log4j.config.info.jar.

2. Create an OC4J shared library that contains the JAR file.

Figure 11–4 shows how to create a shared library in an OC4J instance with Application Server Control.

***Figure 11–4 OC4J Shared Library Containing log4j.properties File***

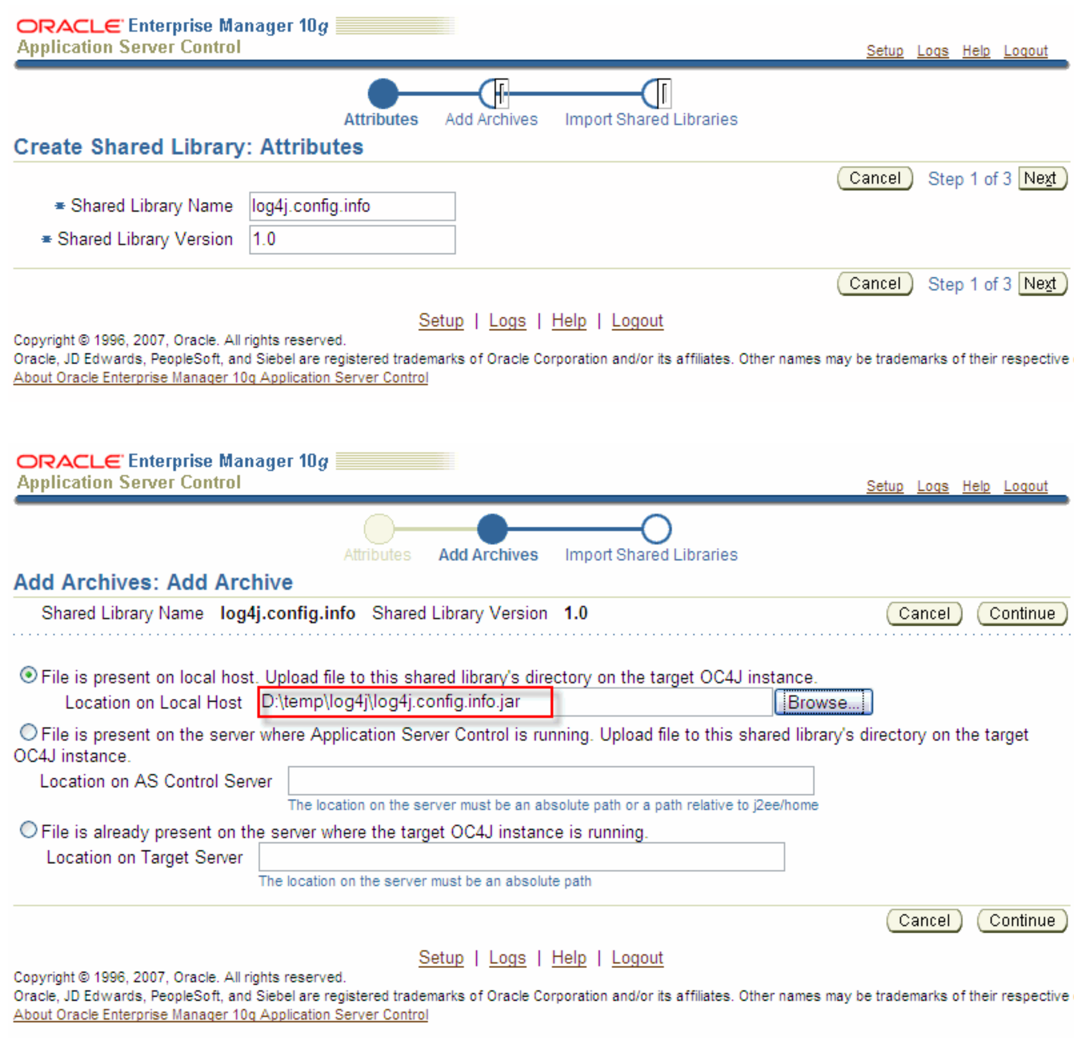
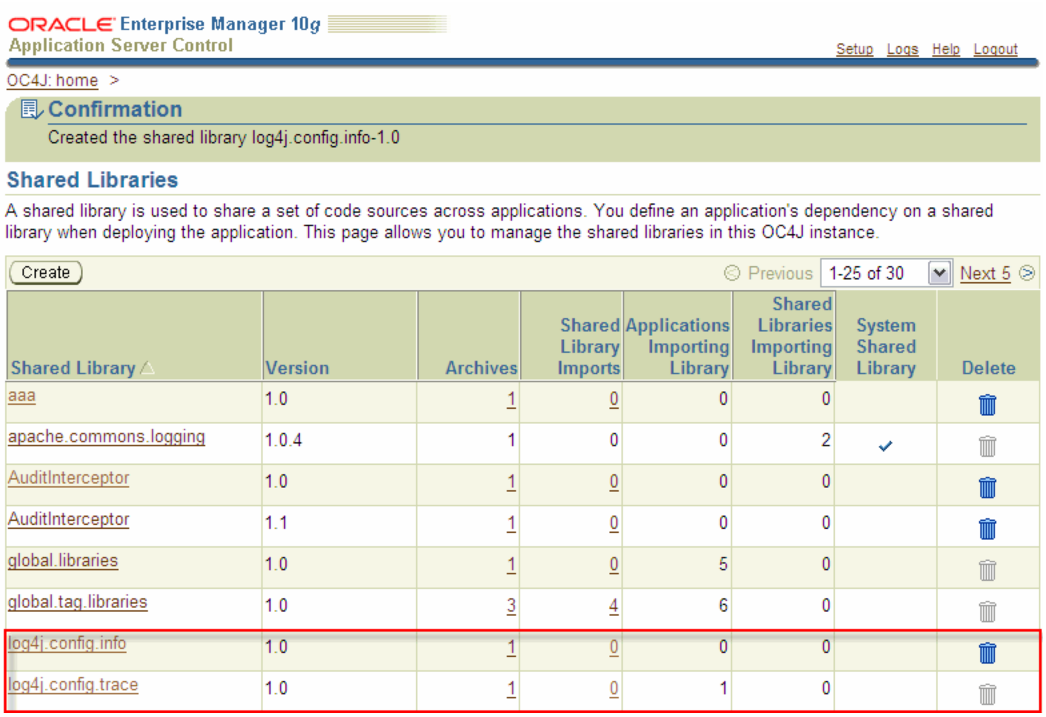



Figure 11–5 shows that two different shared libraries containing a `log4j.properties` file are available for importing into an application.

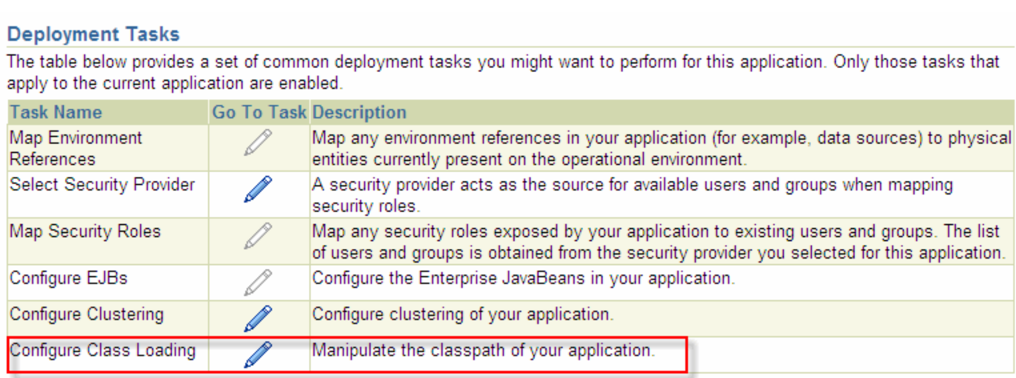
***Figure 11–5 Shared Libraries for log4j Configuration***




3. Import a shared library that contains the `log4j.properties` file, such as `log4j.config.info`, into the application during deployment.

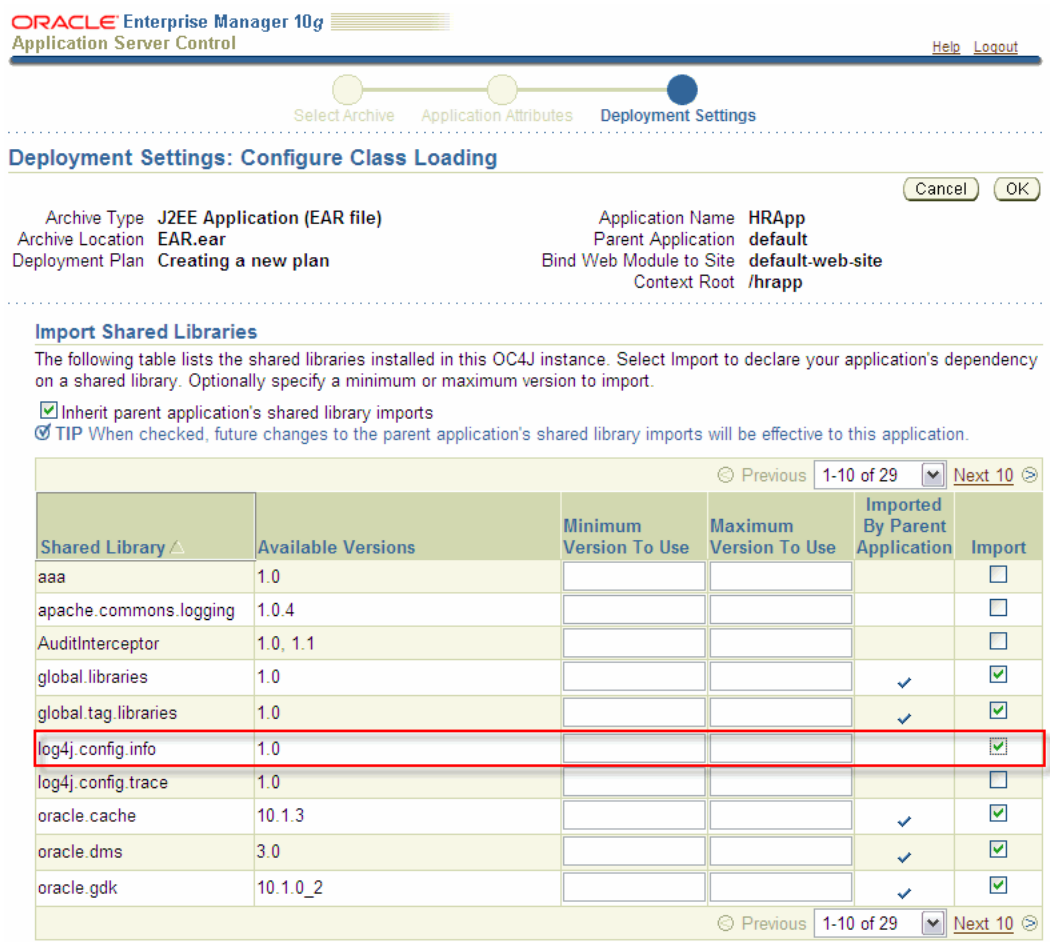
   a. Select **Configure Class Loading** on the Deployment Tasks page, as Figure 11–6 shows.

***Figure 11–6 Configure Class Loading Deployment Task***




   b. In the Import Shared Libraries area of the Configure Class Loading page, select a shared library, as Figure 11–7 shows.

***Figure 11–7 Shared Libraries for an Application to Import***




The `log4.info` shared library dictates how the log4j log entries for the application will be handled. In this case, the root logger is set to the INFO level, and the `OracleAppender` class is being employed to direct the log entries into the OC4J log system.

Ultimately, the customized shared-library settings for the application are written to its `orion-application.xml` configuration file, as Figure 11–8 shows.

***Figure 11–8 Shared-Library Settings in orion-application.xml***

```
<orion-application  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  schema-major-version="10" schema-minor-version="0" >
        <ejb-module remote="false" path="EJB.jar" />
        <web-module id="WEB" path="WEB.war" />
        <persistence path="persistence" />
        <imported-shared-libraries>
                <import-shared-library name="log4j.config.info"/>
        </imported-shared-libraries>
        <principals path="principals.xml" />
        <jazn provider="XML" />
        <log>
                <file path="application.log" />
        </log>
</orion-application>
```

If you want to change an application to use a lower log level, such as DEBUG or TRACE, then you can modify the `<import-shared-library>` element in `orion-application.xml` and restart the application to import the shared library that has the relevant log4j.properties file. Using the OC4J shared-library mechanism and a consistent naming convention should enable you to have as many reusable log4j configurations as you need for your applications.

After you configure the Oracle log4j appender, a single log.xml file will contain any log messages from log4j as well as log entries from OC4J.

## Making log4j Class Libraries Available to an Application

Before an application can use the Oracle log4j appender, you need to make the log4j classes available to the application. These classes are in the log4j and ojdl-log4j libraries. You can make the classes available to an application in one of these ways:

- Include the libraries in the application.

  For a Java EE 5 application, you can use the <library-directory> feature to specify a directory within an EAR file to hold shared libraries and then put the log4j and ojdl-log4j libraries in that directory. In the <library-directory> element of the application.xml file, you specify shared libraries for an application. Then OC4J scans the directories that you specified for archives to include at startup.

- Create a shared library in OC4J that contains the log4j and ojdl-log4j libraries and then import the shared library into the application when it is being deployed. After deployment, the libraries will be available to the application.

- If you have a Web application, you can put the log4j and ojdl-log4j libraries in the WEB-INF/lib directory for the application to use them.

  The ojdl-log4j library has a dependency on the log4j library, so they both have to be accessible at the same class-loader level.

# 12

# Using MBeans in OC4J

This chapter describes how to use the system MBeans provided with OC4J 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
- Subscribing to 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 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 with Application Server Control 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:

- Overview of MBeans
- Overview of the Top-Level OC4J System MBeans
- When Changes Made Through MBeans Take Effect
- How MBean Data Is Persisted

### Overview of 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, an instance of the `J2EESWebSite` MBean is created at OC4J startup to represent each Web site configured within the server.

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

- *Attributes*, name and value pairs of any type that the JMX client can get or set remotely. Attributes are analogous to properties set on an Enterprise JavaBeans (EJB) module. For example, the `state` attribute of the `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 to 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 earlier, the Application Server Control 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 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 12-5 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 12–1 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 states in an OC4J 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.<br>This MBean lazily creates instances of the `ClassLoader` MBean, each representing an instantiated class loader. |
| `EJBCompiler` | Configures the OC4J instance to generate client-side IIOP stubs during EJB deployment. Also used to specify the compiler to use for compiling EJB modules. |

*Table 12–1 (Cont.) Top-Level OC4J System MBeans*

| MBean | Description |
|---|---|
| `J2EEApplication` | Represents a J2EE application deployed into the OC4J instance.<br>Additional MBean instances are visible as child nodes representing the various components of the application:<br>■ `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.<br>■ `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 13, "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` | 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. Any changes made to MBean attributes require an OC4J server restart to take effect. |
| `JTAResource` | Represents a transaction manager instance. 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. 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. |
| `TimerService` | Represents an instance of the EJB timer. See the *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide* for details. |

## When Changes Made Through 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 a managed component - values set in 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, of the affected application, or even of the MBean before becoming available in the OC4J runtime. In these cases, the MBean and Application Server Control 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. 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 12–2 lists the values for this property.

***Table 12–2 Required Restart Property Values***

| Value | Impact |
|---|---|
| OC4J Restart | Indicates that the OC4J 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 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 MBean Data Is Persisted

Persistent data set via an MBean is written to the appropriate XML configuration file(s). For example, new values set in 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 12–3 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 attribute 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 Application Server Control user interface, which is relatively simple to use. To use this feature:

1. Launch Application Server Control.
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 be visible only 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.

## Subscribing to 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. This section describes how to subscribe to and view MBean-generated notifications.

Not all MBeans generate 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 Application Server Control.
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. 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 Application Server Control.
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 to OC4J can be accessed through the application's *home page* in the Application Server Control 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 Application Server Control.
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.

# 13

# Managing Web Sites in OC4J

This chapter explains how additional Web sites can be configured 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:



## Overview of a Web Site in OC4J

In the context of OC4J, Web requests sent to applications deployed to an OC4J instance are received by a *Web site*, a listener configured to accept requests on a specific protocol and port (or range of ports). Every Web module deployed into an OC4J instance must be bound to a Web site through which it will be accessed. This binding is typically performed as part of the application deployment process.

A `default` Web site is created in each OC4J instance upon installation. The configuration for the `default` Web site is defined in a configuration file, `default-web-site.xml`, installed by default in the `ORACLE_HOME/j2ee/instance/config` directory. See "Configuring Web Site Connection Data" on page 13-2 to gain an understanding of Web site configuration.

- Standalone OC4J

  In a standalone OC4J configuration, the `default` Web site is configured to receive HTTP requests directly on a specific port, which is `8888` by default. The site can alternatively be configured to receive secure HTTPS requests.

- Single OPMN-managed OC4J instance

  In a single OPMN-managed OC4J installation, the `default` Web site can be similarly configured to receive HTTP or HTTPS requests directly. A specific listener port can be specified in `default-web-site.xml`, or a range of ports can be set in the OPMN configuration file (`opmn.xml`). See "Configuring Web Site Data in OPMN-Managed OC4J Instances" on page 13-3 for details.

- Multiple OPMN-managed OC4J instances

In a cluster of two or more OPMN-managed OC4J instances, the `default` Web site is configured to receive requests forwarded from Oracle HTTP Server through Apache JServ Protocol (AJP).

The site can alternatively be configured to receive secure AJPS requests. A specific listener port can be specified, or a range of ports can be set in the OPMN configuration file. See "Configuring Web Site Data in OPMN-Managed OC4J Instances" on page 13-3 for details on OPMN configuration.

---

**Note:** In the current release, an OC4J instance supports only one AJP Web site at a time.

If you want to use both AJP and AJPS in Oracle Application Server, you need to configure them on different OC4J instances and use two Oracle HTTP Server (OHS) instances, one for routing AJP requests and the other for routing AJPS requests.

---

In addition to the `default` site, you can configure a new Web sites on each OC4J instance, as needed. (A Web site cannot listen on more than one protocol at a time.) You might want to create a new Web site for one of these reasons:

- Separating management and general Web access

  By default, the Application Server Control application is accessed via the `/em` context through the default Web site. However, you can create a new Web site specifically for the Application Server Control application to separate management access from general application access, if desired.

- Utilizing secure and nonsecure Web sites

  You can configure the default Web site to utilize SSL to create secure connections, or can create an additional site and bind it to Web applications that require a secure connection.

For more information about creating and configuring additional Web sites, see "Creating a New Web Site in OC4J" on page 13-6.

# Configuring Web Site Connection Data

The protocoal and listener ports used by a Web site are configured differently in standalone OC4J and Oracle Application Server environments, as these topics describe:

- Configuring Web Site Data in a Standalone OC4J Installation
- Configuring Web Site Data in OPMN-Managed OC4J Instances

## Configuring Web Site Data in a Standalone OC4J Installation

In a standalone OC4J installation, the protocol and listener ports used by a Web site must be explicitly defined in the corresponding `*-web-site.xml` configuration file. See "Creating the Web Site Configuration File" on page 13-7 for an overview of these files.

The `default` Web site is configured to listen for requests received via the HTTP protocol on port `8888` by default.

## Configuring Web Site Data in OPMN-Managed OC4J Instances

In an Oracle Application Server installation, in which Oracle Process Manager and Notification Server (OPMN) is used to manage OC4J instances, you can use OPMN for efficient management of Web site protocol and port configurations. When OPMN is started, it selects a port value starting at the bottom of the specified range and increments the value by 1 until a free port is found. Allowing OPMN to select from a range of ports in this manner avoids potential conflicts among OC4J processes.

You can change the OC4J port ranges with Application Server Control, with an opmnctl command, or in the `opmn.xml` file.

### Changing Port Ranges with Application Server Control

To change port ranges with Application Server Control:

1. From the Cluster Topology page, click **Runtime Ports**.
2. Click the **Configure Port** icon for the port you want to change.
3. In the Ports section of the Server Properties page, change the port range for the port you want to change.
4. Click **Apply**.
5. Navigate to the Cluster Topology page, select the OC4J instance that you modified, and click **Restart**.
6. Click **Yes** on the confirmation page.

### Changing Protocols and Port Ranges in opmn.xml

In this model, 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 the Web site will use can also be specified within this element.

> **Note:** The `opmnctl` command-line tool provides a command that you can use to update the `<port>` element for a specific Web site defined in the `opmn.xml` file for an OC4J instance.
>
> See "Configuring Web Sites with opmnctl" on page 13-5 for usage details.

The protocol and port values specified in `opmn.xml` override any corresponding values set in the corresponding Web site configuration file. 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.

The `<port>` element is defined in the `opmn.xml` configuration file, which is located in the `ORACLE_HOME/opmn/conf` directory. The syntax of the element follows:

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

Table 13–1 describes the attributes of the `<port>` element.

***Table 13–1 Attributes of the <port> Element***

| Attribute | Description |
|---|---|
| `id` | Required. Defines the name of the Web site, which is the name of the Web site configuration file minus the `.xml` extension. |
| protocol | Optional. Specifies the protocol through which the Web site will receive requests. Valid values follow:<br>■ `http`<br>■ `https`<br>■ `ajp`<br>■ `ajps`<br>If either `https` or `ajps` is specified, the value of the secure attribute of the root `<web-site>` element in the `*-web-site.xml` configuration file defined for the Web site will be overridden.<br>Changing the value of the `protocol` attribute does not reset the port range to the default range for the protocol you specify. To change the port range, you can use the `range` attribute.<br>If you want to use both `ajp` and `ajps` in Oracle Application Server, you need to configure them on different OC4J instances and use two Oracle HTTP Server (OHS) instances, one for routing AJP requests and the other for routing AJPS requests. |
| `range` | Optional. Specifies the start and end ports for the range of ports available for assignment by OPMN.<br>The default listener port ranges used are:<br>■ HTTP: `8888-8987`<br>■ AJP: `12501-12600`<br>You can specify a single port instead of a range by setting the start and end ports to the same port number.<br>Changing the value of the `protocol` attribute does not reset the port range to the default range for the protocol you specify. To change the port range, you can use the `range` attribute.<br>If you want to use both `ajp` and `ajps` in Oracle Application Server, you need to configure them on different OC4J instances and use two Oracle HTTP Server (OHS) instances, one for routing AJP requests and the other for routing AJPS requests. |

All `<port>` elements defining connection protocols are set in the `<process-type>` element defining the OC4J instance. The `<process-type>` element is a subelement of the `<ias-component>` element, in which the `id` attribute equals `default_group`.

For example, the `<port>` element in the following example configures the `default` Web site on the OC4J `home` instance to listen for AJP requests on ports `12501` through `12600`.

```
<ias-component id="default_group">
  <process-type id="home" module-id="OC4J" status="enabled">
   ...
    <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="1"/>
  </process-type>
</ias-component>
```

---

**Note:** The opmn.xml file must be reloaded for changes made to take effect. Run the following command on the affected node to reload opmn.xml:

```
opmnctl reload
```

This command will not affect OPMN-managed components, including Oracle HTTP Server, OC4J, and deployed applications.

---

### Configuring Web Sites with opmnctl

The OPMN command-line tool, opmnctl, provides a config port command that enables you to specify, update, or delete a Web site configuration defined in opmn.xml.

The opmnctl tool is installed in the *ORACLE_HOME*/opmn/bin directory on each node. The tool must be run individually on each node and will update only the opmn.xml file on that node.

#### Inserting or Updating Web Site Configuration Data in opnm.xml

The config port update command sets the specified data in a new or existing <port> element. The syntax of this command follows:

```
opmnctl config port update ias-component=componentName
 process-type=instanceName portid=webSiteName [range=startPort-endPort]
 [protocol=http|https|ajp|ajps]
```

For example, the default Web site for an OC4J instance is currently configured to listen for HTTP requests. The following command modifies the configuration for the default Web site so that it will receive and respond to Apache JServ Protocol (AJP) requests from Oracle HTTP Server.

```
opmnctl config port update ias-component=default_group process-type=home
   portid=default-web-site protocol=ajp

opmnctl reload
```

The opmnctl reload command is invoked to reload the updated opmn.xml file into the OC4J runtime.

Changing the protocol for a Web site does not reset the port range to the default range for the specified protocol. To change the port range, you can use the range parameter in the config port update command.

If you want to use both AJP and AJPS in Oracle Application Server, you need to configure them on different OC4J instances and use two Oracle HTTP Server (OHS) instances, one for routing AJP requests and the other for routing AJPS requests.

#### Deleting Web Site Configuration Data from opnm.xml

The delete command removes the <port> element defined for the specified Web site. The syntax is as follows:

```
opmnctl config port delete ias-component=componentName
    process-type=instanceName portid=webSiteName
```

For example, the following removes the <port> element defined for the default Web site from opmn.xml:

```
opmnctl config port update ias-component=default_group process-type=home
```

```
    portid=default-web-site

opmnctl reload
```

Table 13–2 describes the options that can be set on the `opmnctl config port` command line.

***Table 13–2 opmnctl config port Options***

| Option | Description |
|---|---|
| `ias-component` | Set to `default_group` to update the OC4J configuration in `opmn.xml`. |
| `process-type` | Set to the identifier of the OC4J instance to update; for example, `home`. This value matches the value of the `id` attribute in the `<process-type>` subelement of `<ias-component>` in `opmn.xml`. |
| `portid` | Set to 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. Valid only for the `update` operation. Valid values are:<br>▪ `http`<br>▪ `https`<br>▪ `ajp`<br>▪ `ajps`<br>If either `https` or `ajps` is specified, the value of the secure attribute of the root `<web-site>` element in the `*-web-site.xml` configuration file defined for the Web site will be overridden.<br>Changing the protocol for a Web site does not reset the port range to the default range for the specified protocol. To change the port range, you can use the `range` parameter. |
| `range` | Sets the start and end ports for the range of ports available for assignment by OPMN. Valid only for the `update` operation.<br>The default port ranges follow:<br>▪ HTTP: `8888-8987`<br>▪ AJP: `12501-12600`<br>You can specify a single port instead of a range by setting the start and end ports to the same port number.<br>Changing the protocol for a Web site does not reset the port range to the default range for the specified protocol. To change the port range, you can use the `range` parameter. |

# Creating a New Web Site in OC4J

Bringing a new Web site to life in an OC4J instance is essentially a two- or optionally three-step process:

1. Create the XML configuration file for the Web site within the OC4J installed directory structure.
2. Add a reference to the new Web site configuration file in `server.xml`, the OC4J configuration file.
3. For OPMN-managed OC4J instances, add a `<port>` element defining the Web site's protocol and port range to `opmn.xml`.

After these steps are completed, the Web site will be available for binding with applications. The following topics provide details on Web site configuration.

## Creating the Web Site Configuration File

The key information defined in a Web site configuration file includes the following:

- The Web context for each application bound to the site, which is appended to the URL used to access the site (for example, `/em`).
- The protocol the site uses. In an OPMN-managed environment, this value will be overridden by the protocol specified in `opmn.xml`.
- The port the site listens on. In an OPMN-managed environment, this value will be overridden by the port range specified in `opmn.xml`.
- The location of the access log file, which tracks user access to the site.

The most straightforward way to create a new configuration file is to make a copy of the default Web site configuration file, `default-web-site.xml`, which is located in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/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:

- `host`: The host for this Web site, as either a DNS host name or an IP address. If a server has multiple IP addresses, you can use the ALL setting to listen to all the IP addresses.
- `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 subelements:

- A `<default-web-app>` element defining the Web application accessed by default through the Web site. When a single application is bound to the Web site, such as Application Server Control, specify the application within this element.
- One or more `<web-app>` subelements 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, or context root, on this Web site to which the Web module is to be bound

- An <access-log> element specifying the log file to which requests sent to the site are logged

As an example, assume that you will create a configuration file named ascontrol-web-site.xml, which defines a Web site that will be used exclusively to provide management access to Application Server Control. The root <web-site> element within this file will contain all of the required configuration data, as follows:

```
<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="1810"
  display-name="Application Server Control Web Site">
    <default-web-app application="ascontrol" name="ascontrol" access-log="true" />
    <access-log path="../log/ascontrol-web-access.log" />
</web-site>
```

For details on the structure of this element, see the <web-site> element description on page B-21.

> **Note:** If you are creating a Web site exclusively for use by Application Server Control, as illustrated in this example, you must also update the **Launch Application Server Control** link on the OC4J home page, accessed through *ORACLE_HOME*/j2ee/, with the correct URL.

## Referencing the Web Site Configuration File in server.xml

The location of every Web site configuration file must be referenced in a <web-site> element in server.xml, the OC4J configuration file, located in the *J2EE_HOME*/config directory. 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="./default-web-site.xml" />
  <web-site path="./ascontrol-web-site.xml" />
</application-server>
```

In this example, the locations of all of the Web site configuration files are relative to the location of server.xml.

> **Note:** If OC4J polling is disabled, OC4J must be restarted for changes to server.xml to take effect.

## Defining the Web Site Connection Data in opmn.xml

In an Oracle Application Server installation, in which Oracle Process Manager and Notification Server (OPMN) is used to manage OC4J instances, you can use OPMN can be used to manage Web site protocol and port configuration efficiently. Use the opmnctl config port command to add a new <port> element for the Web site to the OC4J instance definition in opmn.xml.

The following example sets the protocol (HTTP) and port (1810) for the ascontrol Web site:

```
opmnctl config port update ias-component="default_group" id="ascontrol-web-site"
```

```
protocol="http" range="1810"
```

The example command adds the new <port> element to the OC4J home instance definition in the opmn.xml file on the host machine. This OC4J instance is now configured with two Web sites: the default site and the new ascontrol site.

```
<ias-component id="default_group">
  <process-type id="home" module-id="OC4J" status="enabled">
   ...
    <port id="default-web-site" protocol="ajp" range="12501-12600"/>
    <port id="ascontrol-web-site" protocol="http" range="1810"/>
    <port id="rmi" range="12401-12500">
    <port id="jms" range="12601-12700">
    <process-set id="default" numprocs="1"/>
  </process-type>
</ias-component>
```

## 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 50 KB 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 to which the application is bound. 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 default-web-site.xml follows:

```
<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="true"/>
    <access-log path="../log/http-web-access.log" />
</web-site>
```

A similar entry in secure-web-site.xml follows:

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

### Specifying the Cookie Domain

You can 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" />
```

## Configuring a Secure Web Site in OC4J

OC4J supports Secure Socket Layer (SSL) communication between the client and OC4J using HTTPS and AJPS. You can modify the configuration file for the `default` Web site to utilize SSL to create secure connections, or can create an additional site and bind it to Web applications requiring a secure connection.

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

### Creating the Secure Web Site Configuration File

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

```
<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 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 preceding example, 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 attributes and elements:

– The optional `factory` attribute is used to specify the third-party `SSLServerSocketFactory` implementation to use if the application is not using JSSE.

  If the application uses a third-party `SSLServerSocketFactory` implementation, you can use `<property>` subelements of `<ssl-config>` to send parameters to the factory.

  ---

  **Note:** The `factory` attribute and its parameters are deprecated.

  ---

– 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. (Although the example shows the default implementation, it is implicit and does not need to be specified.)

– One or more <property> elements containing parameters to pass to the `SSLServerSocketFactory`. Each element contains a `name` attribute and a `value` attribute, enabling you to specify parameters as name and value pairs.

  ---

  **Note:** Parameters for the `SSLServerSocketFactory` are deprecated.

  ---

When the Web site configuration file is ready, add a `<web-site>` element referencing `server.xml`, the OC4J configuration file located in the *`J2EE_HOME`*`/config` directory. 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="./default-web-site.xml" />
  <web-site path="./mycustom-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 *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 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 either that you 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 want 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.

## Starting and Stopping Web Sites

A Web site is available by default once it has been configured on an OC4J instance. However, Application Server Control 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:** Because Application Server Control uses `ascontrol-web-site`, you cannot stop it through the user interface.

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

## Configuring Web Site Access Logging

OC4J provides the ability to generate an *access log* for each Web site, which records requests submitted by clients 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 with Application Server Control. 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>` element. If neither element is 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
- Viewing Text Access Log Files
- Configuring ODL Access Logging
- Viewing ODL Access Log Files
- Enabling or Disabling Access Logging for a Web Module or Application

### Configuring Text-Based Access Logging

Text-based access logging is configured through the `<access-log>` subelement of the root `<web-site>` element in the corresponding Web site's configuration file (`*-web-site.xml`).

> **Note:** 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.

This `<access-log>` element has the following attributes:

- path: Specifies the path and file name of the access log. This is the only required attribute; specifying it alone will cause access logs to be generated.

  The path must be relative to the *ORACLE_HOME*/j2ee/*instance*/config directory to enable the log to be viewed with Application Server Control, as illustrated by the following entry in default-web-site.xml:

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

- format: Specifies one or more of several supported variables that result in information being prepended to log entries. Supported variables are $time, $timeUsed, $request, $ip, $host, $path, $size, $method, $protocol, $user, $status, $referer, $agent, $cookie:[*name*], $header:[*name*], 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"
  ```

  This default configuration 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 quotation marks (as specified), and the status and size in the first message are 200 and 2929, respectively.

  The $timeUsed variable measures in seconds the time used for processing an HTTP transaction. The name of this variable is case sensitive.

- split: Specifies how often to begin a new access log. Supported values are none (equivalent to never, which is the default value), hour, day, week, and month. If split is specified, the suffix attribute (documented in the following text) 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 value 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/

The following entry in default-web-site.xml will generate a file named default-web-access.log file:

```
<web-site>
```

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

</web-site>
```

The files will be generated in the following locations, depending on your OC4J installation:

- Standalone OC4J:

  Log files will be generated in *ORACLE_HOME*/j2ee/home/log/.

- Oracle Application Server:

  Files will be generated in an OC4J instance-specific directory named *ORACLE_HOME*/j2ee/*instance*/application-deployments/log/*instance*_default_group_1.

## Viewing Text Access Log Files

Access log text files can be viewed by clicking the **Logs** link in Application Server Control. ODL log files are identified in the Log Files page by the .log extension.

1. Click the **Logs** link at the bottom of any Application Server Control page.
2. Expand **OC4J**.
3. Expand **<instanceName>**. The default instance name is home.

## 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 must be relative to the *-web-site.xml configuration file to enable the log files to be viewed with Application Server Control.

  For easier management, include the name of the Web site in the path.

- 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 size specified in the attributes.

For example, the following entry in default-web-site.xml will cause log.xml files to be generated. It will also set log files to a maximum of 1,000 KB and the directory maximum to 10,000 KB in a /default-web-access directory within *ORACLE_HOME*/j2ee/home/log/.

```
<web-site>
...
```

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

The log files will be generated in the following locations, depending on your OC4J installation.

- Standalone OC4J:

  Log files will be generated in *ORACLE_HOME*/j2ee/home/log/default-web-access/.

- Oracle Application Server:

  Files will be generated in an OC4J instance-specific directory named *ORACLE_HOME*/j2ee/*instance*/application-deployments/log/*instance*_default_group_1/default-web-access.

For more information about ODL access logging, see "Managing ODL Log Files" on page 11-6.

## Viewing ODL Access Log Files

ODL-formatted log files can be viewed by clicking the **Logs** link in Application Server Control, enabling 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.

1. Click the **Logs** link at the bottom of any Application Server Control page.
2. Expand **OC4J**.
3. Expand **<instanceName>**. In both standalone OC4J and OAS, the default instance name is home.
4. Expand the **Default Web Site** node.
5. Expand **Diagnostic Message Logs**.

## Enabling or Disabling Access Logging for a Web Module or Application

If either the <access-log> or <odl-access-log> element is defined in a Web site configuration file, access logging is *not* enabled by default for the Web modules within applications bound to the Web site. As of OC4J 10*g* 10.1.3.1.0, the default value of the access-log attribute of any application-specific <web-app> elements in the configuration file is false.

However, it is possible to enable access logging for a specific module by setting the access-log attribute to true for the module.

It may be desirable to leave access logging disabled in situations where a Web module submits such a massive number of requests that text-based access log files will quickly become bloated. If access-log is set to true in the Web site configuration file or for the module, you can disable access logging for the module by setting its access-log attribute to false.

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

```
<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" access-log="true" />
</web-site>
```

# 14

# 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:

- Validating XSDs to Be Registered
- Registering a DTD or XSD

## Validating XSDs to Be Registered

OC4J provides the ability to validate XML deployment descriptors defined by an XSD 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, "OC4J Runtime Configuration" for details on command-line options.

Validation requires that the 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 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

- The *public identifier* is a string
- The *system identifier* is a URL

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.

By default, `entity-resolver-config.xml` already contains registration entries for the standard J2EE 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/`*`instance`*`/config` directory on the OC4J host machine.

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

- `<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` attribute 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 restarted after you make 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
- Additional Help

## Problems and Solutions

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

- java.lang.OutOfMemory Errors
- Application Performance Impacted by Garbage Collection Pauses
- Invalid or Unneeded Library Elements Degrading Performance
- ClassCastExceptions and ClassNotFound Errors
- OC4J Fails to Start: Unable to Find Java Compiler
- Error When Clustering an Application
- Error When Downgrading from JDK 5.0 to JDK 1.4.2
- OC4J Hanging When Starting Applications in Oracle Application Server

### Warning Regarding Maximum Concurrent Timers

#### Problem

A warning such as the following example can occur when the number of concurrent timers exceeds the maximum:

```
WARNING J2EE OJR-10002

The number of concurrent Timers has reached the maximum limit
```

By default, OC4J 10*g* (10.1.3.5.0) allows only eight concurrent timers. (A timer can be triggered through an EJB timer, the timer service, or the scheduler.) This limit is low by default because each timer is expected to be of short duration. When the number of timers is at the limit, such as if timers are running longer for any reason, timers are no longer executed. When a new timer occurs, OC4J logs a warning message.

#### Solution

To work around this problem, you can use either of two OC4J system properties:

- `timer.service.debug`

  This property determines whether to log additional diagnostic information for the timer service, including information about the current number of running timers. For example:

  ```
  -Dtimer.service.debug=true
  ```

- `executor.concurrent.tasks`

  This property specifies the number of concurrent tasks for the Executor Service. Through this property you can increase the maximum number of concurrent timers allowed by OC4J. For example:

  ```
  -Dexecutor.concurrent.tasks=12
  ```

> **Note:** Each timer executes in a separate thread. If the maximum number of timers is set too high, resulting in numerous timers executing, then OC4J uses many threads. Oracle recommends that you recycle threads once they finish executing.

For information about setting system properties, see Chapter 4, "OC4J Runtime Configuration."

## java.lang.OutOfMemory Errors

### Problem

Out-of-memory errors indicate that the heap size of the Java instance is lower than the memory required by applications running within OC4J.

### Solution

Increase the heap size for the OC4J process to the desired amount of memory at OC4J startup:

```
java -Xms512m -Xmx512m -jar oc4j.jar
```

If your application is running in an OPMN-managed environment, these JVM settings are defined within a `<data id="java-options">` element in the `opmn.xml` configuration file. For example:

```
<ias-component id="default_group">
  <process-type id="home" module-id="OC4J" status="enabled">
    <module-data>
      <category id="start-parameters">
        <data id="java-options" value="-Xms512m -Xmx512m -Djava.awt.headless=true
          -Dhttp.webdir.enable=false"/>
        ...
      </category>
      ...
    </module-data>
  </process-type>
</ias-component>
```

If your application is running in a Linux or UNIX environment, 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.

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 1 GB 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 Degrading 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.

## ClassCastExceptions and ClassNotFound Errors

### Problem

Most class-loading errors are related to class visibility—either too much or not enough. Collisions between classes packaged in multiple JARs or inherited by default from parent applications can be a problem.

### Solution

Chapter 3, "Utilizing the OC4J Class-Loading Framework" in the *Oracle Containers for J2EE Developer's Guide* contains detailed documentation on avoiding and troubleshooting issues related to class loading. It also explains how you can use shared libraries to avoid many of these issues within OC4J.

## OC4J Fails to Start: Unable to Find Java Compiler

### Problem

An error similar to the following one is seen at OC4J startup:

```
05/10/28 13:58:49 Error initializing server: Error initializing ejb-modules:
Error generating wrappers for file:/C:/oc4j/j2ee/home/applications/admin_ejb.jar:
javac.exe not found under <directory>, please use a valid jdk or specify the
location of your java compiler in server.xml using the <java-compiler .../> tag
```

### Solution

The error indicates that OC4J is unable to locate the required JDK. To resolve this issue, start OC4J from the `javac.exe` location on the command line. This will set the location of the JDK.

For example:

```
C:\ORACLE_HOME\j2ee\home\C:\jdk\bin\java -jar oc4j.jar
```

## Error When Clustering an Application

### Problem

The following error is thrown when clustering is configured for an application:

```
WARNING: The service implementation <classname> does not implement
java.io.Serializable. *This class is not suitable for clustered environments*
indicated by recoverable=true.
```

### Solution

This error indicates that the class is not serializable, and therefore cannot utilize the OC4J replication framework.

## Error When Downgrading from JDK 5.0 to JDK 1.4.2

### Problem

The following error occurs when configuring an OPMN-managed OC4J instance installed as a component of Oracle Application Server, which uses the JDK 5.0 by default, to use the JDK 1.4.2.

```
oracle.oc4j.loader.util.AnnotatedLinkageError:
MBeanServerEjbHome_StatefulSessionHomeWrapper1 (Unsupported major.minor
```

```
version 49.0)
```

#### Solution

An OPMN-managed OC4J instance installed as a component of Oracle Application Server will use the JDK 5.0 by default. This newer version of the JDK is required to utilize EJB 3.0 and offers numerous performance improvements. However, if applications that will be deployed to OC4J require a JDK 1.4.2 release, it may be necessary to *downgrade* to the earlier version.

Before switching from JDK 5.0 to JDK 1.4.2, you must remove all compiled application files from the OC4J instance:

1. Stop the OC4J instance.
2. Delete the *ORACLE_HOME*/j2ee/*instance*/application-deployments directory.

   Deleting this directory will cause the application files to be recompiled when OC4J is restarted with the JDK 1.4.2.

You can specify the JDK to use for each OC4J instance through manual edits to the opmn.xml configuration file. If you want to use the javac compiler installed with the JDK defined in the JAVA_HOME environment variable, also remove the <java-compiler> element from server.xml and let OC4J rediscover the default settings.

### Unsupported Methods in JMX MBeanServer and MBeanServerConnection Interfaces

#### Problem

A number of methods from the JMX MBeanServer interface are not available to a J2EE application when it uses an MBeanServer object obtained from the following operation:

```
MBeanServer mbsrv = MBeanServerFactory.newMBeanServer();
```

The use of any of the following methods on the returned MBeanServer object will throw an UnsupportedOperationException exception:

```
public final ClassLoader getClassLoaderFor(ObjectName mbeanName)

public final ClassLoader getClassLoader(ObjectName loaderName)

public final ClassLoaderRepository getClassLoaderRepository()

public final Object instantiate(String className)

public final Object instantiate(String className, ObjectName loaderName)

public final Object instantiate(String className, Object[] params, String[]
signature)

public final Object instantiate(String className, ObjectName loaderName, Object[]
params, String[] signature)

public final ObjectInstance createMBean(String className, ObjectName name)

public final ObjectInstance createMBean(String className, ObjectName name,
ObjectName loaderName)
```

```
public final ObjectInstance createMBean(String className, ObjectName name,
Object[] params, String[] signature)

public final ObjectInstance createMBean(String className, ObjectName name,
ObjectName loader, Object[] params, String[] signature)

public final ObjectInputStream deserialize(ObjectName name, byte[] data)

public final ObjectInputStream deserialize(String className, byte[] data)

public final ObjectInputStream deserialize(String className, ObjectName
loaderName, byte[] data)
```

A number of methods from the MBeanServerConnection interface are not supported when an application uses the Oracle JMX connectors. The use of any of the following methods on the `MBeanServerConnection` object that is created will throw an `UnsupportedOperationException` exception:

```
public final ObjectInstance createMBean(String className, ObjectName name)

public final ObjectInstance createMBean(String className, ObjectName name,
ObjectName loaderName)

public final ObjectInstance createMBean(String className, ObjectName name,
Object[] params, String[] signature)

public final ObjectInstance createMBean(String className, ObjectName name,
ObjectName loader, Object[] params, String[] signature)
```

### Solution

If your application uses the JMX MBeanServer or MBeanServerConnection interface, avoid using any of the unsupported methods in the application.

## OC4J Hanging When Starting Applications in Oracle Application Server

### Problem

In an OPMN-managed environment, OPMN appears to *hang* while trying to start OC4J, resulting in an error similar to the following one:

```
ias-component/process-type/process-set:
    default_group/home/default_group/

   Error
   Process (index=1,uid=2012873812,pid=2988)
   time out while waiting for a managed process to start
```

### Solution

An application that requires significant resources, such as an application that attempts to acquire multiple database connections for its various components, can cause OC4J to fail to start. You can manage this by specifying the maximum amount of time to allow applications to start in the `<start-timeout>` element defined for the OC4J instance in `opmn.xml`. After this value is reached, the application will not be started. This value will be applied to all applications deployed to the instance.

The following example increases the timeout value to `800` seconds for applications deployed to the `home` OC4J instance:

```
<ias-component id="default_group">
  ...
  <process-type id="home" module-id="OC4J" status="enabled">
    ...
    <start timeout="800" retry="2"/>
  </process-type>
</ias-component>
```

## Additional 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 Configuration Files Used by OC4J

The configuration data for an OC4J 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.

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***

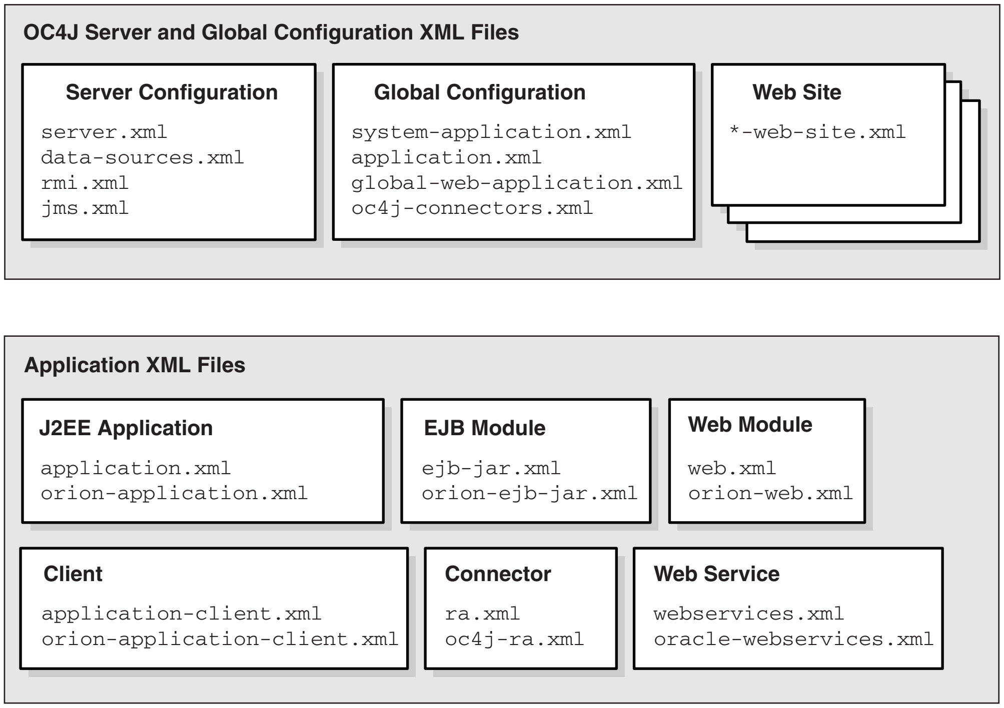



Table B–1 describes the role and function for each OC4J server-level XML file as well as the global configuration files displayed in the preceding figure.

Unless otherwise indicated, all of these files are installed in the `ORACLE_HOME/j2ee/instance/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.<br><br>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. |
| `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. The file provides configuration data used at OC4J startup, such as data needed to load required shared libraries. |

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

| XML Configuration File | Features/Components |
|---|---|
| `application.xml` | Contains the configuration for the `default` application. All user-deployed applications and standalone modules that do not have a designated parent are deployed to this application by default.<br>This file includes common settings that serve as default configuration values applied to deployed applications.<br>This file is completely unrelated to `application.xml`, the J2EE standard deployment descriptor. |
| `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 all standalone resource adapters 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/*instance*/config directory, but can be installed in a different location.<br>The configuration for the `default` Web site created within each OC4J instance is defined in `default-web-site.xml`. |

Table B–2 describes the roles and functions of the various application-level XML files displayed in the preceding figure.

Unless otherwise indicated, all of these files are installed in the *ORACLE_HOME*/j2ee/*instance*/config directory by default.

*Table B–2 Application-Level Configuration Files*

| XML Configuration File | Features/Components |
|---|---|
| `application.xml` | The 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. |

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

| XML Configuration File | Features/Components |
|---|---|
| `orion-application.xml` | The OC4J-specific deployment descriptor, which contains configuration data for a specific deployed application.<br><br>In this file, you can use the `<jazn-web-app>` element to configure the OracleAS JAAS Provider and Oracle Single Sign-On properties for servlet execution. You must set these features appropriately to invoke a servlet under the privileges of a particular security subject.<br><br>When Oracle Identity Management is being used as the security provider for a Web application, with Oracle Single Sign-On for authentication, you can synchronize a servlet session with the Oracle Java Authentication and Authorization Service (JAAS) Provider user context through `<jazn-web-app>`. To synchronize the session with the user context, set the `sso.session.synchronize` property to `true`, the default. You can do this in a `<property>` subelement under `<jazn-web-app>`:<br><br>`<jazn-web-app ...>`<br>`  <property name="sso.session.synchronize" value="true"/>`<br>`</jazn-web-app>`<br><br>Or you can set the property to `false`.<br><br>To take effect, changes to orion-application.xml require an application restart (if the changes were made through Application Server Control or the security provider MBean) or an OC4J restart (if the changes were made manually).<br><br>For additional information about JAAS and the features described for this element, see the *Oracle Containers for J2EE Security Guide*. You can also refer to related Sun Microsystems documentation at the following location:<br><br>`http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASRefGuide.html` |
| `web.xml` | The J2EE Web application deployment descriptor, used to define the Web application deployment parameters that are included in the WAR file.<br><br>In addition, you can specify the URL pattern for servlets and JSPs in this file. For example, a 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 and auto-reload of JSPs are enabled. |
| `ejb-jar.xml` | The J2EE EJB module deployment descriptor, included in the Enterprise JavaBeans (EJB) JAR file. Defines the specific structural characteristics and dependencies of the EJB modules within a JAR and provides instructions for the EJB container about how the beans expect to interact with the container. |

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

| XML Configuration File | Features/Components |
|---|---|
| `orion-ejb-jar.xml` | The OC4J-specific EJB deployment descriptor. Defines OC4J-specific configuration data for all EJB modules within an archive, including EJB pool settings, timeout 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 module'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 end-point addresses and data specific to EJB modules implemented as Web services. The file can be packaged in either a WAR or an EJB JAR containing a Web service. |

# Elements of the OC4J Server Configuration File (server.xml)

The OC4J configuration file, `server.xml`, is located in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/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 because notations are added and updated as needed by OC4J.

The `server.xml` file includes references to the application descriptor of each application within the OC4J instance, either directly or indirectly. In the case of a typical J2EE application, this reference points to the 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.

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

An example of the server.xml configuration file for OC4J follows, 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="./default-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 <application-server> Attributes***

| Name | Description |
|---|---|
| `application-directory` | Values: string<br>Default: `../applications`<br><br>The target directory for deployed archives. |
| `application-auto-deploy-directory` | Values: string<br>Default: n/a<br><br>The directory into which EAR files can be copied, triggering automatic deployment/redeployment of the application. |
| `connector-directory` | Values: string<br>Default: `../connectors`<br><br>The target directory for standalone resource adapters. |
| `deployment-directory` | Values: string<br>Default: `../application-deployments`<br><br>The directory containing the OC4J-specific deployment descriptors and generated files, such as compiled JSP classes and EJB wrapper classes. |
| `check-for-updates` | Values: `all|adminClientOnly|none`<br>Default: `adminClientOnly`<br><br>Enables OC4J polling, which automatically checks for changes made to currently deployed applications and modules and redeploys any components that have been modified.<br><br>See the *Oracle Containers for J2EE Deployment Guide* for an explanation of supported values and the impact of each. |
| `localhostIsAdmin` | Values: Boolean<br>Default: `true`<br><br>If `true`, allows easier access if the process initiating the administrative operation is a process local to the OC4J host machine. |
| taskmanager-granularity | Values: int<br>Default: `1000`<br><br>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 to 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 <application> Attributes***

| Name | Description |
|---|---|
| `name` | Values: string<br>Default: n/a<br><br>The application name; typically the same as the EAR file name without the `.ear` extension. |
| path | Values: string<br>Default: n/a<br><br>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<br>Default: `true`<br><br>If `true`, the application is started with OC4J and is available to serve requests or for configuration through JMX MBeans. 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>

**Required?** Required; multiple allowed

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

***Table B–5 <code-source> Attributes***

| Name | Description |
|---|---|
| path | Values: string<br>Default: n/a<br><br>The path to a JAR or ZIP file included in a shared library.<br><br>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.<br><br>You can optionally set `path="*"` to force OC4J to consume all of the archives within the shared library subdirectory. |

## <custom-thread-pool>

**Parent element:** <application-server>

**Required?** Optional; multiple allowed

**Child elements:**

Contains the configuration for a single thread pool with the specified name within an OC4J process. One or more applications can be configured to use the thread pool. See "Configuring OC4J Thread Pools" on page 10-1 for details.

*Table B–6 <custom-thread-pool> Attributes*

| Name | Description |
|---|---|
| `name` | Values: string<br>Default: required<br>The thread pool name. |
| `min` | Values: string<br>Default: `0`<br>The minimum number of threads that OC4J can simultaneously execute. |
| `max` | Values: string<br>Default: `1024`<br>The maximum number of threads that OC4J can simultaneously execute. |
| queue | Values: string<br>Default: `0`<br>The maximum number of requests that can be kept in the queue. |
| `keepAlive` | Values: string<br>Default: `600000`<br>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.<br>To never destroy threads, set to `-1`.<br>The default value, `600000` milliseconds (10 minutes), is also the minimum value allowed if not `-1`. |
| `stackSize` | Values: string<br>Default: `0`<br>The size of the thread pool stack. |
| debug | Values: Boolean<br>Default: `false`<br>If `true`, prints thread pool information to the console at startup. If `false`, the thread pool information is not printed. |

## <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 the order in which the classes are executed.

## <global-application>

**Parent element:** <application-server>

**Required?** Required; one only

**Child elements:**

Specifies the OC4J global application, known as the `default` application. The `name` attribute defines its name; the `path` attribute specifies what to use as the OC4J global application descriptor.

***Table B–7 <global-application> Attributes***

| Name | Description |
|---|---|
| `name` | Values: string<br>Default: `default`<br><br>The global application name. |
| `path` | Values: string<br>Default: `application.xml`<br><br>The file name and path for the global application descriptor file. The default descriptor is *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config/application.xml`. |

## <global-thread-pool>

**Parent element:** <application-server>

**Required?** Optional; one only

**Child elements:**

Contains the old configuration format for thread pools within an OC4J process. If the `server.xml` file contains the `<global-thread-pool>` element, the `min`, `max`, `keep-alive`, and `queue` attribute values apply to the `http` thread pool, which is created at OC4J startup. The `cx-*` attributes apply to the `rmi connection` thread pool, and the `rmiRequest-*` attributes apply to the `rmi request` thread pool. See "Configuring OC4J Thread Pools" on page 10-1 for details.

The `<global-thread-pool>` element is deprecated. If the `server.xml` file contains this element, OC4J changes it to equivalent <thread-pool> elements that define thread pools in the new configuration format.

***Table B–8 <global-thread-pool> Attributes***

| Name | Description |
|---|---|
| `min` | Values: string<br>Default: n/a<br><br>The minimum number of threads that OC4J can simultaneously execute. |

*Table B–8 (Cont.) <global-thread-pool> Attributes*

| Name | Description |
|---|---|
| max | Values: string<br>Default: n/a<br>The maximum number of threads that OC4J can simultaneously execute. |
| queue | Values: string<br>Default: n/a<br>The maximum number of requests that can be kept in the queue. |
| debug | Values: Boolean<br>Default: false<br>If true, prints thread pool information to the console at startup. If debug is false, the thread pool information is not printed. |
| keep-alive | Values: string<br>Default: 600000<br>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.<br>A value of -1 specifies never to destroy the thread.<br>The default value, 600000 milliseconds (10 minutes), is also the minimum value allowed if not -1. |
| cx-max | Values: string<br>Default: n/a<br>The minimum number of connection threads that OC4J can simultaneously execute. |
| cx-min | Values: string<br>Default: n/a<br>The maximum number of connection threads that OC4J can simultaneously execute. |
| cx-queue | Values: string<br>Default: n/a<br>The maximum number of requests that can be kept in the queue. |
| cx-debug | Values: Boolean<br>Default: false<br>If true, prints thread pool information to the console at startup. If cx-debug is false, the thread pool information is not printed. |
| cx-keep-alive | Values: string<br>Default: 600000<br>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.<br>A value of -1 specifies never to destroy the thread.<br>The default value, 600000 milliseconds (10 minutes), is also the minimum value allowed if not -1. |
| rmiRequest-max | Values: string<br>Default: n/a<br>The minimum number of connection threads that OC4J can simultaneously execute. |

*Table B–8 (Cont.) <global-thread-pool> Attributes*

| Name | Description |
|---|---|
| rmiRequest-min | Values: string<br>Default: n/a<br><br>The maximum number of connection threads that OC4J can simultaneously execute. |
| rmiRequest-queue | Values: string<br>Default: n/a<br><br>The maximum number of requests that can be kept in the queue. |
| `rmiRequest-debug` | Values: Boolean<br>Default: false<br><br>If `true`, prints thread pool information to the console at startup. If `rmiRequest-debug` is `false`, the thread pool information is not printed. |
| `rmiRequest-keep-alive` | Values: string<br>Default: `600000`<br><br>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.<br><br>A value of `-1` specifies never to destroy the thread.<br><br>The default value, `600000` milliseconds (10 minutes), is also the minimum value allowed if not `-1`. |

## <global-web-app-config>

**Parent element:** <application-server>

**Required?** Required; one only

**Child elements:**

Identifies the configuration file for the OC4J global web application, which by default is the parent of all other Web applications.

The name and root directory path, or context root, 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 `default-web-site.xml` file. In standalone OC4J, the default context root for the default Web application is "`/`".

*Table B–9 <global-web-app-config> Attributes*

| Name | Description |
|---|---|
| path | Values: string<br>Default: `global-web-application.xml`<br><br>The file name and path of the global Web application descriptor file. The default descriptor is *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config/global-web-application.xml`. |

## <import-shared-library>

**Parent element:** <shared-library>

**Required?** Optional; multiple allowed

Identifies a shared library to be imported by a shared library defined in the enclosing `<shared-library>` element. For additional information on configuring and using shared libraries, see the *Oracle Containers for J2EE Developer's Guide*.

***Table B–10 <import-shared-library> Attributes***

| Name | Description |
|---|---|
| `name` | Values: string<br>Default: required<br>The name of the shared library to import. |
| version | Values: string<br>Default: required<br>The version number to import. |

## <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 and 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–11 <init-param> Attributes***

| Name | Description |
|---|---|
| path | Values: string<br>Default: `global-web-application.xml`<br>The file name and path of the global Web application descriptor file. The default descriptor is *ORACLE_HOME*/j2ee/*instance*/config/global-web-application.xml. |

## <j2ee-logging-config>

**Parent element:** <application-server>

**Required?** Optional; only one allowed

**Child elements:**

Defines the file to use as the J2EE logging configuration file.

***Table B–12 <j2ee-logging-config> Attributes***

| Name | Description |
|---|---|
| path | Values: string<br>Default: `../j2ee-logging.xml`<br>The file name and path of the logger configuration file. |

## <java-compiler>

**Parent element:** <application-server>

**Required?** Optional; one only

**Child elements:**

Specifies configuration parameters for the Java compiler to use to compile EJB modules. By default, the `javac` compiler installed with the JDK defined in the `JAVA_HOME` environment variable will be used.

***Table B–13 <java-compiler> Attributes***

| Name | Description |
|---|---|
| name | Values: string<br>Default: `javac`<br>`modern\|classic\|javac\|ojc\|jikes`<br>The name of the Java compiler to use. |
| in-process | Values: Boolean<br>Default: `false`<br>Specifies whether to run the compiler in-process or out-of-process.<br>If set to `false`, a separate JVM process is spawned for the compiler to execute within. This is the default compiler execution mode used by OC4J, as it offers better management of memory resources.<br>If set to `true`, the compiler executes within the same JVM process as OC4J. The *`JAVA_HOME`*`/lib/tools.jar` must be located in the OC4J environment:<br>**For standalone OC4J**:<br>Copy `tools.jar` to *`JAVA_HOME`*`/jre/lib/ext` and start OC4J with the following command line option:<br>`-Xbootclasspath/a:`*`JAVA_HOME`*`/lib/tools.jar`<br>**For Oracle Application Server**:<br>Copy *`ORACLE_HOME`*`/jdk/lib/tools.jar` to *`ORACLE_HOME`*`/jdk/jre/lib/ext` and modify `opmn.xml` as follows before starting the server:<br>`<module-data>`<br>`   <category id="start-parameters">`<br>`      <data id="java-options" value="-server`<br>`      -Djava.security.policy=ORACLE_HOME/j2ee/`<br>`      oacore/config/java2.policy -Djava.awt.`<br>`      headless=true -Dhttp.webdir.enable=false`<br>`      -Xbootclasspath/a:ORACLE_HOME/jdk/lib/`<br>`      tools.jar"/>`<br>`   </category>`<br>`</module-data>` |
| `encoding` | Values: string<br>Default: `ISO-8859-1`<br>The source file encoding to use. |

*Table B–13 (Cont.) <java-compiler> Attributes*

| Name | Description |
|---|---|
| bindir | Values: string<br>Default: n/a<br>The absolute path to the directory containing the compiler executable. This attribute does not need to be specified to use the default `javac` compiler. |
| `extdir` | Values: string<br>Default: n/a<br>The compiler extension library location, if applicable. |
| `debug` | Values: Boolean<br>Default: `false`<br>Set to `true` to generate compilation-time debugging output. |

## <javacache-config>

**Parent element:** <application-server>

**Required?** Optional; only one allowed

**Child elements:** None

Specifies the path to `javacache.xml`, the Java Object Cache configuration file.

*Table B–14 <javacache-config> Attributes*

| Name | Description |
|---|---|
| path | Values: string<br>Default: `../../../javacache/admin/javacache.xml`<br>The path to the `javacache.xml` file. |

## <jms-config>

**Parent element:** <application-server>

**Required?** Optional; only one allowed

**Child elements:**

Specifies the file to use as the OC4J JMS configuration file.

*Table B–15 <jms-config> Attributes*

| Name | Description |
|---|---|
| path | Values: string<br>Default: `jms.xml`<br>The file name and path of the OC4J JMS configuration file. |

## <log>

**Parent element:** <application-server>

**Required?** Optional; only one allowed

**Child elements:** <file>

The enclosed <file> element points to the location of the OC4J server log file.

## <max-http-connections>

**Parent element:** <application-server>

**Required?** Optional; only one allowed

**Child elements:**

Defines the maximum number of concurrent connections any given Web site can accept at a single point in time. If text exists inside the tag, it is used as a redirect-URL when the limit is reached.

***Table B–16 <max-http-connections> Attributes***

| Name | Description |
|---|---|
| max-connections-queue-timeout | When the maximum number of connections are reached, this is the number of seconds that can pass before the connections are dropped and a message is returned to the client stating that the server is either busy or connections will be redirected.<br>Values: positive integer<br>Default: `10` seconds |
| socket-backlog | The number of connections to queue up before denying connections at the socket level.<br>Values: positive integer<br>Default: `30` |
| value | The maximum number of connections.<br>Values: positive integer<br>Default: integer maximum |

## <rmi-config>

**Parent element:** <application-server>

**Required?** Optional; only one allowed

**Child elements:**

Defines the file to use as the OC4J RMI configuration file.

***Table B–17 <rmi-config> Attributes***

| Name | Description |
|---|---|
| path | Values: string<br>Default: `rmi.xml`<br>The file name and path of the OC4J RMI configuration file. |

## <shared-library>

**Parent element:** <application-server>

**Required?** Optional; multiple allowed

**Child elements:** <code-source>, <import-shared-library>

Declares a shared library installed within the OC4J instance. For additional information on configuring and using shared libraries, see the *Oracle Containers for J2EE Developer's Guide*.

***Table B–18 <shared-library> Attributes***

| Name | Description |
|---|---|
| `name` | Values: string<br>Default: required<br>The name of the shared library directory created within the `/shared-lib` directory. |
| version | Values: string<br>Default: required<br>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<br>Default: `false`<br>This attribute is intended for internal use only. |

## <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 `<shutdown-classes>` element.

***Table B–19 <shutdown-class> Attributes***

| Name | Description |
|---|---|
| `classname` | Values: string<br>Default: required<br>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–20 <startup-class> Attributes*

| Name | Description |
|---|---|
| `classname` | Values: string<br>Default: required<br>The name of the class that implements the `oracle.j2ee.server.OC4JStartup` interface. |
| failure-is-fatal | Values: Boolean<br>Default: `false`<br>If `true`, OC4J logs an exception and exits when an exception is thrown. If `false`, OC4J logs the exception and continues. |

## <thread-pool>

**Parent element:** <application-server>

**Required?** Optional; multiple allowed

**Child elements:**

Contains the configuration for a single `system`, `http`, `jca`, `rmi request`, or `rmi connection` thread pool within an OC4J process. See "Configuring OC4J Thread Pools" on page 10-1 for details.

*Table B–21 <thread-pool> Attributes*

| Name | Description |
|---|---|
| `name` | Values: string<br>Default: required<br>`system\|rmi request\|rmi connection\|http\|jca`<br>The thread pool name, which must be one of these values:<br>▪ `system`<br>A hidden thread pool that was not exposed in the older format.<br>▪ `rmi request`<br>A thread pool that serves RMI requests.<br>▪ `rmi connection`<br>A thread pool whose threads block-read on the RMI connection.<br>▪ `http`<br>A thread pool serving HTTP and AJP requests and possibly RMI requests (if an `rmi request` thread pool is not configured) and RMI connections (if an `rmi connection` thread pool is not configured).<br>▪ `jca`<br>The work management thread pool, for the J2CA work manager.<br>The names of the threads in these pools are prefixed with `SystemThreadGroup_`, `RMIRequestThreadGroup_`, `RMIConnectionThreadGroup_`, `HTTPThreadGroup_`, and `WorkManager_`, respectively, and suffixed with an incrementing counter. |

*Table B–21 (Cont.) <thread-pool> Attributes*

| Name | Description |
|---|---|
| min | Values: string<br>Default: 0<br>The minimum number of threads that OC4J can simultaneously execute. |
| max | Values: string<br>Default: 1024<br>The maximum number of threads that OC4J can simultaneously execute. |
| queue | Values: string<br>Default: 0<br>The maximum number of requests that can be kept in the queue. |
| keepAlive | Values: string<br>Default: 600000<br>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.<br>To never destroy threads, set to -1.<br>The default value, 600000 milliseconds (10 minutes), is also the minimum value allowed if not -1. |
| stackSize | Values: string<br>Default: 0<br>The size of the thread pool stack. |
| debug | Values: Boolean<br>Default: false<br>If true, prints the application server thread pool information to the console at startup. If false, the thread pool information is not printed. |

## <transaction-manager-config>

**Parent element:** <application-server>

**Required?** Optional; only one allowed

**Child elements:**

Specifies the transaction manager configuration file.

*Table B–22 <transaction-manager-config> Attributes*

| Name | Description |
|---|---|
| path | Values: string<br>Default: transaction-manager.xml<br>The file name and path of the transaction manager configuration file. The default file is *ORACLE_HOME*/j2ee/*instance*/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 13, "Managing Web Sites in OC4J," for details.

***Table B–23 <web-site> Attributes***

| Name | Description |
|---|---|
| path | Values: string<br>Default: n/a<br><br>The file name and path of the `*-web-site.xml` configuration file defining the Web site. |

## <work-manager-thread-pool>

**Parent element:** <application-server>

**Required?** Optional; one only

**Child elements:**

Contains the configuration for a work management thread pool for resource adapters within an OC4J process. See "Configuring OC4J Thread Pools" on page 10-1 for details.

This element is deprecated. If the `server.xml` file contains this element, OC4J changes it to an equivalent <thread-pool> element that defines a `jca` thread pool.

***Table B–24 <work-manager-thread-pool> Attributes***

| Attribute | Description |
|---|---|
| `min` | Values: string<br>Default: n/a<br><br>The minimum number of threads to create in the work management thread pool. To disable the thread pool, set this value to `0`. |
| `max` | Values: string<br>Default: `40`<br><br>The maximum number of threads that can be created in the work management thread pool.<br><br>The work management thread pool uses three worker threads for internal use. For example, if you specify `max="16"`, then only 13 worker threads are available to service requests. Similarly, if the `max` value is `20`, then only 17 threads are available. So you need to set this value to your required maximum number of threads plus 3. |

*Table B–24 (Cont.) <work-manager-thread-pool> Attributes*

| Attribute | Description |
|---|---|
| queue | Values: string<br>Default: 0<br><br>The maximum number of threads that can be kept in the queue in the work management thread pool. If you use the default, 0, no queue is maintained to handle a sudden burst of work requests. |
| keepAlive | Values: string<br>Default: 600000<br><br>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.<br><br>To never destroy threads, set to -1.<br><br>The default value, 600000 milliseconds (10 minutes), is also the minimum value allowed if not -1. |
| debug | Values: Boolean<br>Default: false<br><br>If true, prints the application server work management thread pool information to the console at startup. If false, the thread pool information is not printed. |

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

The element descriptions in this section apply to any OC4J Web site configuration file, including default-web-site.xml.

## <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–25 Web Site Configuration File Attributes*

| Name | Description |
|---|---|
| custom-thread-pool | Values: string<br>Default: n/a<br><br>Optionally specifies a custom thread pool to be used by each application bound to this Web site by a <web-app> element in this configuration file. |
| display-name | Values: string<br>Default: n/a<br><br>Optionally defines a user-friendly or informal Web site name. |

***Table B–25 (Cont.) Web Site Configuration File Attributes***

| Name | Description |
|---|---|
| `host` | Values: string<br>Default: n/a<br><br>Specifies the host for this Web site, as either a DNS host name or an IP address. If a server is a *multihome* machine (having multiple IP addresses), you can use the `ALL` setting to listen to all IP addresses. |
| `log-request-info` | Values: Boolean<br>Default: `false`<br><br>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. ("Enabling or Disabling Access Logging for a Web Module or Application" on page 13-16 provides additional information about enabling the Web site log.) |
| `max-request-size` | Values: string<br>Default: `15000`<br><br>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. |
| `secure` | Values: Boolean<br>Default: `false`<br><br>Specifies whether to support Secure Socket Layer (SSL) functionality.<br><br>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 standalone OC4J), a `true` setting results in HTTPS protocol between the client and OC4J.<br><br>Also, 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.<br><br>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<br>Default: n/a<br><br>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`. Each protocol must have a corresponding port, and each port must have a corresponding protocol.<br><br>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.<br><br>Changing the value of the `protocol` attribute does not reset the port range to the default range for the protocol you specify. To change the port range, you can use the `range` attribute. |

*Table B–25 (Cont.) Web Site Configuration File Attributes*

| Name | Description |
|---|---|
| port | Values: string<br>Default: n/a<br><br>Specifies the port number for this Web site. Each port must have a corresponding protocol, and each protocol must have a corresponding port. In standalone OC4J, a port setting of 8888 is used by default for direct access to the OC4J listener, but you can change this as desired.<br><br>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 Web Cache enabled.<br><br>In a UNIX environment, port numbers less than 1024 require root privileges for access. Also, if there is no port specification from the client browser, port 80 is assumed for HTTP protocol and port 443 for HTTPS.<br><br>Changing the value of the protocol attribute does not reset the port range to the default range for the protocol you specify. To change the port range, you can use the range attribute. |
| use-keep-alive | Values: Boolean<br>Default: true<br><br>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<br>Default: n/a<br><br>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>

Contains an optional, brief description of the Web site.

## <frontend>

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**, a technique for encoding a session ID into the URL.

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.

Table B–26 describes the attributes of <frontend>.

*Table B–26 <frontend> Attributes*

| Name | Description |
|---|---|
| host | Values: string<br>Default: n/a<br>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's root attribute.

> **Note:** It is possible to deploy a WAR file by itself, instead of within an EAR file. In standalone OC4J, such Web applications are added to the OC4J default application. (In OC4J, there must always be a parent application of some sort.) See "Overview of the Application Hierarchy in OC4J" on page 1-10 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 "Deploying a J2EE Application (EAR)" on page 6-9 (for a typical scenario of deploying a WAR file within an EAR file) and "Deploying a Standalone Web Module (WAR)" on page 6-12 (for the scenario of deploying a WAR file by itself to the OC4J default application).

Table B–27 describes the attributes of <web-app>.

*Table B–27 <web-app> Attributes*

| Name | Description |
|---|---|
| access-log | Values: string<br>Default: false<br>Specifies whether OC4J access logging, which logs requests to the Web site, is enabled for the Web module. If you want to enable access logging, set to true. If log file management becomes an issue, set to false to disable access logging for the module.<br>For more on access log configuration, see the descriptions of the <access-log> and <odl-access-log> elements within this section. |

***Table B–27 (Cont.) <web-app> Attributes***

| Name | Description |
|---|---|
| application | Values: string<br>Default: n/a<br><br>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.<br><br>If you deploy a WAR file by itself in standalone OC4J, 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 | Values: Boolean<br>Default: false<br><br>Optional. Specifies whether the Web module should be preloaded 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 or application is deployed with Oracle Enterprise Manager 10*g* Application Server Control. |
| max-inactivity-time | Values: string<br>Default: 0<br><br>Optional. Specifies 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 | Values: Boolean<br>Default: n/a<br><br>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. |

***Table B–27 (Cont.) <web-app> Attributes***

| Name | Description |
|---|---|
| `root` | Values: string<br>Default: n/a<br><br>Specifies the path to which the Web module is to be bound. This attribute defines the context root 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 context root, `/catalog`, then the module can be invoked as follows:<br><br>`http://www.example.com/catalog`<br><br>The `root` attribute overrides the `<context-root>` value of the corresponding `<web>` element in the J2EE `application.xml` file. Specifying a slash (`/`) as the context root will override the OC4J default Web application.<br><br>You can use `/` as the context root when you deploy an application. The following example uses `admin_client.jar` to deploy a WAR file and bind it to `/`:<br><br>`% java -jar admin_client.jar deployer:oc4j:localhost oc4jadmin welcome1 \`<br>`  -deploy -file d:my-web-store.war -deploymentName mws_2 \`<br>`  -bindAllWebApps -contextRoot "/"`<br><br>If an EAR file includes an `application.xml` file that has the context root set to `/`, such as in the following example, then `/` will be the default context root when the application is deployed.<br><br>`<application>`<br>`  <display-name>Web-Store</display-name>`<br>`  <module>`<br>`    <web>`<br>`      <web-uri>my-web-store.war</web-uri>`<br>`      <context-root>/</context-root>`<br>`    </web>`<br>`  </module>`<br>`</application>`<br><br>Because the default ping URL for Oracle HTTP Server is also a slash (`/`), using `/` as the context root when you deploy an application might result in either or both of the following problems:<br><br>■ Pings intended for Oracle HTTP Server go directly to OC4J instead.<br>■ Extraneous `HEAD` requests appear in the `*-web-access.log` file.<br><br>You can avoid these problems by placing an `Oc4jMountCopy off` directive in the *ORACLE_HOME*`/Apache/Apache/conf/dms.conf` file. |

***Table B–27 (Cont.) <web-app> Attributes***

| Name | Description |
|---|---|
| shared | Values: string<br>Default: `false`<br><br>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). Use `shared="true"` only in standalone OC4J.<br><br>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 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. |

## <default-web-app>

This element creates a reference to the default Web application bound to this Web site. When a single application is bound to the Web site, such as Application Server Control, specify the application within this element.

For users, this element is relevant only in a standalone OC4J environment. In an Oracle Application Server environment, the OC4J default Web application has system-level functionality but is not otherwise meaningful.

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

## <user-web-apps>

Use this element to support user directories and applications. Each user can have a Web module and associated `web-application.xml` file. User applications are reached at /*username*/ from the server root.

Table B–28 describes the attributes of `<user-web-apps>`.

***Table B–28 <user-web-apps> Attributes***

| Name | Description |
|---|---|
| max-inactivity-time | Values: int<br>Default: n/a<br><br>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 a UNIX environment, for example, is /home/*username*, in which *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.

See "Configuring Text-Based Access Logging" on page 13-13 for configuration details.

## <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 "Configuring ODL Access Logging" on page 13-15 for configuration details.

## <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`.

Subelement of `<ssl-config>`:

`<property>`

Table B–29 describes the attributes of `<ssl-config>`.

***Table B–29 <ssl-config> Attributes***

| Name | Description |
|---|---|
| keystore | Values: string<br>Default: n/a<br><br>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.<br><br>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` | Values: string<br>Default: n/a<br><br>The password required to open the keystore. |
| `needs-client-auth` | Values: string<br>Default: `false`<br><br>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` | Values: string<br>Default: `com.sun.net.ssl.internal.ssl.Provider`<br><br>You can use this attribute to specify a provider if you are using JSSE (Java Secure Socket Extension).<br><br>By default, OC4J usually employs the Sun Microsystems implementation of SSL. However, OC4J employs the Oracle SSL implementation in some cases, such as for SOAP and `http_client`. |

*Table B–29 (Cont.) <ssl-config> Attributes*

| Name | Description |
|---|---|
| factory | Values: string<br>If you are not using JSSE, use the factory attribute to specify an implementation of SSLServerSocketFactory.<br>If you use a third-party SSLServerSocketFactory implementation, you can use <property> subelements of the <ssl-config> element to send parameters to the factory.<br>The factory attribute and its parameters are deprecated. |

# 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 9-9 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 can 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 user name 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 values together compose the primary key.

*Table C–2 OC4J_HTTP_SESSION_VALUE Table 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 user name 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. |

# D

# Third Party Licenses

This appendix includes the Third Party License for all the third party products included with Oracle Application Server.

## ANTLR

This program contains third-party code from ANTLR. Under the terms of the Apache license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the ANTLR software, and the terms contained in the following notices do not change those rights.

### The ANTLR License

```
Software License

We reserve no legal rights to the ANTLR--it is fully in the public domain. An
individual or company may do whatever they wish with source code distributed with
ANTLR or the code generated by ANTLR, including the incorporation of ANTLR, or its
output, into commerical software.
We encourage users to develop software with ANTLR. However, we do ask that credit
is given to us for developing ANTLR. By "credit", we mean that if you use ANTLR or
incorporate any source code into one of your programs (commercial product,
research project, or otherwise) that you acknowledge this fact somewhere in the
documentation, research report, etc... If you like ANTLR and have developed a nice
tool with the output, please mention that you developed it using ANTLR. In
addition, we ask that the headers remain intact in our source code. As long as
these guidelines are kept, we expect to continue enhancing this system and expect
to make other tools available as they are completed.
```

## Apache

This program contains third-party code from the Apache Software Foundation ("Apache"). Under the terms of the Apache license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Apache software, and the terms contained in the following notices do not change those rights.

The Apache license agreements apply to the following included Apache components:

- Apache HTTP Server
- Apache JServ
- mod_jserv

- Regular Expression package version 1.3
- Apache Expression Language packaged in commons-el.jar
- mod_mm 1.1.3
- Apache XML Signature and Apache XML Encryption v. 1.4 for Java and 1.0 for C++
- log4j 1.1.1
- BCEL v. 5
- XML-RPC v. 1.1
- Batik v. 1.5.1
- ANT 1.6.2 and 1.6.5
- Crimson v. 1.1.3
- ant.jar
- wsif.jar
- bcel.jar
- soap.jar
- Jakarta CLI 1.0
- jakarta-regexp-1.3.jar
- JSP Standard Tag Library 1.0.6 and 1.1
- Struts 1.1
- Velocity 1.3
- svnClientAdapter
- commons-logging.jar
- wsif.jar
- commons-el.jar
- standard.jar
- jstl.jar

## The Apache Software License

### License for Apache Web Server 1.3.29

```
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. The end-user documentation included with the redistribution,
 *    if any, must include the following acknowledgment:
 *       "This product includes software developed by the
 *        Apache Software Foundation (http://www.apache.org/)."
 *    Alternately, this acknowledgment may appear in the software itself,
 *    if and wherever such third-party acknowledgments normally appear.
 *
 * 4. The names "Apache" and "Apache Software Foundation" must
 *    not be used to endorse or promote products derived from this
 *    software without prior written permission. For written
 *    permission, please contact apache@apache.org.
 *
 * 5. Products derived from this software may not be called "Apache",
 *    nor may "Apache" appear in their name, without prior written
 *    permission of the Apache Software Foundation.
 *
 * THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 * ====================================================================
 *
 * This software consists of voluntary contributions made by many
 * individuals on behalf of the Apache Software Foundation.  For more
 * information on the Apache Software Foundation, please see
 * <http://www.apache.org/>.
 *
 * Portions of this software are based upon public domain software
 * originally written at the National Center for Supercomputing
Applications,
 * University of Illinois, Urbana-Champaign.
```

### License for Apache Web Server 2.0

Copyright (c) 1999-2004, The Apache Software Foundation
Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.  You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  See the License for the specific language governing permissions and limitations under the License.
Copyright (c) 1999-2004, The Apache Software Foundation

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions

```
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.
```

## Apache SOAP

This program contains third-party code from the Apache Software Foundation ("Apache"). Under the terms of the Apache license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Apache software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the Apache software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Apache.

## Apache SOAP License

Apache SOAP license 2.3.1

```
Copyright (c) 1999 The Apache Software Foundation.  All rights reserved.
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
```

# DBI Module

This program contains third-party code from Tim Bunce. Under the terms of the Tim Bunce license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Tim Bunce software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the Tim Bunce software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Tim Bunce.



You may distribute under the terms of either the GNU General Public License or the Artistic License, as specified in the Perl README file.

## Perl Artistic License

The "Artistic License"

### Preamble

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

### Definitions

"Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.

"Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or copyrights for the package.

"You" is you, if you're thinking about copying or distributing this Package.

"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.
2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.
3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a. place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b. use the modified Package only within your corporation or organization.

c. rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d. make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

   a. distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

   b. accompany the distribution with the machine-readable source of the Package with your modifications.

   c. give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

   d. make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package's interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whoever generated them, and may be sold commercially, and may be aggregated with this Package. If such scripts or library files are aggregated with this Package through the so-called "undump" or "unexec" methods of producing a binary executable image, then distribution of such an image shall neither be construed as a distribution of this Package nor shall it fall under the restrictions of Paragraphs 3 and 4, provided that you do not represent such an executable image as a Standard Version of this Package.

7. C subroutines (or comparably compiled subroutines in other languages) supplied by you and linked into this Package in order to emulate subroutines and variables of the language defined by this Package shall not be considered part of this Package, but are the equivalent of input as in Paragraph 6, provided these subroutines do not change the language in any way that would cause it to fail the regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always permitted provided that the use of this Package is embedded; that is, when no overt attempt

is made to make this Package's interfaces visible to the end user of the commercial distribution. Such use shall not be construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

# FastCGI

This program contains third-party code from Open Market, Inc. Under the terms of the Open Market license, Oracle is required to license the Open Market software to you under the following terms. Note that the terms contained in the Oracle program license that accompanied this product do not apply to the Open Market software, and your rights to use the software are solely as set forth below. Oracle is not responsible for the performance of the Open Market software, does not provide technical support for the software, and shall not be liable for any damages arising out of any use of the software.

## FastCGI Developer's Kit License

This FastCGI application library source and object code (the "Software") and its documentation (the "Documentation") are copyrighted by Open Market, Inc ("Open Market"). The following terms apply to all files associated with the Software and Documentation unless explicitly disclaimed in individual files.

Open Market permits you to use, copy, modify, distribute, and license this Software and the Documentation solely for the purpose of implementing the FastCGI specification defined by Open Market or derivative specifications publicly endorsed by Open Market and promulgated by an open standards organization and for no other purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions.

No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this Software and Documentation may be copyrighted by their authors and need not follow the licensing terms described here, but the modified Software and Documentation must be used for the sole purpose of implementing the FastCGI specification defined by Open Market or derivative specifications publicly endorsed by Open Market and promulgated by an open standards organization and for no other purpose. If modifications to this Software and Documentation have new licensing terms, the new terms must protect Open Market's proprietary rights in the Software and Documentation to the same extent as these licensing terms and must be clearly indicated on the first page of each file where they apply.

Open Market shall retain all right, title and interest in and to the Software and Documentation, including without limitation all patent, copyright, trade secret and other proprietary rights.

OPEN MARKET MAKES NO EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE SOFTWARE OR THE DOCUMENTATION, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL OPEN MARKET BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY DAMAGES ARISING FROM OR RELATING

TO THIS SOFTWARE OR THE DOCUMENTATION, INCLUDING, WITHOUT LIMITATION, ANY INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES OR SIMILAR DAMAGES, INCLUDING LOST PROFITS OR LOST DATA, EVEN IF OPEN MARKET HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS". OPEN MARKET HAS NO LIABILITY IN CONTRACT, TORT, NEGLIGENCE OR OTHERWISE ARISING OUT OF THIS SOFTWARE OR THE DOCUMENTATION.

### Module mod_fastcgi License

This FastCGI application library source and object code (the "Software") and its documentation (the "Documentation") are copyrighted by Open Market, Inc ("Open Market"). The following terms apply to all files associated with the Software and Documentation unless explicitly disclaimed in individual files.

Open Market permits you to use, copy, modify, distribute, and license this Software and the Documentation solely for the purpose of implementing the FastCGI specification defined by Open Market or derivative specifications publicly endorsed by Open Market and promulgated by an open standards organization and for no other purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions.

No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this Software and Documentation may be copyrighted by their authors and need not follow the licensing terms described here, but the modified Software and Documentation must be used for the sole purpose of implementing the FastCGI specification defined by Open Market or derivative specifications publicly endorsed by Open Market and promulgated by an open standards organization and for no other purpose. If modifications to this Software and Documentation have new licensing terms, the new terms must protect Open Market's proprietary rights in the Software and Documentation to the same extent as these licensing terms and must be clearly indicated on the first page of each file where they apply.

Open Market shall retain all right, title and interest in and to the Software and Documentation, including without limitation all patent, copyright, trade secret and other proprietary rights.

OPEN MARKET MAKES NO EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE SOFTWARE OR THE DOCUMENTATION, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL OPEN MARKET BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY DAMAGES ARISING FROM OR RELATING TO THIS SOFTWARE OR THE DOCUMENTATION, INCLUDING, WITHOUT LIMITATION, ANY INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES OR SIMILAR DAMAGES, INCLUDING LOST PROFITS OR LOST DATA, EVEN IF OPEN MARKET HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS". OPEN MARKET HAS NO LIABILITY IN CONTRACT, TORT, NEGLIGENCE OR OTHERWISE ARISING OUT OF THIS SOFTWARE OR THE DOCUMENTATION.

## Info-ZIP Unzip Package

This program contains third-party code from Info-ZIP. Under the terms of the Info-ZIP license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Info-ZIP software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the Info-ZIP software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Info-ZIP.

## The Info-ZIP Unzip Package License



# JSR 110

This program contains third-party code from IBM Corporation ("IBM"). Under the terms of the IBM license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the IBM software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the IBM software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or IBM.



```
Java APIs for the WSDL specification are available at:
http://www-124.ibm.com/developerworks/projects/wsdl4j/
```

# Jaxen

This program contains third-party code from the Apache Software Foundation ("Apache") and from the Jaxen Project ("Jaxen"). Under the terms of the Apache and Jaxen licenses, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Apache and Jaxen software, and the terms contained in the following notices do not change those rights.

## The Jaxen License

software without prior written permission. For written permission, please contact license@jaxen.org.

Products derived from this software may not be called "Jaxen", nor may "Jaxen" appear in their name, without prior written permission from the Jaxen Project Management (pm@jaxen.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgment equivalent to the following: "This product includes software developed by the Jaxen Project (http://www.jaxen.org/)." Alternatively, the acknowledgment may be graphical using the logos available at http://www.jaxen.org/.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE Jaxen AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Jaxen Project and was originally created by bob mcwhirter and James Strachan . For more information on the Jaxen Project, please see http://www.jaxen.org/.

# JGroups

This program contains third-party code from GNU. Under the terms of the GNU license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the GNU software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the GNU software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or GNU.

## The GNU License

GNU Lesser General Public License
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble
The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the

Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept

this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask

for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS
How to Apply These Terms to Your New Libraries
If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and an idea of what it does.> Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

## mod_mm and mod_ssl

This program contains third-party code from Ralf S. Engelschall ("Engelschall"). Under the terms of the Engelschall license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Engelschall software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the mod_mm software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Engelschall.

### mod_mm



### mod_ssl



# OpenSSL

This program contains third-party code from the OpenSSL Project. Under the terms of the OpenSSL Project license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the OpenSSL software, and the terms contained in the following notices do not change those rights.

## OpenSSL License

```
*    being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
*    the apps directory (application code) you must include an acknowledgement:
*    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed.  i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/
```

# Perl

This program contains third-party code from the Comprehensive Perl Archive Network ("CPAN"). Under the terms of the CPAN license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the CPAN software, and the terms contained in the following notices do not change those rights.

## Perl Kit Readme



This program is free software; you can redistribute it and/or modify it under the terms of either:

1. the GNU General Public License as published by the Free Software Foundation; either version 1, or (at your option) any later version, or
2. the "Artistic License" which comes with this Kit.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See either the GNU General Public License or the Artistic License for more details.

You should have received a copy of the Artistic License with this Kit, in the file named "Artistic". If not, I'll be glad to provide one.

You should also have received a copy of the GNU General Public License along with this program in the file named "Copying". If not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA or visit their Web page on the internet at http://www.gnu.org/copyleft/gpl.html.

For those of you that choose to use the GNU General Public License, my interpretation of the GNU General Public License is that no Perl script falls under the terms of the GPL unless you explicitly put said script under the terms of the GPL yourself. Furthermore, any object code linked with perl does not automatically fall under the terms of the GPL, provided such object code only adds definitions of subroutines and variables, and does not otherwise impair the resulting interpreter from executing any standard Perl script. I consider linking in C subroutines in this manner to be the moral equivalent of defining subroutines in the Perl language itself. You may sell such an object file as proprietary provided that you provide or offer to provide the Perl source, as specified by the GNU General Public License. (This is merely an alternate way of specifying input to the program.) You may also sell a binary produced by the dumping of a running Perl script that belongs to you, provided that you provide or offer to provide the Perl source as specified by the GPL. (The fact that a Perl interpreter and your code are in the same binary file is, in this case, a form of mere aggregation.) This is my interpretation of the GPL. If you still have concerns or difficulties understanding my intent, feel free to contact me. Of course, the Artistic License spells all this out for your protection, so you may prefer to use that.

## mod_perl 1.29 License

```
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 * ====================================================================
 */
```

## mod_perl 1.99_16 License

```
Copyright (c) 1999-2004, The Apache Software Foundation
Licensed under the Apache License, Version 2.0 (the "License"); you may not use
this file except in compliance with the License.  You may obtain a copy of the
License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied.  See the License for the
specific language governing permissions and limitations under the License.
Copyright (c) 1999-2004, The Apache Software Foundation
                                Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).
```

```
    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

 2. Grant of Copyright License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    copyright license to reproduce, prepare Derivative Works of,
    publicly display, publicly perform, sublicense, and distribute the
    Work and such Derivative Works in Source or Object form.

 3. Grant of Patent License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    (except as stated in this section) patent license to make, have made,
    use, offer to sell, sell, import, and otherwise transfer the Work,
    where such license applies only to those patent claims licensable
    by such Contributor that are necessarily infringed by their
    Contribution(s) alone or by combination of their Contribution(s)
    with the Work to which such Contribution(s) was submitted. If You
    institute patent litigation against any entity (including a
    cross-claim or counterclaim in a lawsuit) alleging that the Work
    or a Contribution incorporated within the Work constitutes direct
    or contributory patent infringement, then any patent licenses
    granted to You under this License for that Work shall terminate
    as of the date such litigation is filed.

 4. Redistribution. You may reproduce and distribute copies of the
    Work or Derivative Works thereof in any medium, with or without
    modifications, and in Source or Object form, provided that You
    meet the following conditions:

    (a) You must give any other recipients of the Work or
        Derivative Works a copy of this License; and

    (b) You must cause any modified files to carry prominent notices
        stating that You changed the files; and
```

```
    (c) You must retain, in the Source form of any Derivative Works
        that You distribute, all copyright, patent, trademark, and
        attribution notices from the Source form of the Work,
        excluding those notices that do not pertain to any part of
        the Derivative Works; and

    (d) If the Work includes a "NOTICE" text file as part of its
        distribution, then any Derivative Works that You distribute must
        include a readable copy of the attribution notices contained
        within such NOTICE file, excluding those notices that do not
        pertain to any part of the Derivative Works, in at least one
        of the following places: within a NOTICE text file distributed
        as part of the Derivative Works; within the Source form or
        documentation, if provided along with the Derivative Works; or,
        within a display generated by the Derivative Works, if and
        wherever such third-party notices normally appear. The contents
        of the NOTICE file are for informational purposes only and
        do not modify the License. You may add Your own attribution
        notices within Derivative Works that You distribute, alongside
        or as an addendum to the NOTICE text from the Work, provided
        that such additional attribution notices cannot be construed
        as modifying the License.

    You may add Your own copyright statement to Your modifications and
    may provide additional or different license terms and conditions
    for use, reproduction, or distribution of Your modifications, or
    for any such Derivative Works as a whole, provided Your use,
    reproduction, and distribution of the Work otherwise complies with
    the conditions stated in this License.

 5. Submission of Contributions. Unless You explicitly state otherwise,
    any Contribution intentionally submitted for inclusion in the Work
    by You to the Licensor shall be under the terms and conditions of
    this License, without any additional terms or conditions.
    Notwithstanding the above, nothing herein shall supersede or modify
    the terms of any separate license agreement you may have executed
    with Licensor regarding such Contributions.

 6. Trademarks. This License does not grant permission to use the trade
    names, trademarks, service marks, or product names of the Licensor,
    except as required for reasonable and customary use in describing the
    origin of the Work and reproducing the content of the NOTICE file.

 7. Disclaimer of Warranty. Unless required by applicable law or
    agreed to in writing, Licensor provides the Work (and each
    Contributor provides its Contributions) on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
    implied, including, without limitation, any warranties or conditions
    of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
    PARTICULAR PURPOSE. You are solely responsible for determining the
    appropriateness of using or redistributing the Work and assume any
    risks associated with Your exercise of permissions under this License.

 8. Limitation of Liability. In no event and under no legal theory,
    whether in tort (including negligence), contract, or otherwise,
    unless required by applicable law (such as deliberate and grossly
    negligent acts) or agreed to in writing, shall any Contributor be
    liable to You for damages, including any direct, indirect, special,
    incidental, or consequential damages of any character arising as a
```

```
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.
```

## Perl Artistic License

The "Artistic License"

### Preamble

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

### Definitions

"Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.

"Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or copyrights for the package.

"You" is you, if you're thinking about copying or distributing this Package.

"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.
2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.
3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a. place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b. use the modified Package only within your corporation or organization.

c. rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d. make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

   a. distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

   b. accompany the distribution with the machine-readable source of the Package with your modifications.

   c. give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

   d. make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package's interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whoever generated them, and may be sold commercially, and may be aggregated with this Package. If such scripts or library files are aggregated with this Package through the so-called "undump" or "unexec" methods of producing a binary executable image, then distribution of such an image shall neither be construed as a distribution of this Package nor shall it fall under the restrictions of Paragraphs 3 and 4, provided that you do not represent such an executable image as a Standard Version of this Package.

7. C subroutines (or comparably compiled subroutines in other languages) supplied by you and linked into this Package in order to emulate subroutines and variables of the language defined by this Package shall not be considered part of this Package, but are the equivalent of input as in Paragraph 6, provided these subroutines do not change the language in any way that would cause it to fail the regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always permitted provided that the use of this Package is embedded; that is, when no overt attempt

is made to make this Package's interfaces visible to the end user of the commercial distribution. Such use shall not be construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.
10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

# SAXPath

This program contains third-party code from SAXPath. Under the terms of the SAXPath license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the SAXPath software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the SAXPath software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or SAXPath.

## The SAXPath License

```
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made
by many individuals on behalf of the SAXPath Project and was originally created by
bob mcwhirter and James Strachan . For more information on the SAXPath Project,
please see http://www.saxpath.org/.
```

# W3C DOM

This program contains third-party code from the World Wide Web Consortium ("W3C"). Under the terms of the W3C license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the W3C software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the W3C software is provided by Oracle AS IS and without warranty or support of any kind from Oracle or W3C.

## The W3C License

```
W3C® SOFTWARE NOTICE AND LICENSE
http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
This work (and included software, documentation such as READMEs, or other related
items) is being provided by the copyright holders under the following license. By
obtaining, using and/or copying this work, you (the licensee) agree that you have
read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation,
with or without modification, for any purpose and without fee or royalty is hereby
granted, provided that you include the following on ALL copies of the software and
documentation or portions thereof, including modifications:

The full text of this NOTICE in a location viewable to users of the redistributed
or derivative work.
Any pre-existing intellectual property disclaimers, notices, or terms and
conditions. If none exist, the W3C Software Short Notice should be included
(hypertext is preferred, text is permitted) within the body of any redistributed
or derivative code.
Notice of any changes or modifications to the files, including the date changes
were made. (We recommend you provide URIs to the location from which the code is
derived.)
THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO
REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO,
WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE
USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS,
COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or
publicity pertaining to the software without specific, written prior permission.
Title to copyright in this software and any associated documentation will at all
times remain with copyright holders.
```

# Index

## E



## G



## H



## I



## J



## K



## L



## M

## T



## U



## V



## W



## X