Oracle® Containers for J2EE

Deployment Guide

10*g* (10.1.3.5.0)

E13980-01

July 2009

Oracle Containers for J2EE Deployment Guide, 10*g* (10.1.3.5.0)

E13980-01

Primary Author: Joseph Ruzzi

Contributing Author: Bonnie Vaughan, Dan Hynes

Contributors: Angela Barone, Steve Button, Lars Ewe, Tugdual Grall, Gerald Ingalls, Mike Lehmann, Jianmin Liu, Jasen Minton, Debu Panda, Shiva Prasad, Chaya Ramanujam, Merrick Schincariol, Charlie Shapiro, John Speidel, Gael Stevens, Sindhu Subramanyam

Contents

Preface xi

Intended Audience xi
Documentation Accessibility xi
Related Documents xii
Conventions xiii

1 Getting Started

Overview of Deployment in OC4J 1-1
Valid Components for Deployment 1-1
Support for the J2EE Application Deployment API (JSR-88) in OC4J 1-2
Hot Deployment in OC4J 1-2
Deployment of Applications 1-2
Deployment of Shared Libraries 1-2
Options for Deploying Applications to OC4J 1-3
Application Server Control User Interface 1-3
OC4J Ant Tasks 1-3
The admin_client.jar Command-Line Utility 1-4
The admin.jar Command-Line Utility 1-4
Oracle JDeveloper 1-4

2 Deploying and Undeploying J2EE Applications in OC4J

Overview of the Application Deployment Process 2-1
Designating a Parent Application 2-1
Binding a Web Application to a Web Site 2-2
Creating or Applying a Deployment Plan 2-2
Using Dynamic HTTP Server Mount Points in Oracle Application Server 2-2
Impact of JDK Version on Deployed Applications 2-3
Example of a Deployed Application Directory Structure 2-4
OC4J Application Deployment Process 2-4
Overview of Redeploying an Application 2-5
Restart of OC4J After RMI or Manual Reconfiguration 2-5
Impact of Redeploying a Parent Application 2-6
Overview of Undeploying an Application 2-6
Results of Removing an Application from OC4J 2-6
Impact of Undeploying a Parent Application 2-6

Results of Errors in Deployment Descriptors 2-6

3 Deploying Enterprise JavaBeans Modules

Overview of EJB Deployment 3-1
Generation of Client-Side IIOP Stubs 3-2
Generating Stubs with Application Server Control 3-2
Generating Stubs with admin_client.jar 3-3
Generating Stubs with the OC4J Ant Tasks 3-3
Incremental Redeployment of Updated EJB Modules 3-4
Impact of EJB Redeployment on Application Clients 3-5
Impact of Redeploying Session Beans 3-5
Stateless Session Beans 3-5
Stateful Session Beans 3-5
Impact of Redeploying Entity Beans 3-5
Bean-Managed Persistence Beans 3-6
Container-Managed Persistence Beans 3-6

4 Deploying Large Applications

Specifying the Compilation Mode to Use 4-1
Configuring the Java Compiler 4-2
Specifying an Alternative Java Compiler 4-2
Configuring Out-of-Process or In-Process Compiler Execution 4-3
Summary of Java Compiler Configuration Parameters 4-3
Tuning the OC4J JVM for Large Deployments 4-4

5 Deploying Web Modules

Deploying a Standalone Web Module 5-1
Redeploying a Standalone Web Application 5-1
Redeploying an Updated Web Module into an Existing Application 5-2
Adding or Modifying JavaServer Pages in an Active Web Module 5-2
Setting JSP Configuration Parameters in the XML Configuration File 5-3

6 Deploying Resource Adapters

Deploying a Standalone RAR 6-1
Deploying a Resource Adapter with Dependencies 6-2
Deploying Multiple Versions of a Standalone RAR 6-2
Redeploying or Undeploying a Standalone RAR 6-2

7 Deploying Web Services

Deploying a Web Service 7-1
Redeploying a Web Service 7-1

8 Working with Deployment Plans

Deployment Plan Overview 8-1
How Deployment Plans Interact with Packaged Deployment Descriptors 8-2

Overview of J2EE and OC4J Deployment Descriptors 8-2
Creating or Editing a Deployment Plan 8-3
Creating or Editing Deployment Plans with Application Server Control 8-4
Creating or Editing Deployment Plans with Oracle JDeveloper 8-4
Setting Properties in a Deployment Plan 8-4
Setting J2EE Application Configuration Properties 8-5
Setting Web Module Configuration Properties 8-11
Setting Enterprise JavaBeans Module Configuration Properties 8-17
Setting General EJB Properties 8-17
Setting Entity Bean Properties 8-18
Setting Session Bean Properties 8-21
Setting Message-Driven Bean Properties 8-23
Setting Web Services Configuration Properties 8-24
Setting General Web Services Properties 8-24
Setting Web Service Description Properties 8-25
Setting Application Client Configuration Properties 8-25
Setting Resource Adapter Properties 8-26

9 Using Application Server Control for Deployment

Accessing Application Server Control 9-2
Accessing Application Server Control in Standalone OC4J 9-2
Accessing Application Server Control in Oracle Application Server 9-2
Setting Log Levels 9-3
Deploying an Application to an OC4J Instance or Group of Instances 9-3
Deploying or Redeploying an Application 9-4
Redeploying an Application with Scheduled Jobs 9-5
Completing Configuration Tasks Before Deployment 9-5
Selecting the Security Provider 9-5
Mapping Security Roles to Users and User Groups 9-6
Configuring Enterprise JavaBeans Modules Included in the Application 9-6
Managing Class Loading to Import Shared Libraries 9-7
Configuring Application Clustering 9-8
Providing Resource Mappings 9-9
Deploying a New Application Version with Side-by-Side Application Upgrade 9-9
How to Create an Additional HTTP Listener in an OC4J Instance 9-11
How to Deploy an Application with Side-by-Side Application Upgrade 9-12
Undeploying an Application 9-18
Creating and Managing Shared Libraries 9-18
Starting, Restarting, and Stopping Applications 9-19
Restarting and Stopping OC4J Instances 9-20
Managing Data Sources and Connection Pools for OC4J Instances 9-20
Managing JMS Resources 9-21

10 Using OC4J Ant Tasks for Deployment

Preparing to Use OC4J Ant Tasks 10-3
Meeting Prerequisites for Using OC4J Ant Tasks 10-4

Incorporating OC4J Ant Tasks into Your Environment 10-4
Incorporating OC4J Ant Tasks Using Ant 1.6.5 Outside OC4J 10-5
Incorporating OC4J Ant tasks Using Ant 1.6.5 with the Administrative Client Utility 10-6
Setting the Deployer URI 10-7
Invoking a Task on a Group of OC4J Instances 10-7
Invoking a Task on a Specific OC4J Instance 10-7
Invoking a Task on a Standalone OC4J Server 10-8
Enabling Java Logging 10-8
Invoking OC4J Ant Tasks 10-9
Adding Web Sites 10-10
Deploying an Archive 10-11
Deploying a J2EE Application (EAR) 10-11
Deploying a Standalone Web Module (WAR) 10-13
Deploying a Standalone EJB Module (JAR) 10-14
Deploying a Standalone Resource Adapter (RAR) 10-15
Managing Web Bindings 10-16
Binding Web Modules to a Web Site After Deployment 10-16
Binding All Web Modules to a Single Web Site 10-16
Binding a Specific Web Module to a Web Site and Setting the Context Root 10-17
Unbinding Web Modules from a Web Site 10-18
Unbinding All Web Modules 10-18
Unbinding a Specific Web Module 10-19
Listing Web Bindings 10-20
Redeploying an Archive 10-20
Specifying a Delay Between Sequential Redeployments in a Cluster 10-21
Redeploying an Application with Scheduled Jobs 10-22
Undeploying an Archive 10-22
Updating Modified Classes in a Deployed EJB Module 10-23
Creating and Managing Shared Libraries 10-24
Installing a Shared Library 10-24
Modifying an Existing Shared Library 10-25
Removing a Shared Library 10-27
Importing an Existing Shared Library 10-27
Deleting an Imported Shared Library 10-28
Stopping a Shared Library from being Inherited 10-29
Allowing a Shared Library to be Inherited 10-29
Managing Application Lifecycle 10-30
Starting Applications 10-30
Stopping Applications 10-31
Restarting Applications 10-31
Listing Applications 10-32
Restarting and Stopping OC4J Instances 10-33
Managing Data Sources 10-33
Adding, Testing, Listing, and Removing Data Source Connection Pools 10-33
Adding a Data Source Connection Pool 10-33
Testing a Data Source Connection Pool 10-34
Listing Data Source Connection Pools 10-35

Removing a Data Source Connection Pool 10-35
Adding, Testing, Listing, and Removing Data Sources 10-36
Adding a Managed Data Source 10-36
Removing a Managed Data Source 10-37
Adding a Native Data Source 10-38
Removing a Native Data Source 10-38
Testing a Database Connection 10-39
Testing a Data Source 10-40
Listing Data Sources 10-40
Getting the Data Sources Descriptor for an Application 10-41
Managing JMS Resources 10-41
Managing JMS Connection Factories 10-41
Adding a JMS Connection Factory 10-42
Removing a JMS Connection Factory 10-42
Getting Information About JMS Connection Factories 10-43
Managing JMS Destinations 10-43
Adding a JMS Destination 10-43
Removing a JMS Destination 10-44
Getting Information About JMS Destinations 10-44

11 Using the admin_client.jar Utility for Deployment

Preparing to Use admin_client.jar 11-2
Understanding the admin_client.jar Syntax and URI Specification 11-2
Deploying to a Group of OC4J Instances Within a Cluster 11-3
Deploying to a Specific OC4J Instance 11-4
Deploying to a Standalone OC4J Server 11-4
Validating a URI 11-5
Downloading and Extracting the Remote Administration Client 11-5
Printing Usage Text to the Console 11-7
Enabling Logging 11-7
Adding Web Sites 11-8
Deploying an Archive 11-8
Deploying a J2EE Application (EAR) 11-9
Deploying a J2EE Application from a Remote Client 11-11
Deploying a Standalone Web Module (WAR) 11-12
Deploying a Standalone EJB Module (JAR) 11-13
Deploying a Standalone Resource Adapter (RAR) 11-14
Using a Script File for Batch Deployment 11-15
Managing Web Bindings 11-15
Binding Web Modules to a Web Site After Deployment 11-15
Binding All Web Modules to a Single Web Site 11-15
Binding a Specific Web Module to a Web Site and Setting the Context Root 11-16
Unbinding Web Modules from a Web Site 11-17
Unbinding All Web Modules 11-17
Unbinding a Specific Web Module 11-17
Listing Web Bindings 11-18
Redeploying an Archive 11-19

Specifying a Delay Between Sequential Redeployments in a Cluster 11-20
Redeploying an Application with Scheduled Jobs 11-20
Undeploying an Archive 11-21
Undeploying an EAR, Standalone WAR, and Standalone EJB JAR 11-21
Undeploying a Standalone RAR 11-21
Updating Modified Classes in a Deployed EJB Module 11-21
Creating and Managing Shared Libraries 11-22
Installing a Shared Library 11-22
Modifying an Existing Shared Library 11-23
Viewing the Contents of a Shared Library 11-24
Listing All Shared Libraries 11-24
Removing a Shared Library 11-25
Importing an Existing Shared Library 11-25
Deleting an Imported Shared Library 11-26
Stopping a Shared Library from being Inherited 11-26
Allowing a Shared Library to be Inherited 11-26
Managing Application Lifecycle 11-27
Starting Applications 11-27
Stopping Applications 11-27
Restarting Applications 11-28
Listing Applications 11-29
Restarting and Stopping OC4J Instances 11-29
Restarting an OC4J Instance or Group of Instances 11-29
Stopping an OC4J Instance or Instances 11-30
Managing Data Sources 11-30
Adding, Testing, Listing and Removing Data Source Connection Pools 11-30
Adding a Data Source Connection Pool 11-31
Testing a Data Source Connection Pool 11-31
Listing Data Source Connection Pools 11-32
Removing a Data Source Connection Pool 11-32
Adding, Testing, Listing, and Removing Data Sources 11-32
Adding a Managed Data Source 11-33
Removing a Managed Data Source 11-33
Adding a Native Data Source 11-34
Removing a Native Data Source 11-34
Testing a Database Connection 11-35
Testing a Data Source 11-35
Listing Data Sources 11-36
Getting the Data Sources Descriptor for an Application 11-36
Managing JMS Resources 11-36
Managing JMS Connection Factories 11-37
Adding a JMS Connection Factory 11-37
Removing a JMS Connection Factory 11-37
Getting Information About JMS Connection Factories 11-38
Managing JMS Destinations 11-38
Adding a JMS Destination 11-38
Removing a JMS Destination 11-39

Getting Information About JMS Destinations 11-39
Managing OC4J Through a Remote Client 11-39
Using admin_client.jar Commands Remotely 11-39
Connecting to a Remote Oracle Application Server Instance Using JConsole 11-39
Using a JMX Programmatic Client to Manage OC4J Remotely 11-41

12 Deploying to Standalone OC4J with admin.jar

Understanding the admin.jar Syntax 12-1
Deploying or Redeploying an Application 12-2
Undeploying an Application 12-4
Updating an EJB Module Within a Deployed Application 12-4
Deploying or Redeploying a Standalone Connector 12-5
Undeploying a Standalone Connector 12-6

13 Deploying Web Applications from Eclipse

Deploying a Web Application with the Web Tools Platform 13-1
Connecting to OC4J from Eclipse 13-1
Building a Web Application 13-2
Deploying a Web Application 13-2
Using Ant Tasks from the OC4J Administration Client with Eclipse 13-2

14 Using Automatic Deployment in OC4J

Overview of Automatic Deployment in OC4J 14-1
Exploded-Directory Deployment 14-1
Redeployment of Updated Files As Needed 14-2
When to Use Automatic Deployment 14-2
Using Exploded-Directory Deployment for Application Development and Testing 14-2
Building and Deploying a J2EE Application from a Master Directory 14-2
Cloning an Application Module to an Exploded Directory 14-5
Reloading Modified Classes from an Exploded Directory 14-5
Using an Auto-Deployment Directory 14-6
Using the Check-for-Updates Feature 14-7
Enabling or Disabling Check for Updates 14-7
Redeploying Configuration Files, Deployment Descriptors, and WAR Files Automatically 14-8
Impact of Redeploying a Modified Configuration File Automatically 14-8
Impact of Redeploying a Modified Deployment Descriptor Automatically 14-9
Impact of Redeploying Modified Files in a War Automatically 14-9
Forcing a One-Time Redeployment Using admin.jar 14-10

15 Troubleshooting Deployment Errors

Interruptions During Application Deployment 15-1
Exceptions During Application Deployment 15-1
OC4J Out-of-Memory Errors 15-1
Java Compiler Out-of-Memory Errors 15-2

Stack Overflow Errors 15-2
Errors for Number of Open Files 15-2

A Third Party Licenses

ANTLR A-1
The ANTLR License A-1
Apache A-1
The Apache Software License A-2
Apache SOAP A-6
Apache SOAP License A-7
DBI Module A-10
Perl Artistic License A-10
Preamble A-10
Definitions A-10
FastCGI A-12
FastCGI Developer's Kit License A-12
Module mod_fastcgi License A-13
Info-ZIP Unzip Package A-13
The Info-ZIP Unzip Package License A-14
JSR 110 .... A-14
Jaxen A-14
The Jaxen License A-14
JGroups A-15
The GNU License A-15
mod_mm and mod_ssl A-22
OpenSSL A-23
OpenSSL License A-23
Perl A-25
Perl Kit Readme A-25
mod_perl 1.29 License A-26
mod_perl 1.99_16 License A-27
Perl Artistic License A-30
Preamble A-30
Definitions A-30
SAXPath A-32
The SAXPath License A-32
W3C DOM A-33
The W3C License A-33

Index

Preface

This book covers various aspects of deploying J2EE-compliant applications and standalone modules to Oracle Containers for J2EE 10*g* (10.1.3.5.0), or OC4J.

This preface contains the following sections:

- Intended Audience
- Documentation Accessibility
- Related Documents
- Conventions

Intended Audience

This document is intended for the following audiences:

- Professional services people who deploy applications to OC4J
- A systems administrator responsible for configuring and administering an OC4J installation
- A developer or architect involved in creating or designing a J2EE application who wants to avoid design pitfalls that could cause deployment and scalability problems

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

- General Web technology
- The J2EE environment
- General system administration

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 Configuration and Administration Guide*

  This document discusses how to configure and administer applications for OC4J, including the use of Oracle Enterprise Manager 10*g* Application Server Control, the use of standards-compliant MBeans provided with OC4J, and, where appropriate, the direct use of OC4J-specific XML configuration files.

- *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. This 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 Getting Started

This chapter provides an introduction to deploying J2EE-compliant applications and standalone modules to OC4J. It includes the following topics:

- Overview of Deployment in OC4J
- Options for Deploying Applications to OC4J

Overview of Deployment in OC4J

As a J2EE 1.4 -compliant container, OC4J provides a J2EE-compliant infrastructure for deploying, undeploying, and redeploying J2EE-compliant applications and modules.

Deployment operations can be performed 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. 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 OHS. 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.

Oracle Application Server 10*g* Release 3 (10.1.3.5.0) supports both application server clustering (with cluster topologies) and application clustering (for replication, load balancing, and transparent failover). An **application cluster** is the same set of applications hosted by two or more OC4J instances.

Oracle Application Server provides a number of deployment options with OC4J. See "Options for Deploying Applications to OC4J" on page 1-3 for an overview of these tools.

Valid Components for Deployment

These components can be deployed into an OC4J instance or to each instance in a group:

- A Web application packaged as a Web Application Archive (WAR) file
- Standalone modules packaged as Java Archive files (JARs) containing Web services, Enterprise JavaBeans modules (EJB JARs), application clients (CARs), or resource adapters (RARs)
- A complete J2EE application packaged as an Enterprise Archive (EAR) file, which can contain zero or more WARs, EJB JARs, CARs, and RARs

All J2EE-compliant archive files deployed to OC4J must be packaged in accordance with the guidelines specified in the J2EE 1.4 specification. This includes packaging the

J2EE standard deployment descriptors required for each type of component, such as the J2EE Application Descriptor (`application.xml`) for applications and the J2EE Web Descriptor (`web.xml`) for Web modules.

See the "Application Assembly and Deployment" chapter of the J*ava 2 Platform Enterprise Edition Specification, Version 1.4* for details.

Support for the J2EE Application Deployment API (JSR-88) in OC4J

The OC4J deployment infrastructure implements the functionality outlined in 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.

Specifically, the JSR-88 compliant features in OC4J provide the ability to perform these tasks:

- Start an application immediately upon deployment, making it available to clients
- Stop an application, making it unavailable to clients
- Undeploy an application or module
- Redeploy an application or module, essentially updating the currently installed application with an updated version
- Create a *deployment plan* containing the aggregated OC4J-specific configuration data needed to deploy a component to OC4J. See Chapter 8, "Working with Deployment Plans", for details on the JSR-88 implementation in OC4J.

Hot Deployment in OC4J

The term **hot deployment** refers to the process of deploying archive files - EARs, WARs, JARs, and so on - and their associated XML descriptor files on a production application server without shutting down or restarting (bouncing) the server.

In addition, libraries at the container level cannot be deployed in this manner. If an application is dependent upon a newer library, OC4J must be restarted.

See "Overview of Redeploying an Application" on page 2-5 for details on redeploying applications to an OC4J instance.

Deployment of Applications

Hot deployment or redeployment of an application or standalone module to OC4J is generally supported as long as the following conditions are true:

- No changes are made during the deployment process to existing data source, JMS, or RMI configuration files.
- The structure of an Enterprise JavaBeans module that replaces an existing EJB module has not changed.

Deployment of Shared Libraries

Shared libraries are loaded and managed at a container level. A change to a shared library does require a container restart.

If, however, a redeployed application has an `import-shared-library` declaration that goes from a lower to a higher version, such as from 2.2.8 to 2.2.9, then the version change in the declaration should not require a container restart. The application will be restarted on redeployment and should pick up the new shared library version (as long as the library is available).

Options for Deploying Applications to OC4J

You can use a number of options for deploying applications to OC4J, including utilities packaged with OC4J:

- Application Server Control User Interface
- OC4J Ant Tasks
- The admin_client.jar Command-Line Utility
- The admin.jar Command-Line Utility
- Oracle JDeveloper

Most of these options enable you to deploy an application to a specific OC4J instance or to a group of OC4J instances.

Application Server Control User Interface

Oracle Enterprise Manager 10g Application Server Control provides a Web-based user interface for completing deployment-related tasks:

- Deploy an application (EAR), a standalone Web module (WAR), a standalone EJB module (EJB JAR), or a standalone resource adapter (RAR)
- Undeploy an application, Web module, EJB module, or resource adapter
- Create, modify, or remove shared libraries for an application
- Start, restart, or stop applications
- Restart or stop an OC4J instance or group of instances
- Manage data sources and connection pools
- Manage JMS resource
- Create and edit reusable deployment plans
- Set application-specific security and application-clustering configurations

See Chapter 9, "Using Application Server Control for Deployment" for details.

OC4J Ant Tasks

OC4J includes a set of Ant tasks for performing deployment tasks on an OPMN-managed OC4J instance, a standalone OC4J server, or all OC4J instances in a group within a cluster topology. These tasks provide another option for scripting the deployment process.

Specifically, you can use Ant tasks to perform these tasks:

- Deploy an application (EAR), a standalone Web module (WAR), a standalone EJB module (EJB JAR), or a standalone resource adapter (RAR)
- Undeploy an application, Web module, EJB module, or resource adapter
- Incrementally update a deployed EJB module with modified classes
- Create, modify, or remove shared libraries for an application
- Start, restart, or stop applications
- Restart or stop an OC4J instance or group of instances
- Add, test, and remove data sources and data source connection pools

- Add and remove JMS connection pools and destinations

See Chapter 10, "Using OC4J Ant Tasks for Deployment," for an overview of the deployment-specific Ant tasks and guidelines for integrating the tasks into your application build process.

The admin_client.jar Command-Line Utility

The `admin_client.jar` command-line utility provided with OC4J can be used to perform deployment tasks on an OPMN-managed OC4J instance, a standalone OC4J server, or all OC4J instances in a group within a cluster topology. Also, the administration client distribution enables you to use `admin_client.jar` from a remote client.

Specifically, you can use the `admin_client.jar` tool to perform these tasks:

- Deploy an application (EAR), a standalone Web module (WAR), a standalone EJB module (EJB JAR), or a standalone resource adapter (RAR)
- Undeploy an application, Web module, EJB module, or resource adapter
- Incrementally update a deployed EJB module with modified classes
- Create, modify, or remove shared libraries for an application
- Start, restart, or stop applications
- Restart or stop an OC4J instance or group of instances
- Add, test, and remove data sources and data source connection pools
- Add and remove JMS connection pools and destinations

See Chapter 11, "Using the admin_client.jar Utility for Deployment," for instructions on using this tool.

The admin.jar Command-Line Utility

The `admin.jar` command-line utility provided with OC4J can be used to deploy applications to a standalone OC4J server only. It cannot be used to deploy applications to an OPMN-managed OC4J instance.

- Deploy, undeploy, and redeploy a J2EE application packaged within an EAR file
- Deploy, undeploy, and redeploy a standalone resource adapter packaged within an EAR file
- Update an EJB module within a deployed application
- Convert a data source from an earlier release to the format for the current release before deployment.

Deployment of a standalone Web module packaged in a WAR file is not supported by `admin.jar`.

See Chapter 12, "Deploying to Standalone OC4J with admin.jar" for instructions on deploying applications with this tool.

Oracle JDeveloper

Oracle JDeveloper 10*g* is a J2EE integrated development environment with end-to-end support for developing, debugging, and deploying e-business applications and Web services.

JDeveloper enables you to build J2EE applications and Web services from scratch, or jump-start the process by beginning with a J2EE framework. Whichever approach you prefer, JDeveloper offers a full suite of productivity tools to support your work from start to finish.

JDeveloper provides the ability to deploy a J2EE application into an OC4J instance directly from within the project structure. It also enables you to create a deployment plan and optionally save it as an XML file.

See the online help provided with JDeveloper for instructions on deploying applications.

2

Deploying and Undeploying J2EE Applications in OC4J

This chapter describes deploying a J2EE application packaged within an EAR file into an OC4J instance and undeploying a J2EE application from an OC4J instance. The chapter includes the following sections:

- Overview of the Application Deployment Process
- OC4J Application Deployment Process
- Overview of Redeploying an Application
- Overview of Undeploying an Application

Overview of the Application Deployment Process

OC4J provides a streamlined, user-friendly deployment process. You can deploy a J2EE-compliant EAR file as is, with no changes or repackaging required, simply by pointing to the location of the enterprise application archive (EAR) file. OC4J will automatically deploy the modules that compose the application, including any Web application archive (WAR) or Enterprise JavaBeans (EJB) modules.

The following topics outline the general steps involved in deploying an EAR to OC4J:

- Designating a Parent Application
- Binding a Web Application to a Web Site
- Creating or Applying a Deployment Plan
- Using Dynamic HTTP Server Mount Points in Oracle Application Server
- OC4J Application Deployment Process
- Impact of JDK Version on Deployed Applications
- Example of a Deployed Application Directory Structure

Note: Deploying an EAR from a read-only shared directory is not recommended, as errors might occur. Copy the EAR file to a local directory first, then deploy it.

Designating a Parent Application

Every application deployed to OC4J must have a designated parent application. The default parent is the *global application* packaged with OC4J, which is named `default`.

Designating an application as a parent enables classes and services to be shared among the child applications. A child application sees the namespace of its parent application and inherits the set of shared libraries imported by the parent. Configuration data is also imported from the parent, although it can be overridden at the child application level.

Once an application is deployed, any method within the child application can invoke any method within the parent application. This is a means to enable methods in one JAR to see EJB modules that have been deployed in another JAR. This is useful for deploying all service EJB modules in a single JAR file, for which its users declare the service application as its parent.

Binding a Web Application to a Web Site

A Web application deployed as part of a J2EE application must be bound to the Web site that will be used to access it. This binding is accomplished by specifying the *`name`* portion of the *`name`*`-web-site.xml` configuration file that defines the Web site to bind the Web application to.

In most cases, applications will be bound to the default Web site, which is defined by the `default-web-site.xml` configuration file and listens for requests on port `8888`. All Web site configuration files, including `default-web-site.xml`, are stored in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` directory.

The Web module context root, which will be appended to the URL used to access the application through a Web browser, is also set as part of this Web application enablement process. This value will typically be read from the `application.xml` deployment descriptor packaged with the application.

If Oracle HTTP Server is used, the context root value will be used in the mount point definition used to route incoming Web requests to an appropriate OC4J instance. See "Using Dynamic HTTP Server Mount Points in Oracle Application Server" on page 2-2 for details.

Creating or Applying a Deployment Plan

If you are deploying an application with Oracle Enterprise Manager 10*g* Application Server Control, you will apply a *deployment plan*, which is a client-side aggregation of all the configuration data needed to deploy an archive to OC4J, as the final step of the deployment process. You can either create a new deployment plan for the application or reuse an existing plan, which is especially useful during redeployment.

See Chapter 8, "Working with Deployment Plans" for more information on creating and using deployment plans.

Using Dynamic HTTP Server Mount Points in Oracle Application Server

In a configuration in which Oracle HTTP Server (OHS) is used, a Web request is received through an OHS instance, which then routes the request to an OC4J instance serving the requested application.

To route requests, OHS utilizes a list of application-specific *mount points*, which map the URLs supplied in requests with the OC4J instances that will service the requests.

Prior to Oracle Application Server 10*g* Release 3 (10.1.3.0.0), configuration of these application-specific mount points was completely manual. When a new application was deployed to an OC4J instance, a new mount point had to be added manually to `mod_oc4j.conf`, the configuration file for the `mod_oc4j` module within OHS that forwarded requests to OC4J instances. The OHS instance then had to be restarted.

In Oracle Application Server 10*g* Release 3 (10.1.3.5.0), mount point configuration is completely automated, eliminating the need for manual file configuration or OHS restarts. Every OC4J instance within a cluster topology sends mount point data for each of its deployed applications to OHS, which adds this information to its internal routing table.

When a new application is deployed to an OC4J instance, its mount point information is transmitted to OHS, enabling OHS to dynamically *discover* the application. Mount point information includes these items:

- The OC4J host address
- The Apache JServ Protocol (AJP) listener port

  This value is the lowest available port assigned to AJP in the `opmn.xml` file on the OC4J node.
- The Web application name

  This value is defined in the `*-web-site.xml` configuration file for the Web site the application is bound to.
- The Web context(s) defined for the application

  This value is also set in the `*-web-site.xml` configuration file.

The sending and receiving of mount point notifications is managed by Oracle Notification Server (ONS), a component of Oracle Process Manager and Notification Server (OPMN) that is installed by default with every OC4J and OHS instance in an Oracle Application Server configuration

Impact of JDK Version on Deployed Applications

When you deploy an application (including the OC4J `default` application) to OC4J running on either Java Platform, Standard Edition (J2SE) Development Kit (JDK) 6 or Java Platform 2, Standard Edition (J2SE) Development Kit (JDK) 5.0 (also known as JDK 1.5), you cannot reuse that deployment on an OC4J instance running JDK 1.4.2.

Code compiled with JDK 6 or JDK 5.0 cannot be read by the JDK 1.4 VM. If OC4J is running under JDK 1.4.2 and tries to load a class that was compiled with JDK 6 or JDK 5.0, a class-loading exception will be thrown with the following message:

```
Unsupported major.minor version 49.0
```

This exception can occur in scenarios such as the following ones:

- You deploy an application that contains EJB modules to OC4J running under JDK 5.0, and then, without undeploying the application, you restart OC4J under JDK 1.4.2. The problem is that the generated code associated with the EJB modules will have been compiled with the same JDK version that was used to start the server, and the generated code is cached between server restarts on the file system in the *`ORACLE_HOME`*`/j2ee/home/application-deployments` directory.

  The workaround for this problem is to shut down the server, remove the contents of either the *`ORACLE_HOME`*`/j2ee/home/application-deployments` directory or just the application's subdirectory, and restart the server with JDK 1.4.
- You deploy an EAR file that contains classes compiled with and targeted for JDK 5.0 to OC4J running under JDK 1.4.2.

  The workaround for this problem is to recompile the contents of the EAR using JDK 1.4.2 and redeploy it.

Note: For simplification, these scenarios were based on the assumption that no cross-compilations were being used to target code to specific JDK versions.

Example of a Deployed Application Directory Structure

The following example shows the key areas of the exploded directory structure created when an archive named `utility.ear` is deployed, assuming default settings for the target directories. The EAR includes a Web module (`utility_web.war`) and an EJB JAR (`utility_ejb.jar`) containing a single, stateful-session EJB module.

OC4J cleanly separates the standard J2EE content and the OC4J-specific files within the exploded directory structure. The original archives and the standard J2EE descriptors are copied to the `j2ee/`*`instance`*`/applications` directory, enabling these files to be used in a redeployment of the application. The OC4J-specific descriptors generated during deployment are written to the `/j2ee/`*`instance`*`/application-deployments` directory.

Also note the EJB wrapper class, `UtilityManager_ StatefulSessionBeanWrapper.class`, is generated within the `deployment-cache.jar` archive. During deployment, OC4J generates a wrapper class for each EJB module packaged within an EJB JAR, except when the EJB JAR contains only EJB 3.0 entities. The wrapper classes generated for the EJB modules within an EJB JAR are contained within an archive named `deployment-cache.jar`, which is in turn contained within a generated JAR file with the same name as the deployed EJB JAR.

```
j2ee/oc4j1/
   application-deployments/
      utility/
         orion-application.xml
         utility_web/
            orion-web.xml
         utility_ejb.jar/
            orion-ejb-jar.xml
            deployment-cache.jar/
              UtilityManager_StatefulSessionBeanWrapper.class
   applications/
      utility.ear
      utility/
         utility_web.war
         utility_ejb.jar
         META-INF/
            application.xml
         utility_web/
            index.html
            META-INF/
            WEB-INF/
               web.xml
               classes/
                  Example.class
```

OC4J Application Deployment Process

The following list describes what happens when you deploy an application packaged within an EAR file to OC4J:

1. If the application is being redeployed, the existing installation is first undeployed from OC4J.
2. OC4J copies the EAR file to the deployment directory, which defaults to the *ORACLE_HOME*/`j2ee`/*instance*/`applications` directory.
3. OC4J opens and parses the `application.xml` file packaged within the EAR file. This file is a standard J2EE descriptor that lists all of the modules contained within the EAR file. OC4J notes these modules and initializes the EAR environment.
4. OC4J reads the module deployment descriptors for each module type - Web module (WAR), EJB module, connector module, or client module - into memory. The JAR and WAR file environments are also initialized.
5. OC4J reacts to the configuration details contained in both the J2EE deployment descriptors and any OC4J-specific deployment descriptors. Also, OC4J notes any J2EE component configurations that require action, such as wrapping EJB modules with their interfaces.
6. OC4J writes out new OC4J-specific configuration files to the *ORACLE_HOME*/`j2ee`/*instance*/`application-deployments`/*app_name* directory, according to the contents of the deployment plan. If one or more OC4J-specific deployment descriptors was supplied, you may notice that OC4J added additional elements to the generated files.

   Any generated classes, such as EJB interface wrapper classes, are compiled and put into new subdirectories of this directory. For example, EJB wrapper classes are generated within an archive named `deployment-cache.jar` within the *ORACLE_HOME*/`j2ee`/*instance*/`application-deployments`/*app_name*/*jar_name*`.jar` directory, where *jar_name*`.jar` corresponds to the name of a deployed EJB JAR.
7. Finally, OC4J updates the OC4J `server.xml` configuration file with the notation that this application has been deployed.

Overview of Redeploying an Application

Redeploying a J2EE application packaged within an EAR file prompts OC4J to undeploy the previous instance of the J2EE application, including any embedded resource adapters packaged with the application. Therefore, there is no need to undeploy the application before redeploying.

If you saved the deployment plan from the previous deployment, you can reuse it during the redeployment. By default, the deployment plan will be initialized using the existing application configuration and applied to the redeployment. The previously generated OC4J descriptors will be overwritten based on the contents of the deployment plan.

Restart of OC4J After RMI or Manual Reconfiguration

After you redeploy an application, a restart of OC4J is required only in the following cases:

- A change is made to the `rmi.xml` configuration file.
- A manual edit is made to the server-level `data-sources.xml` or `jms.xml` configuration file. A restart is not required if the file is modified through Application Server Control, `admin_client.jar`, or an OC4J Ant task.

Other than in these cases, a restart of OC4J is not required after redeploying an application. For information about restarting OC4J, see the *Oracle Containers for J2EE Configuration and Administration Guide*.

The application is completely inaccessible during redeployment. Incoming requests will not be processed until the updated application is restarted by OC4J when deployment is complete.

Impact of Redeploying a Parent Application

After redeploying an application that is the parent of one or more child applications, you should restart each child application. Restarting will ensure that the child applications are able to access any inherited classes or shared libraries provided through the parent.

Overview of Undeploying an Application

An application can be removed from an OC4J instance using the following methods:

- The **Applications** section of Application Server Control
- The `undeploy` Ant task

  For instructions on using this task, see "Undeploying an Archive" on page 10-22.
- The `-undeploy` option provided through the `admin_client.jar` command-line utility

  For instructions on using this option, see "Undeploying an Archive" on page 11-21.

Results of Removing an Application from OC4J

Removing a J2EE application from an OC4J instance has the following results:

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

Impact of Undeploying a Parent Application

When an application that is the parent of one or more child applications is undeployed, the child applications are also undeployed. All of the related applications, the parent as well as its dependent applications, must be redeployed.

Results of Errors in Deployment Descriptors

If you manually edit the deployment descriptor for an application, any formatting error would cause undeployment of the application and any child applications it has. When OC4J starts up, if an application described in the *`server.xml`* file fails to load, all entries for the application and its child applications are removed from the file. After you correct the formatting error, restarting OC4J will not load the application because it has been undeployed.

For example, the following lines are added to an `orion-application.xml` file for an application, with an extra slash (/) in the first line of a `<jazn>` element:

```
<jazn provider="LDAP" jaas-mode="doAsPrivileged"/>
```

```
        <jazn-web-app auth-method="COREIDSSO"/>
</jazn>
```

When OC4J starts up, this application fails to load, and its `<application>` element is deleted from the `server.xml` file. Removing the extra slash from the <jazn> element in `orion-application.xml` and restarting OC4J does not load the application because it is no longer deployed.

Before OC4J can load an application that had an error in its deployment descriptor, you need to correct the error and restore the entry for the application, and the entries for any child applications, in *`server.xml`*. You can restore an `<application>` element by deploying the application from scratch or by manually editing the *`server.xml`* file. For more information, see "Options for Deploying Applications to OC4J" on page 1-3 or "Appendix B, Configuration Files Used in OC4J" in the *Oracle Containers for J2EE Configuration and Administration Guide*.

3

Deploying Enterprise JavaBeans Modules

The following topics discuss deployment or redeployment of Enterprise JavaBeans (EJB) modules into an application running in an OC4J instance.

- Overview of EJB Deployment
- Generation of Client-Side IIOP Stubs
- Incremental Redeployment of Updated EJB Modules
- Impact of EJB Redeployment on Application Clients

Overview of EJB Deployment

The EJB deployment process is highly automated. When an application containing one or more EJB JAR files is deployed, OC4J executes as follows:

1. OC4J generates a wrapper class for each of the home interfaces (`EJBHome` and `EJBLocalHome` implementations) and component interfaces (`EJBObject` and `EJBLocalObject` implementations) packaged within each EJB JAR file.
2. OC4J invokes the Java compiler that it is configured to use to compile the generated EJB wrapper classes. The compiled classes are output to an archive named `deployment-cache.jar` in a new subdirectory with the same name as the deployed EJB JAR in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/`*`app_name`*`/application-deployments/`.

   For example, suppose you deployed `mystore.ear`, which contains `inventory-ejb.jar`. The compiled wrapper classes will be generated in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/mystore/application-deployments/inventory-ejb/deployment-cache.jar`.
3. OC4J optionally generates client-side IIOP stubs for each home and component interface if configured to do so.

See "Example of a Deployed Application Directory Structure" on page 2-4 for an example of the directory structure created for an application by OC4J.

Because of the amount of processing performed by OC4J and the Java compiler, deploying an EJB JAR file containing a large number of EJB modules (approximately 100) can significantly increase the amount of time required to deploy an application. See Chapter 4, "Deploying Large Applications," for guidelines on tuning OC4J and the Java compiler for large-application deployment.

Note: If you are using Application Server Control, EJB 3.0 entities deployed with session beans are not visible in the Application Server Control view of the EJB JAR module. After you deploy EJB 3.0 entities to OC4J, you cannot manage them through Application Server Control. If you use Application Server Control to view an EJB module, the Entity Beans area will display the following message:

```
No entity beans found
```

You can manage all other EJB 3.0 beans, such as session beans. If you deploy an EJB module that contains both EJB 3.0 session beans and EJB 3.0 entities, your session beans will be visible though Application Server Control, but the entities will not be visible.

For more information about managing EJB modules, see the *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide*.

Generation of Client-Side IIOP Stubs

OC4J optionally generates client-side IIOP stubs for each home and component interface if configured to do so when you deploy with Application Server Control, `admin_client.jar`, or the OC4J Ant tasks.

- Generating Stubs with Application Server Control
- Generating Stubs with admin_client.jar
- Generating Stubs with the OC4J Ant Tasks

Generating Stubs with Application Server Control

See "Deploying or Redeploying an Application" on page 9-4 for instructions on deploying applications with Application Server Control.

1. Before deploying the EJB modules, configure OC4J to generate client-side IIOP stubs with Application Server Control.
   a. Click the **Administration** tab for an OC4J instance.
   b. Under **Properties**, select **EJB Compiler Settings**.
   c. Under Compile Time Parameters, select **Generate IIOP Client Stubs when Compiling EJBs** .
2. Next, enable stub generation at deployment time:
   a. In the third panel (Deployment Settings) of the deployment wizard, click **Edit Deployment Plan**.
   b. Set **enableIIOP** to **true**.

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/*app_name*` 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/*app_name*/*ejbModuleName*/` directory.

Generating Stubs with admin_client.jar

The `-deploy` command on the `admin_client.jar` command line provides two options for generating IIOP stubs: one for generating stubs on the server, the other for generating stubs on the server and copying the new stubs to another location.

1. Before deploying the EJB modules, set the `-DGenerateIIOP` system property, which configures OC4J to generate client-side IIOP stubs at startup.
   - In standalone OC4J, specify this system property on the OC4J command line:

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

   - In an OPMN-managed OC4J instance, set the property in `opmn.xml`:

     ```
     <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="-DGenerateIIOP=true
              -Dhttp.session.debug=true"/>
           </category>
           ...
         </module-data>
       </process-type>
     </ias-component>
     ```

2. Next, deploy the application with the `admin_client.jar -deploy` command. See "Deploying an Archive" on page 11-8 for details on using this command.
   - Include `-enableIIOP` to generate IIOP client stubs on the OC4J server.

     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.
   - Include `-iiopClientJar` *`path`* to generate stubs in the same locations specified in the preceding item, as well as to the path specified with the parameter.
   - Include `-removeArchive` to delete the deployment archive from the server's file system after deployment.

Generating Stubs with the OC4J Ant Tasks

The `deploy` Ant task also provides two options for generating IIOP stubs: one for generating stubs on the server, the other for generating stubs on the server and copying the new stubs to another location.

1. Before deploying the EJB modules, set the `-DGenerateIIOP` system property, which configures OC4J to generate client-side IIOP stubs at startup. See the preceding section, "Generating Stubs with admin_client.jar," for details.
2. Next, deploy the application with the `deploy` task. See "Deploying a J2EE Application (EAR)" on page 10-11 for details on using this task.
   - Include `enableIIOP` to generate IIOP client stubs on the OC4J server.

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.

- Include `iiopClientJarPath` to generate stubs in the same locations specified in the preceding item, as well as to the path specified with the parameter.

Incremental Redeployment of Updated EJB Modules

OC4J supports incremental or partial redeployment of EJB modules that are part of a deployed application. This feature makes it possible to deploy only those beans within an EJB JAR that have changed, without requiring the entire module to be redeployed. Previously deployed beans that have not been changed will continue to be used.

This functionality represents a significant enhancement over previous releases of OC4J, which treated an EJB module as a single unit, requiring that the module first be undeployed, then redeployed with any updates.

A restart of OC4J is required only if changes are made to the EJB configuration data during the redeployment process. If no changes are made, a “hot deployment” can be performed without restarting OC4J.

The incremental redeployment operation will automatically stop the application containing the EJB module or modules to be updated and then automatically restart the application when finished.

Note: During redeployment, any client connections to an EJB module being updated will be lost. Oracle strongly recommends that you stop the application before redeploying the EJB module. All existing requests will be allowed to complete, but no new requests will be allowed until the application is restarted.

For CMP or BMP entity beans, OC4J uses code generation to generate the server implementation of the EJB interfaces (wrappers). In this case, incrementally redeploying only changed beans is most likely to be more efficient than redeploying the entire application.

For session beans, message-driven beans, and EJB 3.0 JPA entities, OC4J uses byte code generation to generate wrappers. Because this approach reduces deployment time so much, it might be just as efficient to redeploy the entire application as to redeploy only changed beans. In this case, incremental redeployment is optional.

The general procedure for using incremental deployment follows:

1. Deploy an application with a large number of enterprise beans.
2. Change a bean-related class file in an EJB module and rebuild the EJB JAR file (for example, `myBeans-ejb.jar`).
3. Submit the updated EJB JAR to OC4J with any of the following tools:
   - JDeveloper
   - The `updateEJBModule` Ant task

See "Updating Modified Classes in a Deployed EJB Module" on page 10-23.

- The `-updateEBJModule` command of the `admin_client.jar` or `admin.jar` command-line utility

  See "Updating Modified Classes in a Deployed EJB Module" on page 11-21 for information about using `admin_client.jar` or "Updating an EJB Module Within a Deployed Application" on page 12-4 for information about using `admin.jar`.

The following example shows how to use `admin_client.jar` for incremental redeployment:

```
java -jar admin_client.jar deployer:oc4j:rmis://localhost:23791 admin welcome
-updateEJBModule -appName petstore -ejbModuleName myBeans-ejb.jar
-file build/myBeans-ejb.jar
```

4. Repeat steps 2 and 3.

For more information, see the *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide*.

Impact of EJB Redeployment on Application Clients

The impact of EJB redeployment on existing clients differs depending on the type of EJB modules used in the application. This section includes the following topics:

- Impact of Redeploying Session Beans
- Impact of Redeploying Entity Beans

Impact of Redeploying Session Beans

The following describe issues related to redeploying session beans.

Stateless Session Beans

For applications that include stateless-session EJB modules, the redeployment appears seamless to users, with no interruption in service. Existing requests will be served by current bean instances, while new requests will be served with new instances.

Stateful Session Beans

For an application or Web service that utilizes active stateful-session EJB instances, you must explicitly specify a "persistence directory" where client state data will be persisted through serialization during the undeployment and redeployment processes. If this directory is not specified, all serialized state data will be lost during undeployment.

The persistence directory is defined in the `<persistence>` element within the `orion-application.xml` configuration file. This directory can also be set as the value of the `persistencePath` property through the deployment plan editor at the time the EJB archive is deployed. See "Setting Web Module Configuration Properties" on page 8-11 for information.

Also note that for existing clients, a serialization-related exception will occur if there are any changes in the structure of the session beans deployed with the module.

Impact of Redeploying Entity Beans

The following sections describe issues related to redeploying entity beans.

Bean-Managed Persistence Beans

The application developer is responsible for handling any exceptions that might occur as a result of hot deployment of EJB modules.

Container-Managed Persistence Beans

An unknown number of side effects might occur if the structure, types, and relationships of the container-managed fields within an EJB module are changed. For this reason, OC4J should always be restarted after you make any changes.

4

Deploying Large Applications

This chapter provides guidelines for configuring OC4J and the Java compiler for deployment of large J2EE applications that contain large numbers of Enterprise JavaBeans (EJB) modules (approximately 100 or more).

This chapter includes the following sections:

- Specifying the Compilation Mode to Use
- Configuring the Java Compiler
- Tuning the OC4J JVM for Large Deployments

Specifying the Compilation Mode to Use

When you deploy an EJB 3.0 application with one or more annotations, OC4J will automatically write its in-memory ejb-jar.xml file to the same location as the `orion-ejb-jar.xml` file in the deployment directory. This `ejb-jar.xml` file represents configuration obtained from both annotations and a deployed `ejb-jar.xml` file (if present).

When an application containing one or more EJB 2.1 JAR files is deployed, OC4J automatically generates a wrapper class for each EJB module that implements the various component interfaces packaged with the application. OC4J then invokes the Java compiler to compile these generated EJB wrapper classes.

OC4J supports two modes for compiling EJB wrapper classes: batch mode and nonbatch mode.

Batch Mode

This is the default compilation mode used by OC4J. In general, batch mode provides faster time to deployment when you are deploying large, EJB-heavy applications. However, it also requires a greater heap memory allocation than the nonbatch mode of deployment.

In this mode, OC4J makes a single call to the Java compiler to compile all of the Java wrapper code for all of the EJB modules within the EAR being deployed.

If the compiler is configured to run in out-of-process mode, which it is by default in OC4J, then OC4J will create a single JVM process to execute compilation of the wrapper code. See "Configuring Out-of-Process or In-Process Compiler Execution" on page 4-3 for instructions on configuring the compiler to run out-of-process or in the same JVM process as OC4J.

Nonbatch Mode

If you find that OC4J throws `java.lang.OutOfMemory` exceptions while compiling in batch mode, you might want to compile in nonbatch mode instead because it requires less memory allocation. However, using this mode will result in a longer time to deployment.

In this mode, OC4J makes multiple calls to the compiler, one for each EJB JAR file within the EAR being deployed.

If the Java compiler is configured to run in out-of-process mode, OC4J will create a JVM process for each EJB JAR file included in the EAR file being deployed.

To deploy in nonbatch mode, set the `batch-compile` attribute of the `<orion-application>` element in `application.xml` or `orion-application.xml` to `false`. For example:

```
<orion-application ... batch-compile="false" .../>
```

Wrapper Code Debugging

By default, when OC4J deploys an EJB 2.1 CMP application, it generates wrapper code in the `*ORACLE_HOME*/j2ee/*instance*/application-deployments/*ear-name*/*ejb-name*/generated` directory, compiles the code, creates a JAR file that contains the compiled classes, and then deletes the wrapper code it generates. You can configure OC4J to preserve the wrapper code that it generates. Examining the wrapper code can aid in debugging some application problems.

Notes: The `ejbdeploy.batch` system property, which had been used to specify batch versus nonbatch compilation mode, is deprecated in OC4J 10*g* (10.1.3.5.0).

Debugging generated wrapper code is also deprecated in this release.

These options apply only to EJB 2.1 entity beans with container-managed persistence; they do not apply to session beans, message-driven beans, or EJB 3.0 entities. OC4J generates only one file for each EJB 2.1 entity bean with container-managed persistence. OC4J does not generate any artifacts if you use only EJB 3.0 entities.

Configuring the Java Compiler

This section provides guidelines on configuring the Java compiler that will compile the EJB wrapper classes and JSPs during deployment. It includes the following topics:

- Specifying an Alternative Java Compiler
- Configuring Out-of-Process or In-Process Compiler Execution
- Summary of Java Compiler Configuration Parameters

Note that all compiler configuration parameters are specified as attributes of the `<java-compiler>` element in `*ORACLE_HOME*/j2ee/*instance*/config/server.xml`, the OC4J server configuration file.

Specifying an Alternative Java Compiler

By default, OC4J uses the `javac` compiler packaged with the Sun Microsystems JDK to compile generated EJB wrapper classes, JavaServer Pages, Web services classes, and any `.java` files packaged with an application. However, you can configure OC4J to use a different Java compiler by modifying the `<java-compiler>` element in `server.xml` with the alternative compiler configuration.

For example, the following notation will cause OC4J to use the Jikes compiler from IBM:

```
<java-compiler name="jikes" in-process="false"/>
```

Configuring Out-of-Process or In-Process Compiler Execution

The Java compiler can be configured within OC4J to execute in one of two modes: out-of-process or in-process.

Out-of-Process

In this mode, a separate JVM process is spawned for the compiler to execute within. This is the default compiler execution mode used by OC4J because it offers better management of memory resources. Once compilation of the EJB wrapper classes is complete, the memory allocated to the associated JVM will be released and made available to other processes.

Set the `in-process` attribute of the `<java-compiler>` element to `false` to execute the compiler in this mode.

In-Process

In this mode, the compiler executes within the same JVM process as OC4J. Set the `in-process` attribute of the `<java-compiler>` element to `true` to execute in this mode.

Summary of Java Compiler Configuration Parameters

Table 4–1 summarizes the attributes of the `<java-compiler>` element, which defines the Java compiler configuration in `server.xml`, the OC4J configuration file.

Table 4–1 Attributes of the <java-compiler> Element

Attribute	Description
`name`	Value: string Default: javac Specifies the compiler name. Set to `javac`, the default value, to use the Sun javac compiler.
`bindir`	Value: string Specifies the absolute path to the compiler directory. This attribute does not need to be specified to use the default `javac` compiler.
`in-process`	Value: Boolean Default: `false` Indicates whether to execute the Java compiler in-process (`true`) or out-of-process (`false`).
`options`	Used to pass command line options to the Java compiler. Separate multiple options with a space. For example: `<java-compiler name="javac" options="-J-Xmx2048m -J-Xss=8m"/>`
`encoding`	Value: string Default: `ISO-8859-1` Specifies the source file encoding to use.

Table 4–1 (Cont.) Attributes of the <java-compiler> Element

Attribute	Description
`extdirs`	Indicates the compiler extension library location.
`debug`	Value: Boolean Default: `false` Set to `true` to generate compilation-time debugging output.

Tuning the OC4J JVM for Large Deployments

In addition to setting the OC4J JVM heap size (young and old generations), you should also pay attention to the permanent generation size when deploying large applications.

The JVM permanent generation plays an important role when you are deploying large applications. Because a large application can contain hundreds or even thousands of EJB modules, the OC4J JVM needs to load a large number of classes, which requires a higher value for permanent generation size.

If the permanent generation size is set too low, you may see `java.lang.OutOfMemoryError` errors from the JDK even if you have plenty of free memory in the heap. If this occurs, you can increase the permanent generation (Perm) size by setting the `-XX:MaxPermSize` property.

For example, to set the total heap size to 768 MB, set the following at OC4J startup:

```
java -Xms512m -Xmx512m -XX:MaxPermSize=256m -jar oc4j.jar
```

To determine the value for the `-XX:MaxPermSize` property - including young and old generation sizes - you can use `visualgc` to monitor the OC4J JVM during application deployment. The `visualgc` garbage collection monitoring tool is included with the free `jvmstat 3.0` distribution available from Sun Microsystems. See the following Web site for more information and to download the tool:

`http://java.sun.com/performance/jvmstat/`

5 Deploying Web Modules

This chapter discusses deploying and redeploying Web modules into an OC4J instance. It includes the following topics:

- Deploying a Standalone Web Module
- Redeploying a Standalone Web Application
- Redeploying an Updated Web Module into an Existing Application
- Adding or Modifying JavaServer Pages in an Active Web Module

Deploying a Standalone Web Module

A standalone Web module packaged as a WAR file can be deployed to an OC4J instance. However, the Web module must be designated a child of either the `default` application (if a standalone Web application) or another deployed application that does not already contain a Web module component.

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 an `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.

OC4J wraps the standalone WAR file within a generated EAR file, then deploys the EAR to the default deployment directory, *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications`. The EAR includes a generated `application.xml` deployment descriptor, which includes the context root appended to the URL used to access the Web module.

A restart of OC4J is not required after deployment of a standalone Web module.

Redeploying a Standalone Web Application

In this context, a standalone Web application is a WAR file that has the `default` application specified as its parent. When a standalone Web application is redeployed, OC4J performs the following:

- Removes the Web application from its execution space
- Removes the class loader that was associated with execution of the Web application
- Reparses the application-specific `web.xml` and `orion-web.xml` descriptors to pick up any changes
- Reinitializes servlet listeners, filters, and mappings

Existing sessions are either purged or serialized out to a file in the persistence directory specified for the Web application, which is defined in the `persistence-path` attribute of the `<orion-web-app>` element. When the newly deployed application is started, it looks in the persistence directory for the file containing the serialized sessions.

Notes:

- If you update a servlet `.class` file under `/WEB-INF/classes`, then upon the next request, the servlet and its dependency classes are reloaded and the Web application is automatically redeployed if automatic deployment (OC4J polling) is enabled.

  For information about OC4J polling, see Chapter 14, "Using Automatic Deployment in OC4J".
- Redeployment does not significantly affect OC4J-specific descriptors such as `orion-application.xml` and `orion-web.xml` in the server deployment directory. After you trigger reloading, the previously copied or generated files will keep any previously specified nondefault settings.

Redeploying an Updated Web Module into an Existing Application

An updated Web module packaged in a WAR cannot be redeployed into a J2EE application running on OC4J. Instead, the WAR must be repackaged within the application's EAR file, and the entire EAR must then be redeployed.

Adding or Modifying JavaServer Pages in an Active Web Module

In OC4J, you can add new JavaServer Pages (JSPs) to an actively running Web module and modify existing JSPs without an application redeployment or restart.

To use this feature, simply drop a new or updated JSP into the appropriate directory within the exploded WAR file structure in the OC4J instance, which is `*ORACLE_HOME*/j2ee/*instance*/applications/*appName*/*webModuleName*/`. OC4J will translate the page and load (or reload) it into the runtime.

This feature is enabled by default, and you can manage it with the **When a JSP Changes** JSP configuration parameter. You can set this parameter with Application Server Control, as follows:

1. Click the **Administration** tab for an OC4J instance.
2. Select **JSP Properties**.
3. Set the **When a JSP Changes** parameter to one of the following values:
   - **Recompile JSP**
     This is the default setting. OC4J will check the timestamp of the JSP page, retranslate it, and reload if it has been modified since it was last loaded. The functionality in the following description for **Reload Classes** will also be executed.
   - **Reload Classes**
     OC4J will check the timestamp of classes generated by the JSP translator, such as page implementation classes, and reload any that have changed or been redeployed since they were last loaded.

This might be useful, for example, when you deploy or redeploy compiled classes, but not JSP pages, from a development environment to a production environment.

- **Do Nothing**
  Set this parameter to disable the auto-reload feature. New or updated JSPs will not be automatically loaded into the OC4J runtime, and the container will not perform any timestamp, any retranslating of JSP pages, or any reloading of generated Java classes. This is the most efficient mode for a production environment, in which JSP pages are not expected to change frequently.

4. Restart the OC4J instance for your changes to take effect.

This scenario assumes that any dependent classes already exist within the deployed Web module, and that the JSP updates do not require any changes to the `web.xml` or `orion-web.xml` configuration file. If either of these conditions is false, the Web module must be repackaged as a WAR file and redeployed with the full application within an EAR file.

Setting JSP Configuration Parameters in the XML Configuration File

In a standalone OC4J environment, you can configure adding new JSPs to a running Web module by setting the `main_mode` `<init-param>` element directly in the `global-web-application.xml` file, which is the configuration file for the OC4J servlet and JSP containers. The values set through Application Server Control are persisted to this file.

Valid settings for `main_mode` are:

- `reload`: Maps to the **Reload Classes** parameter set with Application Server Control.
- `recompile` (default): Maps to the **Recompile JSP** parameter.
- `justrun`: Maps to the **Do Nothing** parameter.

The following example illustrates how to set the `main_mode <init-param>` element to `recompile`, which forces OC4J to check the timestamp of the JSP page, retranslate it, and reload it if it has been modified since it was last loaded.

```
<servlet>
   <servlet-name>jsp</servlet-name>
   <servlet-class>oracle.jsp.runtimev2.JspServlet</servlet-class>
   <init-param>
      <param-name>precompile_check</param-name>
      <param-value>true</param-value>
   </init-param>
   <init-param>
      <param-name>main_mode</param-name>
      <param-value>recompile</param-value>
   </init-param>
   <init-param>
      <param-name>javaccmd</param-name>
      <param-value>javac -verbose</param-value>
   </init-param>
</servlet>
```

Note: The `javaccmd` parameter is deprecated.

6 Deploying Resource Adapters

A resource adapter can be packaged and deployed as a standalone RAR that is available as a shared library to all applications within an OC4J instance.

This chapter includes the following topics:

- Deploying a Standalone RAR
- Redeploying or Undeploying a Standalone RAR

For additional information about deploying resource adapters, see the *Oracle Containers for J2EE Resource Adapter Administrator's Guide*.

Deploying a Standalone RAR

A resource adapter deployed as a standalone RAR is deployed as a child of the `default` application, making the connector available to all other applications deployed to the OC4J instance. When an application is deployed, the application imports all standalone resource adapters that were previously deployed, by default.

Each standalone resource adapter deployed in OC4J is represented as a shared library, which, by default, is available to all applications. All code sources of a standalone resource adapter are added to a dedicated, shared loader that will be imported by all applications unless the applications are explicitly configured otherwise. When multiple versions of a standalone resource adapter are deployed, an application can be configured to import a specific resource adapter, so all resource adapter classes will be loaded from the same adapter.

A resource adapter is deployed to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/connectors` directory by default. This directory is specified in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config/server.xml`, the OC4J server configuration file.

When a resource adapter is deployed, the following updates are made to the OC4J instance:

- A new <connector> element defining the resource adapter is added to *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config/oc4j-connectors.xml`. This file provides an enumeration of the standalone resource adapters deployed to the OC4J instance.
- An `oc4j-ra.xml` descriptor is generated in a new directory with the same name as the connector in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application-deployments` directory.

For deploying a standalone resource adapter (RAR) to a specific OC4J instance or to all OC4J instances in a group within a cluster, you can use one of these tools:

- Application Server Control

See "Using Application Server Control for Deployment" on page 9-1.

- `deploy` Ant task

  See "Deploying a Standalone Resource Adapter (RAR)" on page 10-15.

- `admin_client.jar` command-line utility

  See "Deploying a Standalone Resource Adapter (RAR)" on page 11-14.

Although `admin_client.jar` is the preferred command-line utility, you can use `admin.jar` instead to deploy a RAR to a standalone OC4J server. For details on deploying a resource adapter with `admin.jar`, see "Deploying or Redeploying a Standalone Connector" on page 6-2.

Deploying a Resource Adapter with Dependencies

When a standalone resource adapter is deployed, all running applications that were previously deployed, except the `default` application, are asked to import the resource adapter. So, an application that is dependent on a standalone RAR can be deployed before the resource adapter as long as the application does not attempt to use the resource adapter prior to its deployment.

Take special care when you deploy a standalone resource adapter after a dependent application because the application might have already loaded classes, which could include resource adapter classes. If importing the standalone RAR causes previously loaded classes to be subsequently loaded by a different loader, unexpected exceptions may occur.

A resource adapter can look up and use another resource adapter. Because each standalone RAR has its own class loader, standalone RARs import other deployed standalone RARs as shared libraries. By default, a standalone RAR will import all previously deployed standalone RARs and all shared libraries. A standalone RAR must be deployed before any dependent standalone RARs.

Deploying a resource adapter to the default application will prevent the resource adapter from using any standalone RAR that is deployed as a shared library. Resources deployed as shared libraries are not imported by the `default` application.

Deploying Multiple Versions of a Standalone RAR

You can deploy multiple standalone resource adapters that contain classes of the same name. Each resource adapter must have a unique name and not a version number.

Because all standalone RARs are available to all applications by default, any application that uses a standalone RAR for which multiple *versions* are deployed must explicitly specify which of the versions it will use. The application must use only one version of a resource adapter that has multiple deployed versions. An application specifies which standalone RARs it will use in the configuration file `orion-application.xml`.

For more information about using multiple versions of a resource adapter, see the *Oracle Containers for J2EE Resource Adapter Administrator's Guide*.

Redeploying or Undeploying a Standalone RAR

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

If you undeploy or redeploy a resource adapter with active endpoints without stopping it first, OC4J throws a DeployerException exception due to the active endpoints. Stop the resource adapter before redeploying or undeploying it.

When stopping a resource adapter, OC4J does not always stop dependent applications. Stop any applications that use a resource adapter before you stop it, to make sure all application activity completes.

7 Deploying Web Services

This chapter discusses deployment and redeployment of Web services. It includes the following topics:

- Deploying a Web Service
- Redeploying a Web Service

Deploying a Web Service

A Web service can be packaged as a WAR or as an EJB JAR containing stateless session beans for deployment to OC4J.

If an archive containing a Web service does not include a Web Services Description Language (WSDL) document, OC4J will generate a WSDL document at deployment time.

See Chapter 5, "Deploying Web Modules" for a discussion on deploying and redeploying WAR files to OC4J.

See Chapter 3, "Deploying Enterprise JavaBeans Modules" for guidelines on deploying and redeploying EJB archives.

The deployment plan editor provided with Oracle Enterprise Manager 10*g* Application Server Control enables you to set values in the OC4J-specific Web services deployment descriptor, `oracle-webservices.xml`, at deployment time. For more information, see "Setting Web Services Configuration Properties" on page 8-24.

Redeploying a Web Service

In general, the guidelines for updating a Web service in OC4J are the same as those for any WAR or EJB JAR containing stateless session beans. However, redeployment of a Web service is essentially required when:

- Changes are made to the existing reliability configuration
- Changes are made to existing security policies or data set in XML configuration files

Redeployment is required so that an updated WSDL document containing the updated reliability and/or security notations can be supplied. Without the updated WSDL document, clients calling the Web service will not work correctly.

Ideally, a new WSDL document containing the updated reliability and/or security notations should be supplied with the WAR or EJB JAR during redeployment. If no WSDL document is supplied, OC4J will generate a new document with the correct notations. However, data set in the previously deployed WSDL document will be lost.

8 Working with Deployment Plans

This chapter provides instructions on creating and using deployment plans, which facilitate the process of editing and reusing configuration data when you are deploying archives to Oracle Containers for J2EE (OC4J). It includes the following sections:

- Deployment Plan Overview
- Creating or Editing a Deployment Plan
- Setting Properties in a Deployment Plan

Deployment Plan Overview

Like other J2EE containers, OC4J utilizes a number of vendor-specific deployment descriptor files that extend the standard J2EE deployment descriptors. For example, the OC4J-specific `orion-application.xml` descriptor extends the J2EE standard `application.xml` descriptor with configuration data specific to OC4J. See "Overview of J2EE and OC4J Deployment Descriptors" on page 8-2 for an overview of the relationships between J2EE and OC4J deployment descriptors.

A key feature of JSR-88 is the ability to create a **deployment plan**: a client-side aggregation of all the configuration data needed to deploy an archive to OC4J. This deployment plan can be edited at the time of deployment using the *deployment plan editor* functionality provided through Application Server Control, providing a straightforward way to modify configuration data for a particular installation. (See "Setting Properties in a Deployment Plan" on page 8-4 for details.)

When the archive is deployed, both the archive and the deployment plan are sent to the OC4J server. OC4J uses the contents of the deployment plan to generate the various OC4J-specific descriptors within the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application-deployments` directory.

For example, if an EAR containing a WAR and an EJB JAR is deployed, the deployment plan will contain the aggregated configuration data for each of these archives. Upon deployment, this data would be written to the `orion-application.xml`, `orion-web.xml`, and `orion-ejb-jar.xml` descriptors, respectively, generated by OC4J.

Once created, a deployment plan can be saved as a file. It can then be reused for redeploying the component or for deploying other components. If an existing deployment plan is not applied to a component at the time of deployment, a new plan is created by default.

Note: Deployment plans include additional properties that do not appear in a deployment descriptor but are required by the deployment process, such as a property specifying whether the EAR should be deleted after deployment has completed.

How Deployment Plans Interact with Packaged Deployment Descriptors

If one or more OC4J-specific descriptors, such as `orion-application.xml` and `orion-ejb-jar.xml`, are packaged within an archive being deployed, the deployment plan is initialized with the data within these files. The configuration data can then be edited through the deployment plan editor in Application Server Control before deployment. (See "Setting Properties in a Deployment Plan" on page 8-4 for details.)

Changes made through the deployment plan editor are not written back to the archive. For example, suppose that the deployment plan editor is used to define a `UserManager` class to use at the application level. When the application is deployed, the `orion-application.xml` file generated within OC4J will contain the added `<user-manager>` element. The `orion-application.xml` file packaged within the archive will not.

Overview of J2EE and OC4J Deployment Descriptors

Deployment descriptors are configuration files that are deployed with J2EE applications and modules. Each J2EE standard deployment descriptor is extended by a corresponding OC4J-specific descriptor. The following table provides a description of these files and illustrates how they relate to one another.

The XML Schema Definition (XSD) file that describes each OC4J-specific descriptor is also noted. You can view the current Oracle XSDs at the following link:

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

Table 8–1 J2EE and OC4J Deployment Descriptors

J2EE Standard Descriptors	OC4J Proprietary Descriptors
`application.xml`	`orion-application.xml`
Specifies the components of a J2EE application, such as Enterprise JavaBeans (EJB) and Web modules, and can specify additional configuration for the application as well. This descriptor must be included in the `/META-INF` directory of the application's EAR file.	Generally defines OC4J-specific configurations, such as security role mappings, data source definitions, JNDI namespace access, and shared library replacements. Can also be used to specify additional modules, beyond those specified in the J2EE `application.xml` descriptor. The format of this file is defined by `orion-application-10_0.xsd`.
`web.xml`	`orion-web.xml`
Specifies and configures a set of J2EE Web components, including static pages, servlets, and JSP pages. It also specifies and configures other components, such as EJB modules, that the Web components might call. The Web components might together form an independent Web application and be deployed in a standalone WAR file.	Extends the standard J2EE descriptor with application-level OC4J-specific configuration data, such as whether or not OC4J features like developer mode or auto-reload of JSPs is enabled. The format of this file is defined by `orion-web-10_0.xsd`.

Table 8–1 (Cont.) J2EE and OC4J Deployment Descriptors

J2EE Standard Descriptors	OC4J Proprietary Descriptors
`ejb-jar.xml` 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.	`orion-ejb-jar.xml` Defines OC4J-specific configuration data for all EJB modules within an archive, including EJB pool settings, time-out and retry settings, JNDI mappings, and finder method specifications. Also includes properties for the TopLink persistence manager. The format of this file is defined in `orion-ejb-jar-10_0.xsd`.
`application-client.xml` 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. The file format is defined in `orion-application-client-10_0.xsd`.

`ra.xml`
Contains information on implementation code, configuration properties, and security settings for a resource adapter packaged within a RAR file.
	`oc4j-ra.xml`
	Contains deployment configuration data for a single resource adapter. This data includes such information as the JNDI name to be used, EIS connection information, connection pooling parameters, and resource principal mappings.
	The file format is defined by `oc4j-connector-factories-10_0.xsd`.
	`oc4j-connectors.xml`
	In an OC4J instance with standalone resource adapters deployed, contains an enumeration of those standalone resource adapters. In a J2EE application with embedded resource adapters deployed, contains a list of embedded resource adapters that have been bundled with the application.
	This file is formatted according to `oc4j-connectors-10_0.xsd`.

`webservices.xml` Describes a Web service, including WSDL information and JAX-RPC mapping data, for a Web service application packaged within a WAR file.	`oracle-webservices.xml` Defines properties used by the OC4J Web services container, such as whether to expose the WSDL file. It also defines endpoint addresses and data specific to EJB modules implemented as Web services. The file can be packaged in either a WAR or an EJB JAR containing a Web service. This file is formatted according to `oracle-webservices-10_0.xsd`.

Creating or Editing a Deployment Plan

Deployment plans can be created or edited through the deployment plan editor functionality available through the Web-based Oracle Enterprise Manager 10*g* Application Server Control interface and the J2EE and Studio Editions of the Oracle JDeveloper 10*g* integrated development environment.

- Creating or Editing Deployment Plans with Application Server Control
- Creating or Editing Deployment Plans with Oracle JDeveloper

Creating or Editing Deployment Plans with Application Server Control

You can access the deployment plan editor provided with Application Server Control through the third and final page of the deployment wizard. Click the **Edit Deployment Plan** button on this page to view the editor.

Each archive being deployed is displayed as an XPath node in the left-hand navigation pane of the editor. Select an archive node to access the deployment plan properties for that archive type. You will be able to view the following:

- The J2EE standard descriptor packaged with the archive
- The current values for XML elements and attributes that will be set in the OC4J-specific descriptor generated at deployment time
- A set of properties that can be set in the deployment plan, each corresponding to an element or attribute in the OC4J-specific descriptor that will be generated at deployment time

  See "Setting Properties in a Deployment Plan" on page 8-4 for a description of each property that can be set. Not all parameters that can be set in an XML descriptor file can be set through the deployment plan editor; parameters set after deployment are not exposed.

After creating a deployment plan, you can save it as an XML file, which you can apply to other application deployments.

You can also select an existing deployment plan to apply during a deployment in the first panel of the "deployment wizard". Once retrieved, a deployment plan can be edited or used as-is.

Creating or Editing Deployment Plans with Oracle JDeveloper

The deployment plan editor functionality provided with Oracle JDeveloper 10*g* is presented as the **Configure Application** panel.

To use the deployment plan editor, you must first create a connection of type **J2EE 1.4 Server** to the target OC4J instance. This is done through the wizard accessible via the Connection Navigator panel.

Once you are connected, the Configure Application panel is displayed after you select an archive and choose **Deploy to ->** ***connection_name***.

After creating a deployment plan, you can save it as an XML file, which you can apply to other application deployments.

See the "Deploying Applications" topic in the Oracle JDeveloper online help for instructions on deploying archives to OC4J.

Setting Properties in a Deployment Plan

This section describes the properties that you can set in a deployment plan and how to set them. It is organized based on the type of archive being deployed:

- Setting J2EE Application Configuration Properties
- Setting Web Module Configuration Properties
- Setting Enterprise JavaBeans Module Configuration Properties
- Setting Web Services Configuration Properties
- Setting Application Client Configuration Properties

- Setting Resource Adapter Properties

Setting J2EE Application Configuration Properties

You can set the following OC4J-specific properties when you are deploying a J2EE application packaged in an EAR file. Each property maps to an element attribute in the `orion-application.xml` descriptor.

applicationId

Contains a server-defined string that specifies the application's unique identifier.

autoCreateTables

Set to `true` to automatically create database tables for CMP Enterprise JavaBeans (EJB) modules in this application. The default is `false`.

autoDeleteTables

Specifies whether old database tables for CMP beans should automatically be deleted when this application is redeployed. The default is `false`.

cluster

Configures OC4J application clustering at the application level. Application clustering is typically configured at the global level; however, application-level settings will override the global configuration. See *Oracle Containers for J2EE Configuration and Administration Guide* for a detailed overview of the OC4J clustering framework.

- `enabled`: Specifies 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. Clustering can be enabled or disabled for a specific application.
- `groupName`: 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 application group name.
- `allowColocation`: Specifies whether to allow the application state to be replicated to an application group member residing on the same host machine. The default is `true`.

  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.
- `writeQuota`: 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 nodes state is written to, similar to the "islands" concept used in previous OC4J releases. The default is `1` group member.
- `cacheMissDelay`: The length of time, in milliseconds, to wait in-process for another application 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 (1 second).
- `replicationPolicy`: Specifies the replication policy to apply. This policy defines when replication of data occurs.

The ideal values to set differ for Web modules and EJB modules. See the *Oracle Containers for J2EE Configuration and Administration Guide* for valid values.

- `propertyConfig`: Contains the properties that define the clustering communication protocol stack.
  - `propertyString`: A string containing the properties that define the clustering communication protocol stack.
  - `url`: A link to the XML configuration file that contains the properties that define the clustering communication protocol stack.
- `protocol`: Defines the mechanism to use for data replication. Note that only one can be specified. The default protocol used is `multicast`.
  - `database`: The connection information required to persist state data to a database.
    - `dataSource`: The name of the data source that will provide the database connection. This must be the JNDI name for the data source, which is the value of the `jndi-name` attribute specified in `data-sources.xml`.
  - `peer`: Contains the configuration required to use peer-to-peer (P2P) communication for replication.
    - `range`: The number of times to increment the port value while looking for a potential peer node. The default is 5 times. Valid only for configuring static peer-to-peer replication in a standalone OC4J installation.
    - `startPort`: The initial port to attempt to allocate for usage by this application cluster configuration for peer communication. The default is port `7800`. 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 (3 seconds). Valid only for configuring static peer-to-peer replication in a standalone OC4J installation.
    - `nodes`: Contains the host name and port of a node to poll for the list of available OC4J servers. Multiple nodes may be specified. Valid only for configuring static peer-to-peer replication in a standalone OC4J installation.
    - `bindAddr`: Optionally specifies the IP address of a Network Interface Card (NIC) to bind to. This is useful if you have OC4J host machines with multiple network cards, each with a specific IP address, and you wish to define which NIC is used to send and receive the multicast messages.
  - `multicast`: Contains the configuration required to use multicast communication for replication. This is the default mechanism used.
    - `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`.
    - `bindAddr`: Optionally specifies the IP address of a Network Interface Card (NIC) to bind to. This is useful if you have OC4J host machines with multiple network cards, each with a specific IP address, and you wish to define which NIC is used to send and receive the multicast messages.
- `replicationPolicy`: Defines when replication of `HttpSession` or 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. On the other hand, replicating data too infrequently can result in lost data in the event of server failure.

- `scope`: Defines what data is replicated: Either all attribute or variable values, or changed values only. By default, only modified HTTP session attributes are replicated; for stateful session beans, all member variables are replicated.
- `trigger`: 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.

connectorsPath

Defines a plug-in connector deployed with the application.

- `path`: The name and path of the `oc4j-connectors.xml` file. If not specified, then OC4J uses the default path, *`ORACLE_HOME`*`/j2ee/`*`instance`*`/connectors/`*`rarName`*`./oc4j-connectors.xml`.

dataSourcesPath

Specifies the path and file name of the XML file defining data sources to be used by the application.

The default `data-sources.xml` file, which defines the default data source used by OC4J, is installed in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config/` directory.

- `path`: The path to the XML file. The path can be fixed or relative to the location of the `orion-application.xml` descriptor, which is *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config/` by default.

defaultDataSource

Defines the default data source to use if other than server default. This must point to a valid CMT data source for this application if specified.

deploymentVersion

Defines the version of OC4J that this JAR was deployed against. If this version does not match the current version, then the JAR will be redeployed. This is an internal server value; do not edit.

enableIIOP

Set to `true` to enable IIOP. The default is `false`.

importedLibraries

Defines one or more shared libraries to be imported by the application, as well as one or more shared libraries to delete from the set of libraries inherited by default from the parent application.

- `editImport`: Specifies a shared library to be imported by the application.
  - `maxVersion`: The highest implementation version of the shared library to import. (To import the latest version of a shared library, do not specify a version.)
  - `minVersion`: The lowest implementation version of the shared library to import.
  - `name`: The name of the shared library to import.

- `editRemove`: Specifies a shared library to be removed from the set of shared libraries inherited by default from the application's parent. This includes shared libraries inherited from the server-level `system` and `default` applications, which are inherited by all applications deployed to OC4J by default.
  - `name`: The name of the shared library to remove.

jazn

Configures the Java Authentication and Authorization Service (JAAS) to use the XML-based configuration provider type. This property maps to the `<jazn>` element of `orion-application.xml`. For information about using the `<jazn>` element for an application, see the description of the `<jazn>` element of the `jazn.xml` file in the *Oracle Containers for J2EE Security Guide*.

- `provider`: Set to `XML`
- `location`: Set the path to the `jazn-data.xml` file (for example: `./jazn-data.xml`). This can be an absolute path, or a path relative to the `jazn.xml` file, where the JAAS Provider first looks for the `jazn-data.xml` in the directory containing the `jazn.xml` file. Optional if `jazn.xml` file configured, otherwise required.
- `persistence` Values are `NONE` (do not persist changes), `ALL` (persist changes after every modification), `VM_EXIT` (The default - persist changes when the JVM exits)
- `default-realm`: A realm name. For example: `sample_subrealm`. Optional if only one realm is configured.

For more information about JAAS configuration and the `jazn-data.xml` and `jazn.xml` files, see the *Oracle Containers for J2EE Security Guide*.

jaznLoginConfig

Contains data used to associate the application with a JAAS login module. These properties correspond to the `<jazn-loginconfig>` element and its subelements in the `jazn-data.xml` file and should be displayed only if the JAZN XML-based provider will be used by the application.

- `className`: The login module implementation class.
- `controlFlag`: Indicates whether the login module is required to succeed to proceed with authentication.
- `options`: A collection of one or more properties to pass to the login module as name and value pairs. For example: name `debug`, value `true`.

See the *Oracle Containers for J2EE Security Guide* for details about the `jazn-data.xml` file and these properties.

jmxMBeans

Defines one or more MBeans deployed with the application. Specify each of the following properties for every MBean being deployed.

- `className`: The MBean implementation class.
- `description`: A string containing a readable name for the MBean. This name will be displayed in the Application Server Control user interface.
- `objectName`: The name to register the MBean under. The domain part of the name will be ignored even if specified; application MBeans are registered using the application's deployment name as the domain name.

For example, if you deploy an MBean named MyMBeanA with an application named widget, supply :name=MyMBeanA as the value of this attribute. The name will then be displayed as widget:name=MyMBeanA.

libraries

Specifies either a relative or absolute path or URL to a directory or a JAR or ZIP archive to add as a library path for this OC4J instance. Directories are scanned for archives to include at OC4J startup.

- path: The path to the directory or archive.

log

Sets the logging configuration for the application.

- file: The optional path to the directory where text log files will be generated if text logging will be used by the application. The default location for text log files is *ORACLE_ HOME*/j2ee/*instance*/application-deployments/*application_ name*/application.log. The default log file name and path should *not* be modified.
- mail: Optionally set the e-mail address to which to mail log output. A valid mail-session must also be specified through the mailSessions property if this option is selected.
- odl: Configures Oracle Diagnostic Logging to be used by the application. The ODL framework 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 in XML, enabling other Oracle Application Server and custom-developed components to parse and reuse the log files more easily.
  - maxDirectorySize: 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.
  - maxFileSize: The maximum size, in bytes, that an individual log file is allowed to grow to. When this limit is reached, a new log file is generated.
  - path: The path to the folder to output the log.xml files for this component to. The path can be absolute or relative to the XML configuration file you are modifying.

    For example, to output log.xml files in a /hello-planet-xml directory within *ORACLE_HOME*/j2ee/*instance*/log/, set the path attribute to "../../log/hello-planet-xml".

mailSessions

Defines the mail session SMTP host, if SMTP is used by the application.

- location: The location within the namespace to store the mail session in.
- smtp-host: The session SMTP server host.

namespaceAccess

Sets the namespace or naming context security policy for RMI clients.

parentApp

Specifies the name of the application that serves as the parent of this application. This value is set during deployment.

persistencePath

Defines the path to a directory where application state data should be stored across application restarts. The path can be absolute or relative to the application root.

- `path`: The path to the persistence directory.

resourceProviders

Defines a resource provider.

- `className`: The name of the resource provider class.
- `name`: The name used to identify the resource provider. This name will be used in finding the resource provider in the application's JNDI as "`java:comp/resource/name/`".
- `description`: An optional description of the specific resource provider.
- `properties`: Set name and value property pairs to pass to the resource provider as parameters.

seeParentDataSources

Specifies whether the application should inherit the existing data sources defined for its parent application. The default is `false`.

This property is deprecated in OC4J 10*g* (10.1.3.5.0).

taskManagerInterval

Defines the interval at which the task manager performs its duties, in milliseconds. The task manager is a background process that performs cleanup activities. By default, it is started every second (`1000` milliseconds).

treatZeroAsNull

Specifies whether to read the value zero as null when it represents a primary key. The default is `false`.

userManagerByClass

Specifies an optional `UserManager` class to use, which can be any class that implements the `com.evermind.security.UserManager` interface. For example, `com.evermind.sql.DataSourceUserManager` or `com.evermind.ejb.EJBUserManager`. These classes are typically used to integrate existing systems and provide custom user-managers for Web applications.

- `className`: The fully qualified name of the `UserManager` class.
- `description`: An optional text description.
- `displayName`: A descriptive name for this `UserManager` instance.
- `properties`: One or more properties as name and value pairs to be passed to the `UserManager`. For example:

  ```
  name="groupMembershipGroupFieldName" value="group"
  name="groupMembershipUsernameFieldName" value="Userid"
  ```

Note: The `com.evermind.security` package and its classes are deprecated. They will no longer be supported in the 11*g* release. Instead of `com.evermind.security.UserManager` implementations, you can use JAAS custom login modules, described in the *Oracle Containers for J2EE Security Guide*.

---

webSiteBinding

Specifies the name of the Web site the application's Web module is bound to. By default, all applications deployed to OC4J are bound to the `default` Web site. See the *Oracle Containers for J2EE Configuration and Administration Guide* for details on creating and using Web sites in OC4J.

Setting Web Module Configuration Properties

The following subsections describe the OC4J-specific properties that you can set when deploying a Web module packaged in a WAR file. Each property maps to an element attribute in the `orion-web.xml` descriptor.

accessMask

Specifies optional access masks for this application. You can specify host names or domains to filter clients through the `hostAccess` property; specify IP addresses and subnets to filter clients in `ipAccess`; or define both.

- `default`: Specifies whether to allow requests from clients not identified through a `hostAccess` or `ipAccess` property. Supported values are `allow` (default) and `deny`.
- `hostAccess`: Specifies a host name or domain from which to allow or deny access.
  - `domain`: The host or domain.
  - `mode`: Whether to allow or deny access from the specified host or domain. Supported values are `allow` (default) or `deny`.
- `ipAccess`: Specifies an IP address and subnet mask from which to allow or deny access.
  - `ip`: The IP address, as a 32-bit value; for example: `123.124.125.126`.
  - `netmask`: The relevant subnet mask; for example: `255.255.255.0`.
  - `mode`: Whether to allow or deny access from the specified IP address and subnet mask. Supported values are `allow` (default) or `deny`.

autoJoinSession

Specifies whether users should be assigned a session as soon as they log in to the application. The default is `false`.

classpath

Specifies additional code locations for Web application class loading. These can be either library files or locations for individual class files.

- `path`: The path(s) to one or more code locations, separated by commas or semicolons. A location can be one of the following:
  – The complete path to a JAR or ZIP file, including the file name

– A directory path

If you specify a directory path, the class loader recognizes only individual class files in the specified directory, not JAR or ZIP files (unless those are specified separately).

The class loader recognizes the following:

– The `lib1.jar` and `zip1.jar` libraries (but no other libraries in `/abc/def`)
– Any class files in `/abc/def`
– Any class files in `mydir`, relative to the location of the generated `orion-web.xml` file

defaultBufferSize

Specifies the default size of the output buffer for servlet responses, in bytes. The default is `2048`.

defaultCharset

Specifies the ISO character set to use by default. The default is `"iso-8859-1"`.

defaultMimeType

Specifies the default MIME type to use for files with unknown or unrecognized extensions.

deploymentVersion

Specifies the version of OC4J under which this Web application was deployed. If this value does not match the current version, then the application is redeployed. This is an internal server value and should not be changed.

development

This property is a convenience for use during development. If set to `true`, then the OC4J server checks a particular directory for updates to servlet source files. If a source file has changed since the last request, then OC4J will, upon the next request, recompile the servlet, redeploy the Web application, and reload the servlet and any dependency classes.

The directory is determined by the setting of the `sourceDirectory` attribute (described in the following text).

directoryBrowsing

Specifies whether to allow directory browsing for a URL that ends in "`/`". Values are `allow` and `deny` (default).

Assume the following circumstances:

■ There is no `index.html`file in the directory the URL is mapped to.
■ There is no "welcome" page defined in the `web.xml` file.

If set to `allow` under these circumstances, then a URL ending in "`/`" results in the contents of the corresponding directory being displayed in the user's browser.

If set to `deny` under these circumstances, then a URL ending in "`/`" results in an error indicating that the directory contents cannot be displayed.

If there *is* a defined welcome file or there is an `index.html` file in the directory the URL is mapped to, then the contents of that file are displayed, regardless of this setting.

enableJspDispatcherShortcut

A `true` setting, which is the default, results in significant performance improvements by the OC4J JSP container, especially in conjunction with a `true` setting for the `simpleJspMapping` property. This is particularly true for JSP pages with numerous `jsp:include` tags.

Use of the `true` setting assumes, however, that if you define JSP files with `<jsp-file>` elements in `web.xml`, you have corresponding *`url-pattern`* definitions for those files.

expirationSettings

Defines the length of time before a specified set of resources - such as image files - will expire in the user's browser. This is useful for caching policies, such as for not reloading images as frequently as documents.

- `expires`: The number of seconds before the resources expiration, or `"never"` for no expiration. The default setting is "`0`" (zero), for immediate expiration.
- *`urlPattern`*: The URL pattern that the expiration applies to, such as in the following example:

  ```
  urlPattern="*.gif"
  ```

fileModificationCheckInterval

Defines the interval, in milliseconds, at which OC4J checks for modified files. This property applies only to static files, such as HTML files.

Within the time period since the last check, further checks are not necessary. Zero or a negative number specifies that a check always occurs. The default is `1000`. For performance reasons, a very large value (`1000000`, for example) is recommended in a production environment.

id

Defines a unique internal identifier generated at the time of deployment.

jaznWebApp

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

- `authMethod`: Supported values are `BASIC` (for basic J2EE authentication, the default) and `SSO`. Use `SSO` to employ Oracle Single Sign-On for HTTP client authentication. Use `BASIC` mode if the application uses a custom `LoginModule` instance.
- `runAsMode`: Set to `true` to invoke the servlet using the privileges of a particular *subject*. A subject is defined by an instance of the `javax.security.auth.Subject` class and includes a set of facts regarding a single entity, such as a person. Such facts include identities and security-related attributes, such as passwords and cryptographic keys.

  With the default `runas-mode="false"` setting, `doasprivileged-mode` is ignored.

- `doAsPrivilegedMode`: Assuming `runAsMode` is set to `true`, use the default `true` setting to use privileges of a particular subject without being limited by the access-control restrictions of the server.

  For additional information about JAAS and the features described for this element, see the *Oracle Containers for J2EE Security Guide*.

jspCacheDirectory

Specifies the JSP cache directory, which is used as a base directory for output files from the JSP translator. It is also used as a base directory for application-level TLD caching. The default value is `./persistence`, relative to the application deployment directory.

jspCacheTlds

Indicates whether persistent TLD caching is enabled for JSP pages. TLD caching is implemented both at a global level, for TLD files in "well-known" tag library locations, and at an application level, for TLD files under the `/WEB-INF` directory. Values are `standard` (default), `on` or `off`. Well-known locations are defined in the `jspTaglibLocations` property (documented in the following text). For more information on these values, see the *Oracle Containers for J2EE Support for JavaServer Pages Developer's Guide*.

jspPrintNull

Set to `false` to print an empty string instead of the default "null" string for null output from a JSP page. The default is `true`.

jspTaglibLocations

Defines the locations of shared tag libraries. This value can be set at the global application level only.

jspTimeout

Specifies an integer value, in seconds, after which any JSP page will be removed from memory if it has not been requested. This frees up resources in situations in which some pages are called infrequently. The default value is `0` (zero), for no timeout.

mimeMappings

Defines the path to a file containing MIME mappings to use.

- `path`: The path or URL for the file, either absolute or relative to the location of the `orion-web.xml` file.

persistencePath

Indicates where to store servlet `HttpSession` objects for persistence across server restarts or application redeployments. Specify a relative path, which will be relative to an OC4J temporary storage area under the `application-deployments` directory. There is no default value. If no value is defined, then there is no persistence of session objects across restarts or redeployments.

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

requestTrackers

Specifies one or more servlets to use as dedicated request trackers. Request trackers are useful for logging information, for example.

A request tracker is invoked for each separate request sent from a browser to the server, at the time that the corresponding response is committed (immediately before the response is actually sent).

There can be multiple request trackers, each one defined in a separate property.

- `servletName`: The servlet to invoke. You can specify either the servlet name or the class name, according to the corresponding `<servlet-name>` or `<servlet-class>` element (both of which are subelements of a `<servlet>` element) in the `web.xml` file.

servletChaining

Specifies a servlet to call when the response of the current servlet is set to a specified MIME type. The specified servlet is called after the current servlet. This is known as *servlet chaining*, for filtering or transforming certain kinds of output.

- `mimeType`: The MIME type used to trigger the chaining, such as `"text/html"`.
- `servletName`: The servlet to call when the specified MIME type is encountered. The servlet name is tied to a servlet class through its definition in the `<web-app>` element of `global-web-application.xml`, `web.xml`, or `orion-web.xml`.

servletWebdir

Specifies the path for invoking a servlet by class name. Anything appearing after this path in a URL is assumed to be a class name, including the package.

The following example illustrates servlet invocation by class name, assuming this property is set to `"/servlet/"`:

```
http://www.example.com:8888/servlet/foo.bar.ServletSession
```

This feature is typically for use in a standalone OC4J environment during development and testing. For deployment, use the standard `web.xml` mechanisms for defining the context path and servlet path.

sessionTracking

Specifies the session-tracking settings for this application. Session tracking is accomplished using cookies, assuming a cookie-enabled browser.

- `cookies`: Set to `enabled` (default) to send session cookies; set to `disabled` to disable this setting.
- `cookieDomain`: Set to the desired domain to use for cookies. You can use this attribute to track a single client or user over multiple Web sites. The setting must start with a period (`"."`) and must consist of at least two elements, such as `.us.oracle.com` or `.oracle.com`.

  In this case, the same cookie is used when the user visits any site that matches the `.us.oracle.com` domain pattern, such as `webserv1.us.oracle.com` or `webserv2.us.oracle.com`.
- `cookieMaxAge`: Set the maximum length of time, in seconds, that the browser will keep a cookie. By default, the cookie is kept in memory during the browser session and discarded afterward.
- `sessionTracker`: The name of the servlet to use as a session tracker.

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

- `servletName`: The servlet to invoke as a session tracker. Specify either the servlet name or the class name, according to the corresponding `<servlet-name>` or `<servlet-class>` element, both of which are subelements of a `<servlet>` element, in the Web module's `web.xml` file.

simpleJspMapping

Set to `true` if "`*.jsp`" is mapped *only* to the OC4J front-end JSP servlet, `oracle.jsp.runtimev2.JspServlet`, in the `<servlet>` elements of any Web descriptors affecting your application (`global-web-application.xml`, `web.xml`, and `orion-web.xml`). This allows performance improvements for JSP pages. The default setting is `false`.

sourceDirectory

Specifies the location of servlet source files to auto-compile if the `development` property is set to `true`. The default location is `/WEB-INF/src` if it exists; otherwise, it is `/WEB-INF/classes`.

temporaryDirectory

Contains the path to a temporary directory that can be used by servlets and JSP pages for scratch files. The path can be either absolute or relative to the deployment directory. The default setting is "`./temp`".

A servlet may use a temporary directory, for example, to write information to disk as a user is entering data in a form, for interim or short-term storage before the information is written to a database.

virtualDirectories

Adds a virtual directory mapping for static content. This is conceptually similar to symbolic links in a UNIX environment, for example.

The virtual directory enables you to make files in the *real* document root directory available to the application, even though the files do not physically reside in the Web application WAR file. This would be useful, for example, to link an enterprise-wide error page into multiple WAR files.

- `realPath`: The real path to the files that the virtual path will map to. For example, `/usr/local/realpath` in a UNIX environment or `C:\testdir` in a Windows environment.
- `virtualPath`: The virtual path to map to the specified real path.

webAppClassLoader

Contains class-loading instructions passed to the OC4J server at Web module startup.

- `searchLocalClassesFirst`: Set to `true` to search and load WAR file classes before system classes. The default setting is `false`.
- `includeWarManifestClassPath`: Set to `false` to *not* include the classpath specified in the WAR file manifest `Class-Path` attribute when searching and loading classes from the WAR file, regardless of the `searchLocalClassesFirst` setting. The default setting is `true`.

If both attributes are set to `true`, the overall class path is constructed so that classes physically residing in the WAR file are loaded prior to any classes defined in the WAR file manifest classpath. In the event of conflict, classes physically residing in the WAR file will take precedence.

Setting Enterprise JavaBeans Module Configuration Properties

The following details the OC4J-specific properties that can be set when you are deploying a J2EE application packaged in an EAR file. Each property maps to an element attribute in the `orion-ejb.xml` descriptor.

The properties that can be set differ for entity beans, session beans and message-driven beans. As such, the properties are broken out as follows:

- Setting General EJB Properties
- Setting Entity Bean Properties
- Setting Session Bean Properties
- Setting Message-Driven Bean Properties

Setting General EJB Properties

The following properties apply to all EJB modules within the archive being deployed.

defaultMethodAccess

Sets the default method access policy for insecure methods not associated with a role mapping. Methods are automatically mapped to the default security role.

- `impliesAll`: Set to `true` to disable security-role checking for insecure methods.

  If this attribute is set to `false`, you must map the default role defined in the `name` attribute to an Oracle Java Authentication and Authorization Service (JAAS) Provider or XML user group or user through the `groups` and `users` properties (described in the following items).

  The default is `false` if `<security-role-mapping>` is specified in the `orion-ejb-jar.xml` file and `impliesAll` is not set.

  If `<security-role-mapping>` is not specified in the `orion-ejb-jar.xml` file, the OC4J EJB layer defaults this attribute to `true`, and no security-role checking occurs for these methods.
- `name`: The default security role that insecure methods will be mapped to. The default is the `<default-ejb-caller-role>` role; however, this value can be changed to any valid role.
- `groups`: One or more user group names that will be used by clients to access insecure methods.
- `users`: One or more user names that will be used by clients to access insecure methods.

deploymentVersion

The version of OC4J the EJB JAR is being deployed into. This is an internal server value.

persistenceManager

Defines the persistence manager component to use to manage the persistence layer of entity EJB modules.

The TopLink utility is the default persistence manager (PM) used with OC4J, and by default, all EJB modules deployed to OC4J are managed by the TopLink PM. No configuration is required for TopLink. See the *Oracle TopLink Getting Started Guide* for details.

- `descriptor`: The file name of the persistence manager's deployment descriptor file.
- `name`: The name of the persistence manager implementation to use. Valid values are `toplink` if using TopLink, or `orion` if using the Orion CMP implementation. The default is `toplink`.
- `pmProperties:` Contains configuration properties for the TopLink persistence manager.
  - `customizationClass`: An optional Java class implementing the `oracle.toplink.ejb.cmp.DeploymentCustomization` interface used to allow deployment customization of TopLink mapping and runtime configuration. The class must be fully qualified and included in the EJB JAR being deployed.
  - `dbPlatformClass`: A TopLink database platform class containing TopLink support specific to a particular database. The specified class must be fully qualified.
  - `projectClass`: An optional TopLink project class containing mapping metadata. This class will replace the TopLink descriptor specified in the `descriptor` property. The class must be fully qualified and must exist in the EJB JAR file being deployed.
  - `remoteRelationships`: Set to `true` to maintain relationships between remote objects through the entities' remote interfaces. Note that this flag is not compliant with EJB 2.0. The default is `false`.
  - `sessionName`: A unique name for the EJB JAR being deployed. The name must be unique among all TopLink-persisted JARs deployed in OC4J. If no value is specified, a unique name will be generated by the TopLink persistence manager.
  - `mode`: Set whether updates should be propagated to another data store server synchronously or asynchronously. The default value is `asynchronously`.
  - `serverUrl`: The URL to the data store host.
  - `serverUser`: The user name to use to access the host.
  - `dbTableGen`: Specifies how TopLink will create or use the database tables being mapped to. This setting is ignored if the mappings are already defined for the entities. Values are as follows:
    - `Create`: Attempt to create the tables. This is the default.
    - `DropAndCreate`: Attempt to drop existing tables before re-creating them.
    - `UseExisting`: Use existing tables.
  - `extendedTableNames`: Set to `true` only if the generated table names are not long enough to be unique. This setting is ignored if the mappings are already defined for the entities. The default is `false`.

Setting Entity Bean Properties

The following properties apply to an entity bean included in the EJB archive.

callTimeout

Specifies the maximum time, in milliseconds, to wait for any resource to make a business/life-cycle method invocation. This is not a timeout for how long a business method invocation can take. The default is `90000` milliseconds (90 seconds).

If the timeout is reached, a `TimedOutException` is thrown. This excludes database connections.

Set to `0` if you want the timeout to be forever. See the EJB module section in the *Oracle Application Server Performance Guide* for more information.

copyByValue

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

dataSource

Specifies the name of the data source used if using container-managed persistence.

disableWrapperCache

Set to `true` to disable the wrapper class cache for the EJB module.

doSelectBeforeInsert

Specifies whether to execute a `SELECT` statement before an insert into the database. The extra select normally checks to see if the entity already exists before doing the insert to avoid duplicates.

The default value is `true`. For performance, Oracle recommends setting this property to `false` to avoid executing the extra statement.

However, if a unique key constraint is not defined for the entity, setting this value to `false` will allow a duplicate insert to avoid detection. To prevent duplicate inserts in this case, leave the value set to `true`.

findByPrimaryKeyLazyLoading

Set to `true` to turn on lazy loading and enforce only a single execution of the `select()` finder method. For entity bean finder methods, lazy loading can cause this method to be invoked more than once. The default is `false`.

instanceCacheTimeout

Specifies the amount of time, in seconds, that entity wrapper instances are assigned to an identity. Set to `never` to retain the wrapper instances until they are garbage collected. The default is `60` seconds.

iorSecurityConfigs

Configures CSIv2 security policies for interoperability. See *Oracle Containers for J2EE Security Guide* for details.

isolation

Specifies the isolation level for database actions.

- Valid values for Oracle databases are `serializable` and `committed` (default).
- Valid values for non-Oracle databases include the following: `none`, `committed`, `serializable`, `uncommitted`, and `repeatable_read`.

location

Defines the JNDI name to which the EJB module will be bound.

lockingMode

Configures the concurrency modes, which specify when to block for resource contention management, or when to execute in parallel. Values are:

- `optimistic`: Allows multiple users to execute the entity bean in parallel. It does not monitor resource contention; thus, the burden of the data consistency is placed on the database isolation modes. This is the default.
- `pessimistic`: Manages resource contention and does not allow parallel execution. Only one user at a time is allowed to execute the entity bean at a single time.
- `read-only`: Enables multiple users to execute the entity bean in parallel. However, the container does not allow any updates to the bean's state.

maxInstances

Sets the number of maximum bean implementation instances to be kept instantiated or pooled. The default is `100`. Set to `0` to indicates no maximum.

maxTxRetries

Specifies the number of times to retry a transaction that was rolled back due to system-level failures. The default is `3`.

Generally, you should add retries only where errors are seen that could be resolved through retries. For example, if you are using serializable isolation and you want to retry the transaction automatically if there is a conflict, you might want to use retries.

minInstances

Sets the number of minimum bean implementation instances to be kept instantiated or pooled. The default is `0`.

name

Specifies the name of the bean, which matches the name specified in the assembly section of the standard EJB deployment descriptor (`ejb-jar.xml`).

poolCacheTimeout

Defines the amount of time, in seconds, that the bean implementation instances are to be kept in the "pooled" or unassigned state. Set to `0` to retain bean instances until they are garbage collected. The default is `60`.

txRetryWait

Specifies the time to wait in seconds between retrying the transaction. The default is `60` seconds.

The `tx-retry-wait` attribute of the `<entity-deployment>` or `<session-deployment>` element in the `orion-ejb-jar.xml` file is not in `orion-ejb-jar-10_0.xsd` or `orion-ejb-jar.dtd`.

You can still specify `txRetryWait` to use the `tx-retry-wait` attribute in your `orion-ejb-jar.xml` file. If you use this attribute, however, do not configure OC4J to perform XML file validation (by using the `-validateXML` option on the OC4J startup command line).

validityTimeout

Sets the maximum amount of time, in milliseconds, that an entity is valid in the cache before being reloaded. This is useful for loosely coupled environments where rare updates from legacy systems occur. This attribute is only valid for entity beans with a locking mode of `read_only`.

If the EJB module is generally not modified externally, meaning that the table is occasionally updated and cache updates are required, set this to a value corresponding to the interval at which you think the data might be changing externally.

If the data is never modified externally the value can be set to `0` or `-1` to disable this option. The data in the cache will always be valid for read-only EJB modules that are never modified externally.

Setting Session Bean Properties

The following properties apply to a session bean included in the EJB archive.

callTimeout

Specifies the maximum time, in milliseconds, to wait for any resource to make a business/life-cycle method invocation. This is not a timeout for how long a business method invocation can take. The default is `90000` milliseconds (90 seconds).

Set to `0` for no timeout. See the EJB section in the Oracle Application Server 10g Performance Guide for more information.

copyByValue

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

idleTime

Specifies the timeout, in seconds, applied to stateful-session EJB modules. If the value is `0` or negative, then all timeouts are disabled. The default is `1800` seconds (30 minutes).

The timeout parameter is an inactivity timeout for stateful session beans. Every 30 seconds the pool cleanup logic is invoked. Within the pool cleanup logic, only the sessions that timed out, by passing the timeout value, are deleted.

Adjust the timeout based on your application’s use of stateful session beans. For example, if stateful session beans are not removed explicitly by your application, and the application creates many stateful session beans, then you may want to lower the timeout value.

iorSecurityConfigs

Configures CSIv2 security policies for interoperability. See the *Oracle Containers for J2EE Security Guide* for details.

location

Defines the JNDI name to which the EJB module will be bound.

maxInstances

Defines the maximum number of bean instances, either instantiated or pooled, allowed in memory. The default is `100`.

When this limit is reached, the container attempts to passivate the oldest bean instance from memory. If unsuccessful, the container waits the number of milliseconds set in `callTimeout` to see if a bean instance is removed from memory, through passivation, its `remove()` method, or bean expiration, before a `TimeoutExpiredException` is thrown back to the client.

Set to `0` to allow an infinite number of bean instances. This property applies to both stateless and stateful session beans.

maxInstancesThreshold

Specifies how many active beans can exist in relation to the value set for `maxInstances` before passivation is initiated.

Specify an integer that is translated as a percentage. For example, if `maxInstances` is `100` and `maxInstancesThreshold` is `90` percent, passivation of beans occurs when active bean instances reaches past a total of `90`.

The default is `90`. Set to `0` to disable this feature.

maxTxRetries

Specifies the number of times to retry a transaction that was rolled back due to system-level failures. The default is `3`.

Generally, you should add retries only where errors are seen that could be resolved through retries. For example, if you are using serializable isolation and you want to retry the transaction automatically if there is a conflict, you might want to use retries.

memoryThreshold

Defines a threshold for how much used JVM memory is allowed before passivation should occur. Specify an integer that is translated as a percentage. When the percentage is reached, beans are passivated, even if their idle timeout has not expired. The default is `80` percent. To disable, specify `never`.

minInstances

Sets the number of minimum bean implementation instances to be kept instantiated or pooled. The default is `0`.

name

Contains the name of the EJB module, which matches the name of a bean in the assembly section of the standard EJB deployment descriptor (`ejb-jar.xml`).

passivateCount

Defines the number of beans to be passivated if any of the resource thresholds - such as `maxInstancesThreshold` or `memoryThreshold` - has been reached. Passivation of beans is performed using the least recently used algorithm. By default, the number of beans is one-third of the value set for `maxInstances`. Set the count to `0` or a negative number to disable.

persistenceFilename

Defines the path to the file where sessions are stored across OC4J restarts.

poolCacheTimeout

Sets the amount of time, in seconds, that stateless sessions are to remain cached in the pool. At the specified interval, all unassigned beans in the pool are removed. The default is `60` seconds.

If the value specified is `0` or negative, beans are not removed from the pool.

replication

Defines when state replication should occur for stateful session beans in a clustered environment. Values are `VMTermination`, `EndOfCall` or `None` (default). See the *Oracle Containers for J2EE Configuration and Administration Guide* for details.

resourceCheckInterval

Defines the interval, in seconds, at which OC4J checks all resources to see which have passed the specified thresholds, such as `maxInstancesThreshold`. Passivation of beans occurs if any threshold has been reached. The default is `180` seconds (3 minutes). To disable, set to `0`.

timeout

Defines the inactivity timeout, in seconds, after which stateful session beans are considered ready for deletion from the pool. If the value is `0` or negative, then all timeouts are disabled. The default is `1800` seconds (30 minutes).

Adjust the timeout based on your application's use of stateful session beans. For example, if stateful session beans are not removed explicitly by your application, and the application creates many stateful session beans, then you might want to lower the timeout value. The pool cleanup logic is invoked every 30 seconds.

txRetryWait

Specifies the length of time to wait, in seconds, between retrying a transaction. The default is `60` seconds.

The `tx-retry-wait` attribute of the `<entity-deployment>` or `<session-deployment>` element in the `orion-ejb-jar.xml` file is not in `orion-ejb-jar-10_0.xsd` or `orion-ejb-jar.dtd`.

You can still specify `txRetryWait` to use the `tx-retry-wait` attribute in your `orion-ejb-jar.xml` file. If you use this attribute, however, do not configure OC4J to perform XML file validation (by using the `-validateXML` option on the OC4J startup command line).

Setting Message-Driven Bean Properties

The following properties apply to a message-driven bean within the archive being deployed. Each property pertains to attributes or sub-elements of a `<message-driven-deployment>` element in the `orion-ejb-jar.xml` descriptor.

connectionFactoryLocation

Contains the JNDI location of the connection factory to use.

The syntax is `java:comp/resource` + *`resource provider name`* + `TopicConnectionFactories` or `QueueConnectionFactories` + *`user defined name`*. The `ConnectionFactories` property details the type of factory that is being defined.

dequeueRetryCnt

Specifies how often the listener thread tries to reacquire the JMS session once database failover has occurred. The default is `0`. This value is only for container-managed transactions.

dequeueRetryInterval

Specifies the interval between attempts to reacquire the JMS session. The default is `60` seconds.

destinationLocation

Specifies the JNDI location of the destination (topic or queue) to use.

The syntax is `java:comp/resource` + *`resource provider name`* + `Topics` or `Queues` + *`Destination name`*. The `Topic` or `Queue` details what type of Destination is being defined. *`Destination name`* is the actual queue or topic name defined in the database.

listenerThreads

Sets the number of listener threads spawned to listen for incoming JMS messages on the topic or queue. The threads concurrently consume JMS messages. Topics can only have one thread; queues can have more than one thread. The default is `1` thread.

maxInstances

Do not set his property; use `listenerThreads` instead.

minInstances

Do not set this property.

name

Contains the name of the EJB module, which matches the name of an EJB module in the assembly section of the J2EE standard EJB deployment descriptor (`ejb-jar.xml`) packaged in the EJB archive.

resourceAdapter

Contains the name of the resource adapter to be created by the connection factory for use by this bean. This value is defined in `orion-ra.xml`, the OC4J-specific resource adapter descriptor.

subscriptionName

Contains the topic subscription name, if the bean represents a topic.

transactionTimeout

Sets the transaction timeout interval, in seconds, for any container-managed transactional MDB. The default is `86400` seconds (one day). If the transaction has not completed within this timeframe, the transaction is rolled back.

Setting Web Services Configuration Properties

Each property maps to an element attribute in the `oracle-webservices.xml` descriptor.

- Setting General Web Services Properties
- Setting Web Service Description Properties

Setting General Web Services Properties

The following properties apply to all Web services deployed with the archive.

contextRoot

Specifies the context root of the exposed Web service, which is required only for EJB 2.1 Web services. If the context root is not specified, the server defaults to the EJB JAR file name minus the `.jar` extension. For example, `foo-ejb.jar` will be translated into the context root `/foo-ejb`.

For Java class Web services, the context root is derived from the context root specified in the parent application's `application.xml` descriptor.

Setting Web Service Description Properties

The following properties apply to a Web service contained within the archive being deployed.

downloadExternalImports

Specifies whether relative imports should be downloaded and resolved to absolute URLs. If set to `true`, `resolveRelativeImports` is automatically set to `true`. The default is `false`.

exposeTestpage

Set to `true` to expose the test page. The default is `true`.

exposeWsdl

Set to `true` to expose the WSDL file describing the Web service. The default is `true`.

name

Defines the name of the Web service for an EJB module implemented as a Web service. This value matches the name defined in the J2EE standard EJB deployment descriptor (`ejb-jar.xml`).

resolveRelativeImports

Set to `true` to force relative imports to be resolved to absolute URLs. This property is automatically set to `true` if `downloadExternalImports` is `true`.

wsdlFileFinalLocation

Specifies the URI for the final updated WSDL that describes the Web service.

wsdlPublishLocation

Specifies the URI to the WSDL if `exposeWsdl` is set to `true`.

Setting Application Client Configuration Properties

The following details the OC4J-specific properties that can be set when deploying an application client packaged in a JAR or CAR file. Each property maps to an element attribute in the `orion-application-client.xml` descriptor.

clientInvocationMappings

Configures the client module to start when the OC4J container is started.

- `autoStart`: Set to `true` to start the client when OC4J is started. The default is `false`. If this property is set to `true`, the `user` property must be set to `anonymous`.

- `user`: The unique identifier of the user the client will run as. If the `autoStart` property is set to `true`, the `user` property must be set to `anonymous`.
- `path`: The path to the client JAR file. Can be absolute or relative to the parent application's root.
- `arguments`: One ore more string arguments to be passed to the client's `main()` method at startup.

mailSessions

Contains the configuration for a mail session resource.

- `description`: An optional description for the resource.
- `location`: The JNDI name to bind the mail session to.
- `smtpHost`: The name or IP address of the SMTP host to use, if using SMTP.
- `username`: A user name used to log in to the resource, if required.
- `password`: The password used to access the resource.
- `properties`: Can contain additional properties that should be given to the mail session as name and value pairs. For example:

  ```
  name="mail.from" value="mail.sender@server.com
  name="mail.transport.protocol" value="smtp"
  name="mail.smtp.from" value="mail.sender@server.com
  ```

ejb-ref

Declares a reference to the home interface of an EJB module that will be used by the client.

- `location`: The JNDI location to which the EJB module is bound.
- `name`: The EJB reference name used by the application client.

Setting Resource Adapter Properties

The following text details the OC4J-specific properties that you can set when deploying a resource adapter packaged in a RAR file. Each property maps to an element attribute in the `oc4j-ra.xml` descriptor.

See the *Oracle Containers for J2EE Resource Adapter Administrator's Guide* for detailed information on resource adapter configuration.

connectionPools

Contains a name and one or more name and value pairs defining a shared connection pool. Valid property names are as follows:

- `maxConnections`: The maximum number of connections permitted within the pool. If no value is specified, there is no limit on the number of connections.
- `minConnections`: The minimum number of connections. If greater than 0, the specified number of connections are opened when OC4J is initialized. OC4J may not be able to open the connections if necessary information is unavailable at initialization time.

  For instance, if the connection requires a JNDI lookup, it cannot be created, because JNDI information is not available until initialization is complete. The default value is `0`.

- `scheme`: Defines how OC4J handles connection requests after the maximum permitted number of connections is reached. One of the following values must be specified:
  - `dynamic`: OC4J creates a new connection and returns it to the application, even if this violates the maximum limit. When these limit-violating connections are closed, they are destroyed instead of being returned to the connection pool.
  - `fixed`: OC4J raises an exception when the application requests a connection and the maximum limit has been reached.
  - `fixed_wait`: OC4J blocks the application's connection request until an in-use connection is returned to the pool. If `waitTimeout` is specified, OC4J throws an exception if no connection becomes available within the specified time limit.
- `waitTimeout`: The maximum number of seconds that OC4J waits for an available connection if maxConnections has been exceeded and the fixed_wait scheme is in effect. In all other cases, this property is ignored.

connectorFactories

Contains properties defining an installed J2EE Connector Architecture-compliant resource adapter.

- `connectionFactoryInterface`: The name of the factory that will create managed connection instances, such as `javax.jms.QueueConnectionFactory`.
- `connectorName`: The name of the connector.
- `description`: An optional short description of the connector.
- `location`: The JNDI location which OC4J will bind the connection factory to. For example, if the value is set to `eis/myEIS1`, an application component can look up the connection factory using JNDI lookup on location `java:comp/env/eis/myEIS1`.
- `configProperties`: Configuration properties for the connector factory defined in `connectionFactoryInterface`.

  Configuration properties are connector-specific and are defined in the `<config-property>` element in the J2EE deployment descriptor (`ra.xml`) packaged with the resource adapter. For a JCA 1.0 resource adapter, the `<config-property>` element is a sub-element of the `<resourceadapter>` element. For JCA 1.5, they are found inside the `<connection-definition>` element(s).
- `connectionPooling`: Contains properties defining a connection pool to use. Properties are:
  - `use`: Specifies whether a connection pool should be used for the connection factory; and if so, whether a shared pool or a private pool should be used. Possible string values for this attribute are:
    - `shared`: The shared connection pool to be used for this connection factory configuration.
    - `private`: The private, nonshared connection pool defined in the `properties` field, described in the following text under `principalMappingInterface`, will be used for this connection factory configuration.

    * `none`: Connection pooling is disabled for this connection factory configuration.

- `log`: The path to the log file generated by OC4J for this connector. If the path name is not specified or if the directory does not exist, logging is not enabled. For example: `./logConnFctry1.log`
- `securityConfig`: Specifies the security mechanism to use and corresponding credential information needed by the connector to access the EIS.
  - `use`: The security mechanism to use. Only one may be specified. Valid values are: `jaasModule`, `principalMappingEntries` or `principalMappingInterface`.
  - `jaasModule`: If using JAAS, the name of the JAAS application to use.
  - `principalMappingEntries`: If using the principal mapping entries mechanism, specify the following:
    * `defaultMapping`: Specify the resource user name and password for the default resource principal. This principal is used to log in to the EIS if there is no principal mapping entry whose initiating user corresponds to the current initiating principal.
    * `principalMappingEntries`: Specify the initiating principal user name and the resource user name and password. This maps the initiating principal to the resource principal and password it will use to log in to the EIS.
  - `principalMappingInterface`: If using the OC4J-specific programmatic container-managed sign-on mechanism, which utilizes a class implementing the `oracle.j2ee.connector.PrincipalMapping` interface, specify the following:
    * `implClass`: The class implementing the `oracle.j2ee.connector.PrincipalMapping` interface. A JAR file containing the class must be placed into the directory containing the decompressed RAR file.
    * `properties`: The user name and password that will be used to log in.
- `xaRecoveryConfig`: The user name and password of the privileged user that will log in to the EIS during a two-phase commit transaction recovery.

9 Using Application Server Control for Deployment

Oracle Enterprise Manager 10g Application Server Control provides a Web-based interface for completing a range of deployment-related tasks 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.

You can perform these deployment operations with Application Server Control:

- Deploy an application (EAR), a standalone Web module (WAR), a standalone EJB module (EJB JAR), or a standalone resource adapter (RAR)
- Undeploy an application, Web module, EJB module, or resource adapter
- Create, modify, or remove shared libraries
- Start, restart, or stop applications
- Restart or stop an OC4J instance or group of instances
- Manage data sources and connection pools
- Manage JMS resources
- Create or edit a reusable deployment plan
- Set application-specific security and application-clustering configurations

See the online help provided with Application Server Control for detailed instructions on using the various features provided.

You can perform similar deployment tasks with Ant tasks or the `admin_client.jar` command-line utility. Chapter 10, "Using OC4J Ant Tasks for Deployment" describes the OC4J Ant tasks for deployment and how to use them. Chapter 11, "Using the admin_client.jar Utility for Deployment" explains how to use `admin_client.jar` for deployment tasks.

This chapter includes the following sections:

- Accessing Application Server Control
- Setting Log Levels
- Deploying an Application to an OC4J Instance or Group of Instances
- Deploying a New Application Version with Side-by-Side Application Upgrade
- Undeploying an Application
- Creating and Managing Shared Libraries

- Starting, Restarting, and Stopping Applications
- Restarting and Stopping OC4J Instances
- Managing Data Sources and Connection Pools for OC4J Instances
- Managing JMS Resources

Accessing Application Server Control

Application Server Control is installed and configured automatically when you install OC4J. This section covers the following topics:

- Accessing Application Server Control in Standalone OC4J
- Accessing Application Server Control in Oracle Application Server

For more information about using this interface, see the online help provided with Application Server Control.

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.

You can access Application Server Control through the `default` Web site, which is configured to listen for HTTP requests on port 8888. To access Application Server Control, 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 automatically when you install OC4J using the Oracle Universal Installer.

You can use the OPMN command-line tool to start Application Server Control with all other installed Oracle Application Server components. The OPMN command-line tool, `opmnctl`, is installed in the *`ORACLE_HOME`*`/opmn/bin` directory on each server node. Start all installed components by issuing the following command:

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

Note: In a cluster topology that includes multiple OC4J instances, use the `-sequential` flag after `startall` to prevent resource contention that might occur if you started all instances in parallel. You can specify the `-sequential` option in the OPMN configuration file for the cluster topology, `opmn.xml`, 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>
```

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

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

- *`ohs_host_address`* is the address of the OHS host machine; for example, `server07.company.com`
- *`port`* is an HTTP listener port assigned to OHS by OPMN. Run the following `opmnctl` command on the OHS 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
  ```

Setting Log Levels

In OC4J 10*g* (10.1.3.5.0), you can set the log levels for loggers through Application Server Control, as follows:

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.

Deploying an Application to an OC4J Instance or Group of Instances

Application Server Control enables you to deploy an application to a specific OC4J instance or to a group of OC4J instances, as follows:

1. Click **Cluster Topology** on the Application Server Control home page. The resulting page displays the following items:
   - All Oracle Application Server instances that are currently part of the cluster topology
   - The active OC4J instances within each Oracle Application Server instance
   - The applications deployed into each OC4J instance
2. Select the deployment target:
   - To deploy to a specific OC4J instance, click the link for the target instance to which you want to deploy the application.
   - To deploy to a group of OC4J instances, click the name of the group under Groups at the bottom of the page.
3. After the target instance or instances have been accessed, deploy the application, as the following topics explain:
   - Deploying or Redeploying an Application

- Completing Configuration Tasks Before Deployment

Deploying or Redeploying an Application

Application Server Control includes a three-page deployment wizard that provides a streamlined, user-friendly deployment process.

Note: If the HTTP session times out due to browser inactivity while you are using the deployment wizard, you will have to restart the deployment process from the beginning.

---

1. Click the **Applications** tab for an OC4J instance and then the **Deploy** button to access the deployment wizard.
2. Select the archive to upload to the OC4J server in the first page of the wizard.

   You also have the option to navigate to the location of an existing deployment plan, which can be applied to the archive or used as a template for a new deployment plan. If no deployment plan is specified, a new deployment plan will be created by default. See Chapter 8, "Working with Deployment Plans" for more information.
3. Set application attributes and bind the Web application to a Web site in the second page of the deployment wizard.

   Specify the application or module name, which will be used to identify the application within OC4J. This name will also be displayed in Application Server Control.

   Next, select the parent application of the application or module being deployed. If no parent application is specified, the `default` application is used.

   Finally, if deploying a Web application, bind the application to the Web site that will be used to access it. This binding is accomplished by selecting the name of the XML configuration file that defines the Web site from the list.

   The list contains all Web sites currently defined for the OC4J server instance. In most cases, applications will be bound to the `default` Web site, which is defined by the `default-web-site.xml` configuration file.
4. Complete the deployment tasks, edit the deployment plan directly, or do both before deploying the archive in the third page of the deployment wizard.

   In this final screen, you have the option of completing a number of configuration tasks before deploying the application. These tasks provide an alternative to editing the deployment plan, which contains configuration data that will be set in the OC4J deployment descriptors generated during deployment. See "Completing Configuration Tasks Before Deployment" on page 9-5 for details on each optional task.

   You also have the option of editing the deployment plan directly before deploying the archive. The edited deployment plan can then be saved for reuse. See Chapter 8, "Working with Deployment Plans" for more details on creating and editing deployment plans.
5. Deploy the application.

   The archive is not actually deployed until you click the **Deploy** button. The deployment plan, which up until this point exists only on the client side, will also be sent to the OC4J server with the archive. Once started, the deployment process will continue, even if the Web browser is closed.

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.

Completing Configuration Tasks Before Deployment

The third page of the Application Server Control deployment wizard gives you the option to complete a number of configuration tasks before deploying an application. You can complete similar tasks through the deployment plan editor, as "Creating or Editing Deployment Plans with Oracle JDeveloper" on page 8-4 describes.

Just as with the deployment plan editor, values set through the various deployment tasks pages are written to the appropriate OC4J-specific deployment descriptor at deployment time. The following topics describe the configuration tasks for deployment:

- Selecting the Security Provider
- Mapping Security Roles to Users and User Groups
- Configuring Enterprise JavaBeans Modules Included in the Application
- Managing Class Loading to Import Shared Libraries
- Configuring Application Clustering
- Providing Resource Mappings

Selecting the Security Provider

OC4J supports two different provider types: XML for application development mode, and LDAP for production environments. Each provider type implements a repository for secure, centralized storage, retrieval, and administration of provider data.

- Select **File-Based Security Provider** to use the XML-based provider.

  The XML-based provider is a lightweight implementation suitable for application prototyping in a development environment. User, realm, and policy information is stored in an XML file, normally `system-jazn-data.xml`.

- Select **Oracle Identity Management Security Provider** to use the LDAP-based Oracle Internet Directory provider.

  This security provider is useful for applications being deployed into a production environment. User, realm, and policy information is persisted to the LDAP-based Oracle Internet Directory (OID).

  Note that the OC4J instance must be configured to use Oracle Internet Directory before an application can be configured to use it.

- Select **Oracle Security Provider for 3rd Party LDAP Server** to use a non-Oracle LDAP provider.

  Select this option to configure the application to use Active Directory, Sun Directory Server or another LDAP server. Use the tools provided by the LDAP server vendor for realm and principals management.

Mapping Security Roles to Users and User Groups

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

Configuring Enterprise JavaBeans Modules Included in the Application

You can configure a number of properties for the Enterprise JavaBeans (EJB) modules packaged with the application being deployed:

- Configure the following properties for each entity bean packaged with the application. The values displayed are the default values.

Table 9–1 Entity Bean Properties

Property	Values
Persistence Type	Indicates whether the bean will use bean-managed or container-managed persistence.
Min Instances	Sets the number of minimum bean implementation instances to be kept instantiated or pooled.
Max Instances	Defines the maximum number of bean instances, either instantiated or pooled, allowed in memory.
Max Transaction Retries	Specifies the number of times to retry a transaction that was rolled back due to system-level failures.
Pool Cache Timeout	Sets the amount of time, in seconds, that stateless sessions are to remain cached in the pool. At the specified interval, all unassigned beans in the pool are removed.
JNDI Name	Defines the JNDI name to which this EJB module will be bound.

- Select the data sources to associate with EJB modules containing entity beans.

  By default, all entity beans deployed to OC4J will use the Oracle TopLink persistence manager. You can create data sources can be created through the JDBC Resources page of Application Server Control. To get to this page, select **JDBC Resources** under **Services** on the Administration tab of the OC4J Home page.

- Configure the following properties for each session bean packaged with the application. The values displayed are the default values.

Table 9–2 Session Bean Properties

Property	Values
Min Instances	Sets the number of minimum bean implementation instances to be kept instantiated or pooled.
Max Instances	Defines the maximum number of bean instances, either instantiated or pooled, allowed in memory.
Max Transaction Retries	Specifies the number of times to retry a transaction that was rolled back due to system-level failures.

Table 9–2 (Cont.) Session Bean Properties

Property	Values
Call Timeout	Specifies the maximum time, in milliseconds, to wait for any resource to make a business/life-cycle method invocation.
Pool Cache Timeout	Sets the amount of time, in seconds, that stateless sessions are to remain cached in the pool. At the specified interval, all unassigned beans in the pool are removed.
JNDI Name	Defines the JNDI name to which the EJB module will be bound.

- Configure the following for each message-driven bean packaged with the application. The values displayed are the default values.

Table 9–3 Message-Driven Bean Properties

Property	Values
Dequeue Retry Count	Specifies how often the listener thread tries to re-acquire the JMS session once database failover has occurred. Valid for container-managed transactions only.
Dequeue Retry Interval	Specifies the interval between attempts to re-acquire the JMS session.
Transaction Timeout	Sets the transaction timeout interval, in seconds, for any container-managed transactional MDB.
Number of Listener Threads	Sets the number of listener threads spawned to listen for incoming JMS messages on the topic or queue. Topics can only have one thread; queues can have more than one thread.

For information about deploying EJB modules, see Chapter 3, "Deploying Enterprise JavaBeans Modules." For information about updating EJB modules in a deployed application, see "Incremental Redeployment of Updated EJB Modules" on page 3-4.

Managing Class Loading to Import Shared Libraries

You can manage the libraries imported by the application. By default, an application inherits the same set of shared libraries present in its parent application, including any shared libraries inherited from the `default` application.

The default page displays all of the shared libraries currently installed on the OC4J server instance. Shared libraries must have already been installed on the OC4J server instance to be imported.

- Select the **Import** box next to each shared library to import. To use the latest installed version of the library, do not specify a version number.
- Specify a minimum or maximum version of a shared library to import.

  Use this feature to import a different version of a shared library than that inherited from the application's parent. For example, you could import an earlier version of the Oracle JDBC driver than that inherited from the OC4J `default` application by specifying the maximum version to import.
- Deselect the **Import** box for any shared library that should not be inherited from the parent application.

  To remove all inherited shared libraries, deselect the **Inherit parent application's imported shared libraries** checkbox. This action adds the following `<remove-inherited>` element to the `orion-application.xml` file that will be generated for the application at deployment time:

```
<imported-shared-libraries>
  <remove-inherited name="*" />
</imported-shared-libraries>
```

- Optionally specify additional code sources to add as library paths to the OC4J instance.

  To add a code source, specify either a relative or absolute path or URL to the archive. Specified directories are scanned for archives to load at OC4J server startup.

- Manage the class loader created for each Web module.

  OC4J creates a class-loader instance for each Web module deployed as a WAR into the server instance.

  Check the **Search Local Classes First** checkbox to force the class loader to also load JARs containing classes and resources packaged within the WAR before loading JARs external to the WAR.

  For example, suppose you want to ensure that your Web module uses the version of `log4j` packaged within your WAR, and not use the version of `log4j` bundled with a resource adapter deployed into the OC4J instance. Selecting this option forces the class loader to load the local log4j JAR file contained within the WAR.

  If the WAR contains a MANIFEST.MF, you can force JARs declared as named extensions to be loaded by checking the **Include WAR Manifest Class Path** box. You can also specify the relative or absolute paths to one or more additional code sources containing classes or resources to be loaded in the **Classpath** field. Using either of these features causes classes to be loaded by the Web module's class loader, just as if they were packaged within the WAR.

Configuring Application Clustering

The OC4J clustering framework supports replication of objects and values contained in an HTTP session or an instance of a stateful session EJB module across OC4J instances.

By default, applications inherit the clustering configuration set at the parent application level. However, clustering can also be configured at the application level at deployment time. A configuration property set at the application level overrides the corresponding value inherited from the parent application.

- Select **Enable Clustering** to enable clustering support for the application. The value selected overrides the setting inherited from the application's parent.

  If clustering has been enabled at the parent application level, the parent's configuration will be applied by default.

- Select **Peer-to-Peer Replication** to enable the peer-to-peer replication mechanism. The mechanism used is dependent on the type of OC4J installation:

  – OC4J instances within Oracle Application Server

    In a cluster topology, dynamic peer-to-peer discovery is used to enable OC4J instances to dynamically discover and communicate with one another.

    Dynamic peer-to-peer discovery is used by default if clustering is enabled; no additional configuration is needed.

    You can specify the IP address for a 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.

  - Standalone OC4J installation

    In a standalone configuration, each JVM in the OC4j instance can be statically configured to recognize at least one other peer JVM. As a JVM becomes aware of each of its peers, it also becomes aware of each peer's peer or peers, with the end result that all of the JVMs in the OC4J instance become aware of one another.

- Select **Multicast Replication** to configure OC4J to send and receive HTTP session and stateful session bean state changes via multicast packages.

  This is the default replication protocol used in a standalone OC4J installation. The multicast address and port must be the same for all instances in the cluster. The OC4J default address is `230.230.0.1`; the default port is `45566`.

  Note that you can optionally specify the IP address for a 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

- Select **Database Replication** to replicate application state to a database.

  Select the JNDI name of the data source providing the connection to the database from the **Database JNDI Location** pull-down menu. See the *Oracle Containers for J2EE Services Guide* for details on defining and using data sources.

Providing Resource Mappings

Any resources included in the EAR file being deployed will be displayed in Application Server Control. Map any environment references in your application to physical entities currently present in the operational environment, including JMS topics and queues, data sources, and resource adapters.

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

For most applications, the resource reference you must designate is the data source JNDI name. You cannot configure the data source information through deployment tasks in Application Server Control, which enable you to designate an already configured data source or a data source that you can configure later. Designate the JNDI location name of the data source that the application will use.

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

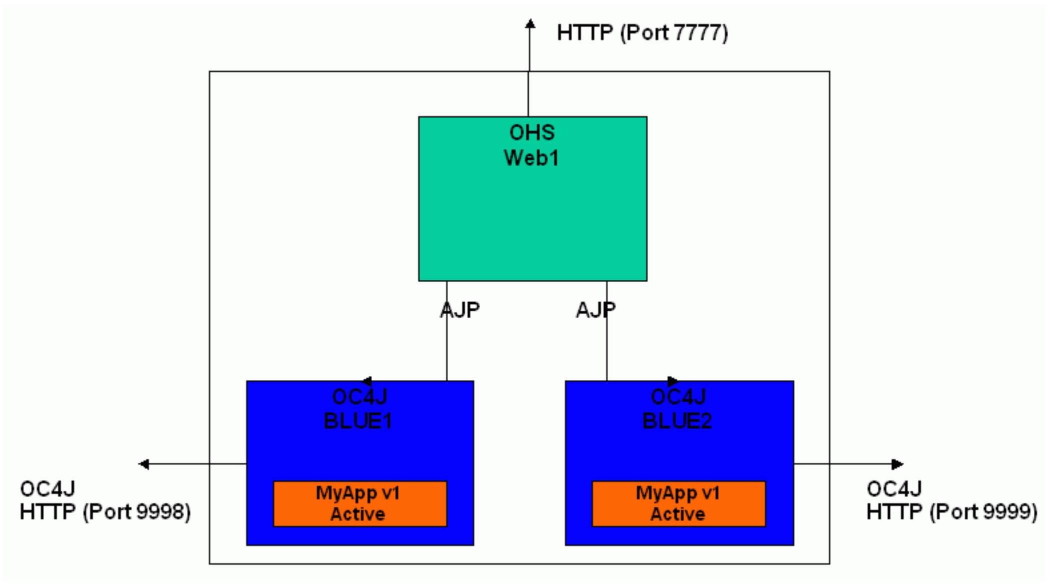
Deploying a New Application Version with Side-by-Side Application Upgrade

Side-by-side application upgrade enables hot deployment of a new version of an application while one or more active instances of an earlier version of the application complete gracefully. This feature includes version management capabilities, such as changing the default version of an application, deploying and undeploying different versions, and retiring a version.

With side-by-side application upgrade, you can have one version of a J2EE application running and then deploy a second version of the application without having your end users experience any down time. You can test the new version of the application when it gets deployed to the production environment, before exposing the new version to end users. This testing ensures that the application is configured correctly for the production environment.

The minimum architecture for a side-by-side application upgrade to work is two independent OC4J instances managed by OPMN and an OHS instance within an Oracle Application Server instance. Each OC4J instance must have an internal HTTP Web site through which requests can go directly to OC4J as well as an AJP Web site through which OHS can route requests to OC4J. Both OC4J instances must be running the application. Figure 9–1 shows the initial state of the two OC4J instances before deployment of a new application version with side-by-side application upgrade.

Figure 9–1 Two OC4J Instances Set Up for Side-by-Side Application Upgrade



The OC4J instances do not have to be members of the same group. If they are, you can do the initial deployment and configuration of the application with group operations. The deployment tasks for a side-by-side application upgrade, however, must be done to one OC4J instance at a time, whether or not it is a member of a group.

In the side-by-side application upgrade procedure, the second OC4J instance continues running version 1 of the application and preserves session state while you stop the application on the first instance, which forces all new traffic on the production Web site (AJP) to the second instance. Then you can deploy version 2 of the application to the first instance and bind version 2 to the HTTP Web site for testing. When the testing succeeds, you can bind version 2 to the AJP Web site to make it publicly accessible. After the production system stabilizes, you can stop version 1 on the second OC4J instance and deploy and test version 2 there.

For this procedure to work, you need to make sure that the requests being served by both application versions are compatible so that version 1 and version 2 can coexist with requests being routed to them. If you use state replication to maintain state across version 1 and version 2, you need to make sure that the state being maintained is compatible between the application versions.

This section shows how to use side-by-side application upgrade to deploy an application mainly with Application Server Control. You can also perform the deployment tasks with Ant tasks, described in Chapter 10, Using OC4J Ant Tasks for Deployment," or with `admin_client.jar` commands, described in Chapter 11, "Using the admin_client.jar Utility for Deployment."

How to Create an Additional HTTP Listener in an OC4J Instance

An OC4J instance has an AJP listener by default. You can create an additional HTTP listener in an OC4J instance so that it has an internal Web site that uses the `http` protocol as well as the default, public Web site that uses the `ajp` protocol. Then you can bind an application the HTTP Web site for testing before you make the application available to end users on the AJP Web site.

To create an additional HTTP listener in an OC4J instance:

1. Edit the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config/server.xml` file to add a `<web-site>` element like this one:

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

2. Create a new `*-web-site.xml` file and specify a different port number from the port for the AJP Web site.

   Example 9–1 shows the contents of a file named `c:\OracleAS\j2ee\blue1\config\internal-web-site.xml`.

Example 9–1 *-web-site.xml file to Add an HTTP Listener

```
<?xml version="1.0"?>
<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="9999" protocol="http" display-name="Internal HTTP Site"
schema-major-version="10" schema-minor-version="0" >
<default-web-app application="default" name="defaultWebApp" root="/j2ee" />
<access-log path="../log/internal-web-access.log" split="day" />
</web-site>
```

3. Edit the `opmn.xml` file to add the new port entry for the OC4J instance:

   ```
   <start timeout="600" retry="2"/>
   <stop timeout="120"/>
   <restart timeout="720" retry="2"/>
   <port id="default-web-site" range="7700" protocol="ajp"/>
   <port id="internal-web-site" range="9999" protocol="http"/>
   <port id="rmi" range="12401-12500"/>
                  <port id="rmis" range="12701-12800"/>
                  <port id="jms" range="12601-12700"/>
                  <process-set id="default_group" numprocs="1"/>
   ```

4. Reload the `opmn.xml` file.

5. Restart the OC4J instance.

   You now have your external listener coming through OHS (AJP), and you have an internal listener going directly to the OC4J instance. You can see the HTTP port in the output from the `opmnctl status -l` command:

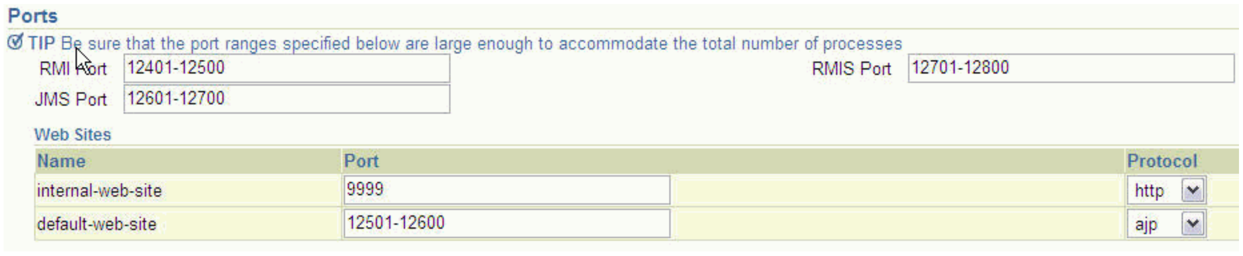
   ```
   Processes in Instance: ohs_j2ee_changed
   -------------------------------+--------------------+---------+----------+---
   ---------+----------+----------+------
   ias-component                  | process-type       |     pid | status   |
   uid |  memused |    uptime | ports
   -------------------------------+--------------------+---------+----------+---
   ---------+----------+----------+------
   OC4JGroup:default_group        | OC4J:foo           |    4856 | Alive    |
   768545086 |   194428 |   0:05:56 |
   ```

```
jms:12601,ajp:12501,http:9999,rmis:12701,aussie-tripper:3762,rmi:12401
```

6. Go to the new HTTP listener and verify that it is valid and accepting requests.

You can also use Application Server Control to control the port and protocol of the Web site, on the Runtime Ports page, as Figure 9–2 shows.

Figure 9–2 Ports and Protocols for OC4J Web Sites



Ports

TIP Be sure that the port ranges specified below are large enough to accommodate the total number of processes

RMI Port	12401-12500	RMIS Port	12701-12800
JMS Port	12601-12700		

Web Sites

Name	Port	Protocol
internal-web-site	9999	http
default-web-site	12501-12600	ajp

For more information about creating and configuring Web sites, see "Managing Web Sites in OC4J" in *Oracle Containers for J2EE Configuration and Administration Guide*.

How to Deploy an Application with Side-by-Side Application Upgrade

With side-by-side application upgrade, you can deploy a new version of an application while still servicing the old version, on two OC4J instances within a single Oracle Application Server 10*g* Release 3 (10.1.3.5.0) instance. The Oracle Application Server instance also includes Oracle HTTP Server (OHS), based on Apache. This setup enables you to perform an in-place upgrade of an application.

To deploy an application with side-by-side application upgrade:

1. Set up two OC4J instances front-ended by OHS to have both AJP and HTTP listeners, as in Figure 9–1.

   The AJP listener will handle normal user traffic, routed from OHS. You can use the HTTP listener for internal testing. Application instances will be bound to the AJP listener when running in their normal state.

   For information about adding HTTP listeners, see "How to Create an Additional HTTP Listener in an OC4J Instance" on page 9-11.

2. With the same version of the application running in both OC4J instances, ensure that requests are being served correctly.

3. Stop the application on the first OC4J instance:

   a. From the Cluster Topology page of Application Server Control, click the name of the first OC4J instance.

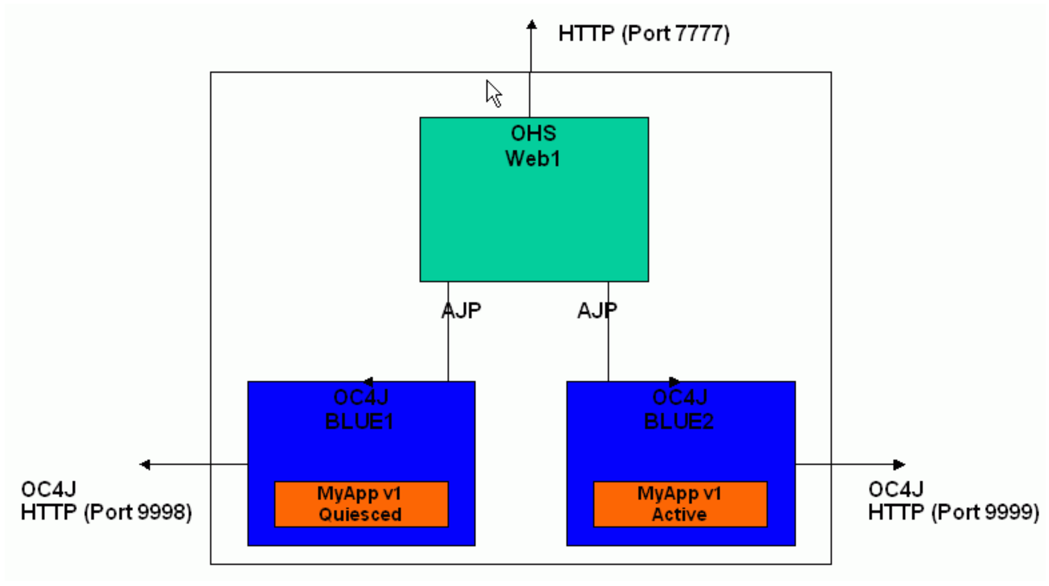
   b. On the OC4J page, click **Applications**.

   c. Click the application name.

   d. On the Application page, click **Stop**.

   Stopping an application begins to quiesce in-process requests as they complete and prevents any new traffic from being sent to the application instance, as Figure 9–3 shows.

Figure 9–3 Application Quiesced on First OC4J Instance



4. Wait two minutes for all in-process requests to be quiesced off of the application on the first OC4J instance.

After a short time, all the in-process requests will be gone, normally well within two minutes. You can make this wait more or less for your application.

5. Undeploy the application from the first OC4J instance:

a. From the Cluster Topology page of Application Server Control, click the name of the first OC4J instance.

b. On the OC4J page, click **Applications**.

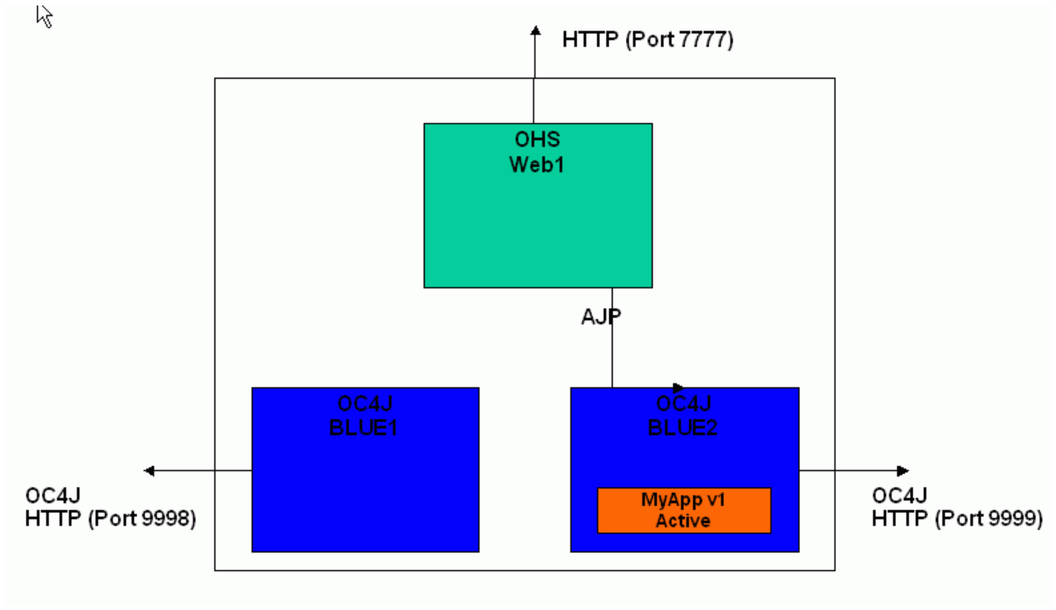
c. Click the application name.

d. On the Application page, click **Undeploy**.

e. Click **Return**.

The application is now undeployed from the first OC4J instance, as Figure 9–4 shows.

Figure 9–4 Application Undeployed from First OC4J Instance



Note: **Redeploy** is also a valid option for a side-by-side application upgrade, instead of **Undeploy** and **Deploy**, provided the HTTP Web site is selected for the redeployment. Redeploying the application enables you to maintain its existing settings. Undeploying the application and then deploying the new version removes the application, including its configuration, and then deploys it anew.

---

6. Deploy the new version of the application to the first OC4J instance, binding the application to the HTTP Web site to prevent OHS from routing any traffic to the new version:

   a. On the Applications tab of the OC4J page for the first OC4J instance, click **Deploy**.

   b. Specify your EAR file location and click **Next**.

   c. Specify your application name.

   d. Select the HTTP Web site from the drop-down menu.

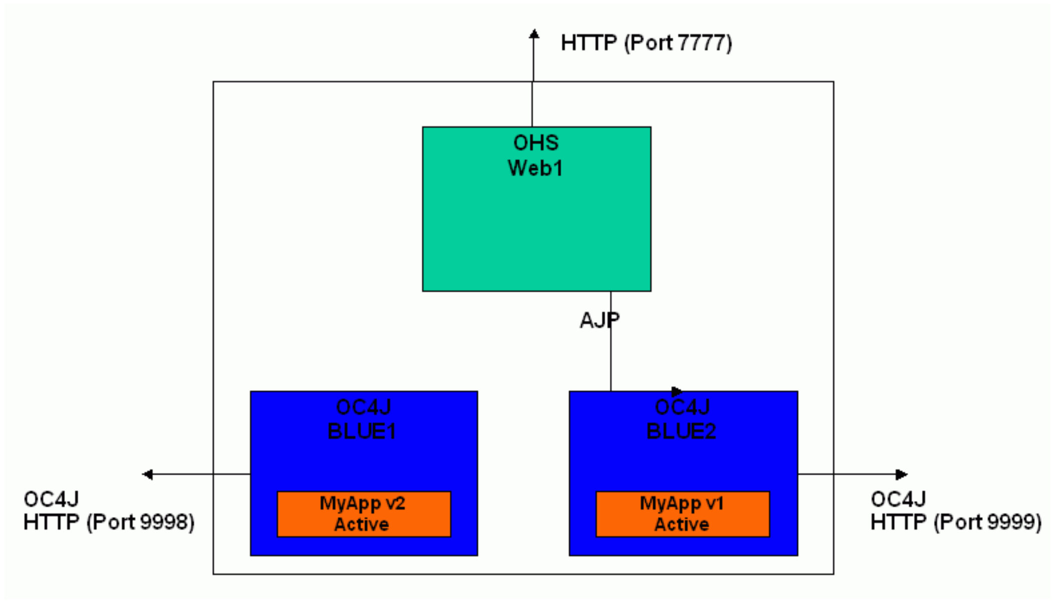
   e. Click **Next**.

   f. Click **Deploy**.

   g. Click **Return**.

   The new version of the application is now deployed to the first OC4J instance and exposed to the internal Web site. Figure 9–5 shows the state of the OC4J instances running two different versions of the application.

Figure 9–5 New Application Version Deployed to Internal Web Site on First OC4J Instance



7. Test the new version of the application on the first OC4J instance, through the HTTP listener, which enables you to get to the application internally.

   Make sure the application is behaving as you would expect it to:

   - The new version of the application is available from the internal Web site:

     ```
     http://host_name:port_number/application_name
     ```

   - The old version of the application is available from the external Web site:

     ```
     https://host_name:port_number/application_name
     ```

8. Bind the new version of the application to the AJP listener to enable the OHS server to route requests to it:

   a. Change your current directory to the home directory for the first OC4J instance; for example:

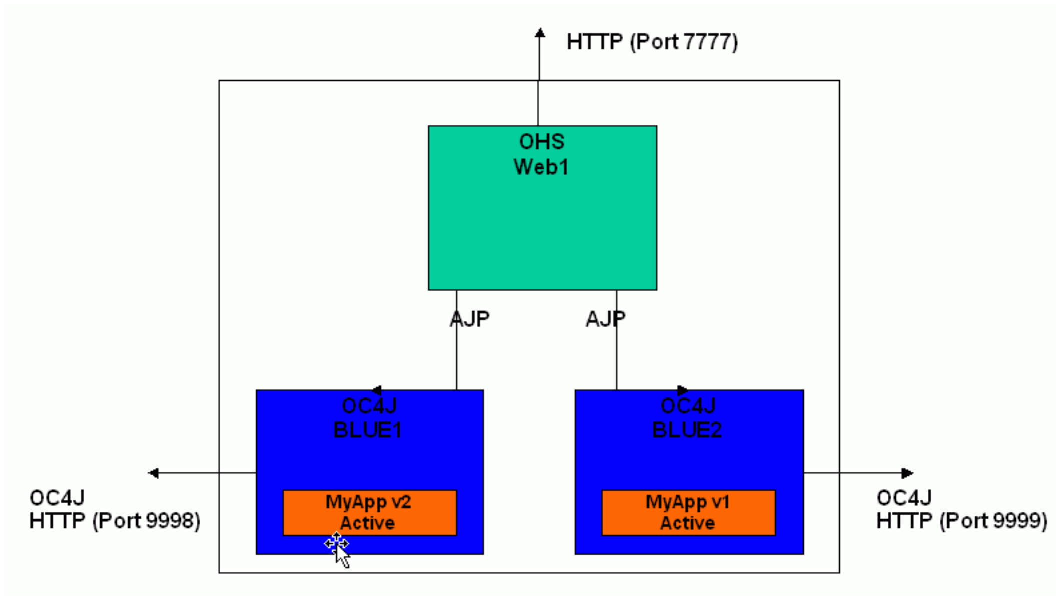
      ```
      CD c:\OracleAS\j2ee\blue1
      ```

   b. Issue an `admin_client.jar -bindWebApp` command like the following one:

      ```
      java -jar admin_client.jar deployer:oc4j:opmn://hostname/blue1 oc4jadmin
      password -bindWebApp -appName testApp -webModuleName testAppWeb
      -webSiteName default-web-site
      ```

   Figure 9–6 shows the new and old versions of the application running side by side, each on the production Web site in its OC4J instance.

Figure 9–6 New and Old Application Versions Running Side by Side on Production Web Sites



9. Repeat steps 3 through 8 to deploy and test the new version of the application on the second OC4J instance.

Figure 9–7 shows the old version of the application quiesced on the second OC4J instance.

Figure 9–7 Application Quiesced on Second OC4J Instance

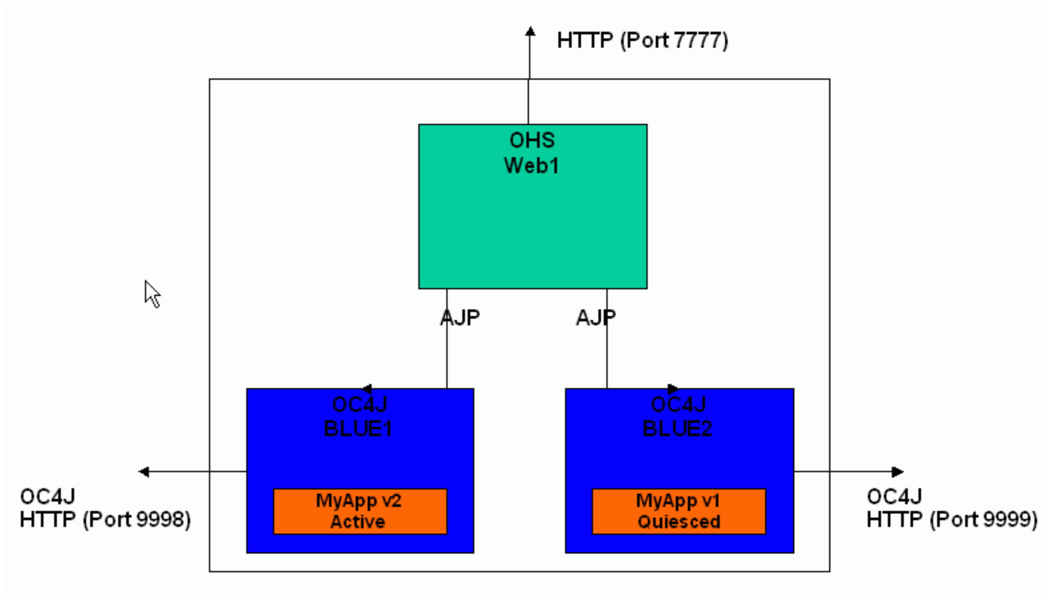


Figure 9–8 shows that the new version keeps running on the production Web site on the first OC4J instance while the old version of the application is undeployed from the second OC4J instance.

Figure 9–8 Old Application Version Undeployed from Second OC4J Instance

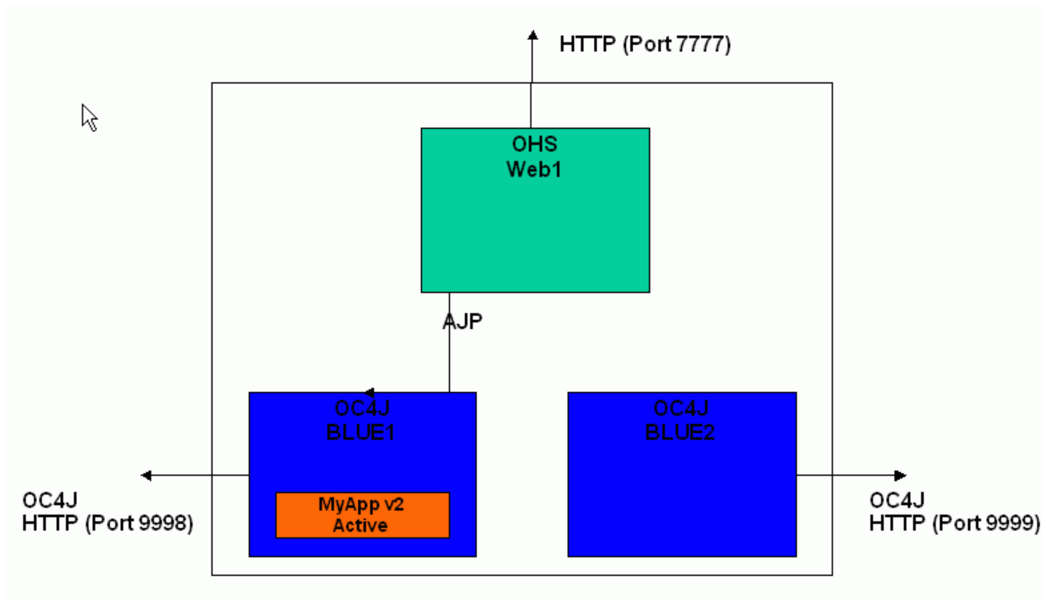
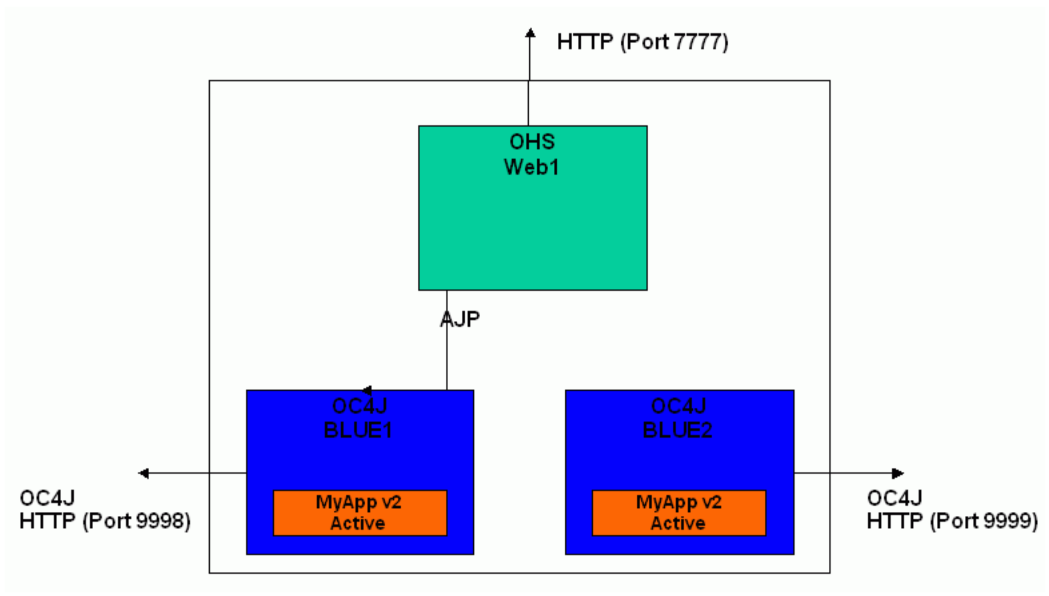


Figure 9–9 shows the new version of the application deployed to the HTTP Web site of the second OC4J instance for testing.

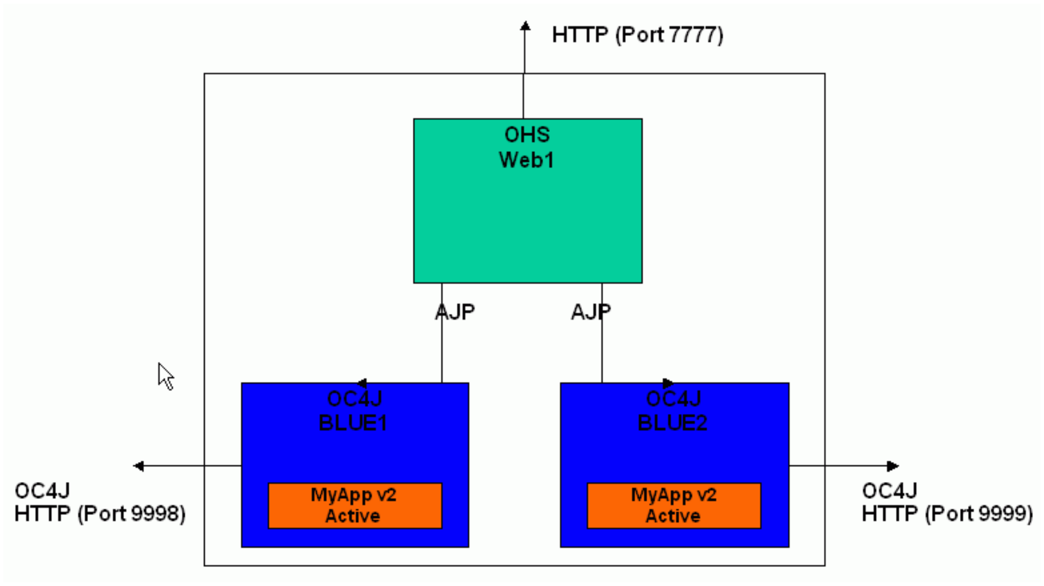
Figure 9–9 New Application Bound to Internal Web Site on Second OC4J Instance for Testing



10. After you bind the new version of the application to the AJP listener on the second OC4J instance, verify that the new version is running on the production Web site on both OC4J instances.

Figure 9–10 shows the state of the OC4J instances after deployment of an application with side-by-side application upgrade.

Figure 9–10 OC4J Instances Running New Application Version After Side-by-Side Application Upgrade



Undeploying an Application

Undeploying an application or module removes the code from the OC4J runtime and deletes all existing Web site bindings.

1. Click the **Applications** tab ->**Undeploy** button.
2. Select the application, then click the **Undeploy** button.

Creating and Managing Shared Libraries

You can create new shared libraries, add or remove archives from a shared library, and import other shared libraries into an existing shared library through Application Server Control.

To manage the shared libraries available in the OC4J instance:

1. Navigate to the OC4J Home page for the OC4J instance.
2. Click **Administration** to display the OC4J Administration page, which contains a table listing the various administration tasks you can perform for this OC4J instance.
3. If necessary, expand the Properties section of the table by clicking the expand icon or by clicking **Expand All**.
4. Click the task icon in the Shared Libraries row of the table.

   Application Server Control displays the Shared Libraries page, which lists the shared libraries currently available in the OC4J instance. You can also add and remove shared libraries from the OC4J instance.
5. Click **Help** for information about the options on the page.

For more information about creating and managing shared libraries through Application Server Control, see the online help topics.

Starting, Restarting, and Stopping Applications

To start, restart, or stop an application that has been deployed to the OC4J instance:

1. Navigate to the OC4J Home page for the OC4J instance.
2. Click **Applications**.
3. Select the application.
4. Click the **Start**, **Restart**, or **Stop** button.

Note: You can also start, stop, or restart an application from the Application Home page.

The following restrictions apply to starting, stopping, or restarting deployed applications:

- If you stop a parent application (such as the `default` application), then Application Server Control automatically stops any child applications that depend upon the parent application.
- If you start a child application, Enterprise Manager automatically starts the required parent application.
- Application Server Control restricts you from performing certain management tasks on the `ascontrol` application because this application represents Application Server Control, which you can use to manage your application server environment:
  - If you are managing one, standalone OC4J instance, then you cannot stop, start, or restart the `ascontrol` application. If you stopped this application, you would be unable to display or use Application Server Control.
  - If you are in clustered environment, where you are managing multiple OC4J instances, then you can use the Cluster Topology page to start, stop, or restart the `ascontrol` application, as well as the OC4J instances on which it is deployed. However, Application Server Control displays a warning that describes the implications of stopping the active `ascontrol` application.

    If you are using the Cluster Topology page to manage multiple OC4J instances, you will notice that each OC4J instance includes an `ascontrol` application. In most cases, only the active `ascontrol` application is up and running.
  - If you attempt to view the log files for a remote OC4J instance using the Log Viewer, Application Server Control checks to see if an `ascontrol` application is running in any OC4J instance within the remote Oracle Application Server instance. If no `ascontrol` application is running in that application server instance, Application Server Control displays a message stating that the remote `ascontrol` must be started.

    You can then choose to start the remote `ascontrol` application or cancel the operation. If you choose to start `ascontrol` from the Log Viewer, note that the remote `ascontrol` will not be configured to receive HTTP requests from Oracle HTTP Server. However, from the Cluster Topology page, you will see that the remote `ascontrol` is running. Later, when you are finished viewing and managing the remote log files with the Log Viewer, you can stop the remote `ascontrol` from the Cluster Topology page.

Restarting and Stopping OC4J Instances

With Application Server Control, you can restart or stop an OC4J instance or group of OC4J instances in an Oracle Application Server environment.

To restart or stop an OC4J instance with Application Server Control:

1. Navigate to the OC4J Home page.
2. To restart an OC4J instance that is in a stopped state, click the **Restart** button.
3. To stop an OC4J instance, click the **Stop** button.

You can also restart or stop an OC4J instance from the Cluster Topology page by selecting the instance in the Members section and clicking the **Restart** or **Stop** button.

To restart or stop a group of OC4J instances with Application Server Control:

1. In the Groups section of the Cluster Topology page, select the group.
2. To restart a group of OC4J instances that are in a stopped state, click the **Start** button.
3. To stop a group of OC4J instances, click the **Stop** button.

Managing Data Sources and Connection Pools for OC4J Instances

With Application Server Control, you can view, create, and delete data sources and connection pools (JDBC resources) for an OC4J instance or group of OC4J instances in an Oracle Application Server environment.

To manage data sources and connection pools for an OC4J instance with Application Server Control:

1. Navigate to the OC4J Home page for the OC4J instance.
2. Click **Administration** to display the OC4J Administration page, which contains a table listing the various administration tasks you can perform for the OC4J instance.
3. If necessary, expand the Services section of the table by clicking the expand icon or by clicking **Expand All**.
4. Click the task icon in the JDBC Resources row of the table.

   Application Server Control displays the JDBC Resources page, which lists the data sources and connection pools currently available in the OC4J instance.
5. Use the drop-down menu to view the data sources and connection pools specific to a particular deployed application.
6. Click **Help** for more information about viewing, creating, and deleting data sources and connection pools.

To manage data sources and connection pools for a group of OC4J instances with Application Server Control:

1. Navigate to the Cluster Topology page.
2. Scroll to the Groups section of the page and click the name of the group you want configure.
3. Click **Administration** to display the Group Administration page, which contains a table listing the various administration tasks you can perform for the group.

4. If necessary, expand the Services section of the table by clicking the expand icon or by clicking **Expand All**.
5. Click the task icon in the JDBC Resources row of the table.

   Application Server Control displays the JDBC Resources page, which lists the data sources and connection pools currently available for the OC4J instances in the group. Use the drop-down menu to view the data sources and connection pools specific to a particular deployed application.
6. Click **Help** for more information about viewing, creating, and deleting data sources and connection pools.

Managing JMS Resources

To manage JMS destinations through Application Server Control:

1. Navigate to the OC4J Home page for the OC4J instance.
2. Click **Administration** to display the OC4J Administration page, which contains a table listing the various administration tasks you can perform for this OC4J instance.
3. If necessary, expand the Services section of the table by clicking the expand icon or by clicking **Expand All**.
4. Click the task icon in the JMS Destinations row of the table.

   Application Server Control displays the JMS Destinations page, which lists the JMS Destinations available for the OC4J instance. You can then create additional JMS destinations or delete existing JMS destinations.

For more information about managing JMS resources with Application Server Control, see the online help topics.

10

Using OC4J Ant Tasks for Deployment

OC4J provides a set of Ant tasks for performing deployment-related 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.

This chapter describes the Ant tasks and provides guidelines for integrating them into your application build process. With OC4J Ant tasks, you can perform the following operations on an OC4J instance or group of OC4J instances:

- Adding Web sites
- Deploy an application (EAR), a standalone Web module (WAR), a standalone EJB module (EJB JAR), or a 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

Table 10–1 lists the OC4J ant tasks with references to their descriptions.

Table 10–1 OC4J Ant Tasks

Ant Task Name	Description
addDataSourceConnectionPool	"Adding a Data Source Connection Pool" on page 10-33
addDestination	"Adding a JMS Destination" on page 10-43
addWebSite	"Adding Web Sites" on page 10-10
addImportSharedLibrary	"Importing an Existing Shared Library" on page 10-27
addJMSConnectionFactory	"Adding a JMS Connection Factory" on page 10-42
addManagedDataSource	"Adding a Managed Data Source" on page 10-36
addNativeDataSource	"Adding a Native Data Source" on page 10-38
addRemoveInheritedSharedLibrary	"Stopping a Shared Library from being Inherited" on page 10-29

Table 10–1 (Cont.) OC4J Ant Tasks

Ant Task Name	Description
bindWebApp	"Binding a Specific Web Module to a Web Site and Setting the Context Root" on page 10-17
bindAllWebApps	"Binding All Web Modules to a Single Web Site" on page 10-16
compileJsp	"Using an Ant Task to Precompile a JSP" in the *Oracle Containers for J2EE Support for JavaServer Pages Developer's Guide*
deleteImportSharedLibrary	"Deleting an Imported Shared Library" on page 10-28
deleteRemoveInheritedSharedLibrary	"Allowing a Shared Library to be Inherited" on page 10-29
deploy	"Deploying a J2EE Application (EAR)" on page 10-11 "Deploying a Standalone Web Module (WAR)" on page 10-13 "Deploying a Standalone Resource Adapter (RAR)" on page 10-15
getDataSourcesDescriptor	"Getting the Data Sources Descriptor for an Application" on page 10-41
getDestinations	"Getting Information About JMS Destinations" on page 10-44
getJMSConnectionFactories	"Getting Information About JMS Connection Factories" on page 10-43
listApplications	"Listing Applications" on page 10-32
listDataSourceConnectionPools	"Listing Data Source Connection Pools" on page 10-35
listDataSources	"Listing Data Sources" on page 10-40
listWebBindings	"Listing Web Bindings" on page 10-20
modifySharedLibrary	"Modifying an Existing Shared Library" on page 10-25
publishSharedLibrary	"Installing a Shared Library" on page 10-24
redeploy	"Redeploying an Archive" on page 10-20
removeDataSourceConnectionPool	"Removing a Data Source Connection Pool" on page 10-35
removeDestination	"Removing a JMS Destination" on page 10-44
removeJMSConnectionFactory	"Removing a JMS Connection Factory" on page 10-42
removeManagedDataSource	"Removing a Managed Data Source" on page 10-37
removeNativeDataSource	"Removing a Native Data Source" on page 10-38
removeSharedLibrary	"Removing a Shared Library" on page 10-27
restartApp	"Restarting Applications" on page 10-31
restartServer	"Restarting and Stopping OC4J Instances" on page 10-33
shutdownServer	"Restarting and Stopping OC4J Instances" on page 10-33
start	"Starting Applications" on page 10-30
stop	"Stopping Applications" on page 10-31
testDatabaseConnection	"Testing a Database Connection" on page 10-39

Table 10–1 (Cont.) OC4J Ant Tasks

Ant Task Name	Description
testDataSource	"Testing a Data Source" on page 10-40
testDataSourceConnectionPool	"Testing a Data Source Connection Pool" on page 10-34
unbindAllWebApps	"Unbinding All Web Modules" on page 10-18
unbindWebApp	"Unbinding a Specific Web Module" on page 10-19
undeploy	"Undeploying an Archive" on page 10-22
updateEJBModule	"Updating Modified Classes in a Deployed EJB Module" on page 10-23

You can perform similar deployment tasks with Application Server Control or the `admin_client.jar` command-line utility. Chapter 9, "Using Application Server Control for Deployment" describes how to use the Application Server Control for deployment. Chapter 11, "Using the admin_client.jar Utility for Deployment" explains how to use `admin_client.jar` for deployment tasks.

This chapter includes the following sections:

- Preparing to Use OC4J Ant Tasks
- 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

Note: The OC4J Ant tasks discussed in this chapter are intended to be used with Apache Ant version 1.6.5.

See the following link to access the most recent Apache Ant product documentation:

`http://ant.apache.org/manual/`

You can also use OC4J Ant tasks to deploy Web applications through Eclipse, as "Using Ant Tasks from the OC4J Administration Client with Eclipse" on page 13-2 describes.

Preparing to Use OC4J Ant Tasks

This section provides prerequisites and guidelines for using OC4J Ant tasks. It includes the following topics:

- Meeting Prerequisites for Using OC4J Ant Tasks
- Incorporating OC4J Ant Tasks into Your Environment
- Incorporating OC4J Ant Tasks Using Ant 1.6.5 Outside OC4J
- Incorporating OC4J Ant tasks Using Ant 1.6.5 with the Administrative Client Utility
- Setting the Deployer URI
- Enabling Java Logging

Meeting Prerequisites for Using OC4J Ant Tasks

The following prerequisites are required to use the deployment-related OC4J Ant tasks that this document describes:

- Ant version 1.6.5 or later

  Ant 1.6.5 is installed with OC4J in the *ORACLE_HOME*/`ant` directory structure.
- An `ORACLE_HOME` environment variable set to the OC4J installed directory
- A `JAVA_HOME` environment variable set to the location of the Java2 Standard Edition SDK

For information about setting these environment variables, see the *Oracle Containers for J2EE Configuration and Administration Guide*.

Incorporating OC4J Ant Tasks into Your Environment

The OC4j installation includes Ant 1.6.5 and the files for the OC4J Ant tasks. Before you can use the Ant tasks, you need to incorporate them into your environment.

The `ant-oracle.jar` file is installed by default within the *ORACLE_HOME*/`ant/lib` directory. The following Ant-related files are installed with OC4J in the *ORACLE_HOME*/`j2ee/utilities` directory:

- `ant-oracle-classes.jar`

  A JAR file containing the compiled Ant task classes
- A properties file, `ant-oracle.properties`, that you can edit to specify execution properties for the Ant tasks
- `ant-oracle.xml`

  An XML file that you can import into the Ant build file (`build.xml`) using the Ant `<import>` task. This is necessary only if `ant-oracle.jar` is not installed in the *ORACLE_HOME*/`ant/lib` directory.

Perform the following procedure to set up your build environment for using the Ant 1.6.5 implementation, which is installed with OC4J by default in *ORACLE_HOME*/`ant`:

1. Add *ORACLE_HOME*/`ant/bin` to the system PATH environment variable.
2. Declare the `oracle` namespace in the `<project>` element in the Ant build file (`build.xml`). The OC4J Ant tasks will be referenced in `build.xml` using this namespace. For example:

```
<project name="test" default="all" basedir="."
  xmlns:oracle="antlib:oracle">
```

3. (OPTIONAL) Copy the `ant-oracle.properties` file from the *`ORACLE_HOME`*`/j2ee/utilities` directory to the directory containing your build file (`build.xml`).

   Although you can modify the file in *`ORACLE_HOME`*`/j2ee/utilities` and reference it from your build scripts, it is better to maintain the original file as a template.

4. (OPTIONAL) Set the values for arguments to pass to the Ant tasks in the `ant-oracle.properties` file.

   The properties within the file are set to the OC4J default values. The file also reads in environment variable settings, such as `ORACLE_HOME` and `JAVA_HOME`. You can edit any of these properties as necessary to reflect the configuration of the target OC4J instance or instances.

5. (OPTIONAL) If you copied the `ant-oracle.properties` file to your build directory, you must reference it in the build script (`build.xml`). For example:

   ```
   <property file="ant-oracle.properties"/>
   ```

Incorporating OC4J Ant Tasks Using Ant 1.6.5 Outside OC4J

This section outlines the procedure for setting up your build environment to use the Ant 1.6.5 implementation outside OC4J.

1. Add *`ANT_HOME`*`/ant/bin` to the system PATH environment variable.

2. Set the ANT_HOME environment variable to point to your Ant installation and the JAVA_HOME environment variable to point to the location of the Java2 Standard Edition SDK.

   The common ANT installation directory is *`ORACLE_HOME`*`/ant`.

3. Declare the `oracle` namespace in the `<project>` element in the Ant build file (`build.xml`). The OC4J Ant tasks will be referenced in `build.xml` using this namespace.

   ```
   <project name="test" default="all" basedir="."
     xmlns:oracle="antlib:oracle">
   ```

4. Copy the `ant-oracle.properties` file from the *`ORACLE_HOME`*`/j2ee/utilities` directory to the directory containing your build file (`build.xml`).

   Although you can modify the file in *`ORACLE_HOME`*`/j2ee/utilities` and reference it from your build scripts, it is better to maintain the original file as a template.

5. Set the values for arguments to pass to the Ant tasks in the `ant-oracle.properties` file.

   The properties within the file are set to the OC4J default values. The file also reads in environment variable settings, such as for ORACLE_HOME and JAVA_HOME. You can edit any of these properties as necessary to reflect the configuration of the target OC4J instance or instances.

6. Copy the `ant-oracle.xml` file from the *`ORACLE_HOME`*`/j2ee/utilities` directory to the directory containing your build file (`build.xml`).

7. At the top level of your build file, add this `<import>` element:

   ```
   <import file="ant-oracle.xml"/>
   ```

Incorporating OC4J Ant tasks Using Ant 1.6.5 with the Administrative Client Utility

The Administrative Client Utility enables you to use OC4J Ant tasks for configuration and deployment.

To incorporate OC4J Ant tasks using Ant 1.6.5 with the Administrative Client Utility:

1. Download the `oc4j_admin_client_101350.zip` file from the Oracle Technology Network at

   `http://www.oracle.com/technology/software/products/ias/htdocs/utilsoft.html`

   For information about the Administrative Client Utility and how to use it, see "Downloading and Extracting the Remote Administration Client" on page 11-5.

2. Extract the contents of `oc4j_admin_client_101350.zip` into a local directory of your choice, such as `oc4j_admin_client`.

3. Copy the `*ORACLE_HOME*/ant/lib/ant-oracle.jar` file from an Oracle Application Server 10*g* (10.1.3.5.0) home directory to `*OC4J_ADMIN_CLIENT_DIR*\ant\lib`, in the local directory to which you extracted the contents of `oc4j_admin_client_101350.zip`.

4. Set the `ORACLE_HOME` environment variable to the `*OC4J_ADMIN_CLIENT_DIR*` directory.

5. Add `*ANT_HOME*/ant/bin` to the system `PATH` environment variable.

6. Set the `ANT_HOME` environment variable to point to your Ant installation and the `JAVA_HOME` environment variable to point to the location of the Java 2 Standard Edition SDK.

   The common ANT installation directory is `*ORACLE_HOME*/ant`.

7. Declare the `oracle` namespace in the `<project>` element of the Ant build file (`build.xml`)., as follows:

   ```
   <project name="test" default="all" basedir="."
     xmlns:oracle="antlib:oracle">
   ```

   References to the OC4J Ant tasks in `build.xml` will use this namespace.

8. Copy the `ant-oracle.properties` file from the `*ORACLE_HOME*/j2ee/utilities` directory to the directory containing your build file (`build.xml`).

   Although you can modify the file in `ORACLE_HOME/j2ee/utilities` and reference it from your build scripts, it is better to maintain the original file as a template.

9. Set the values for arguments to pass to the Ant tasks in the `ant-oracle.properties` file.

   The properties within this file are set to the OC4J default values. The file also reads in environment variable settings, such as for `ORACLE_HOME` and `JAVA_HOME`. You can edit any of these properties as necessary to reflect the configuration of the target OC4J instance or instances.

10. Copy the `ant-oracle.xml` file from the `*ORACLE_HOME*/j2ee/utilities` directory to the directory containing your build file (`build.xml`).

11. At the top level of your build file, add this `<import>` element:

```
<import file="ant-oracle.xml"/>
```

Setting the Deployer URI

The key attribute passed to an Ant task is `deployerUri`, which specifies the OC4J target for the task. The syntax for the URI varies depending on the target.

For the format of this URI, see the following topics:

- Invoking a Task on a Group of OC4J Instances
- Invoking a Task on a Specific OC4J Instance
- Invoking a Task on a Standalone OC4J Server

Invoking a Task on a Group of OC4J Instances

Use the following URI to specify all OC4J instances in a group as the deployment target. A **group** is a synchronized set OC4J instances that belong to the same cluster topology. For example, you could specify `default_group` as the target to perform a deployment operation simultaneously on all OC4J instances that belong to the default group (named `default_group`) in a cluster.

The URI utilizes the OPMN-based clustering framework. You need to supply only the host name and, optionally, an 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://host[:opmnPort]/groupName
```

For example:

```
deployer:cluster:opmn://node1/default_group
```

Table 10–2 URI Parameters for Targeting a Group

Parameter	Description
`rmis`	Optional. Include if the target utilizes ORMI over SSL, or ORMIS.
`host`	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 not specified, the default port, `6003`, will be used.
`groupName`	Required. The name of the group to which the target OC4J instances belong.

Invoking a Task on a Specific OC4J Instance

Use the following URI to target a specific OPMN-managed OC4J instance, including an instance within a cluster. In the prefix of the URI, `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:6015/instance2/oc4j_2
```

Table 10–3 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.
`opmnPort`	Optional. The OPMN request port, as specified in `opmn.xml`. If not 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.

Invoking a Task on a Standalone OC4J Server

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

If you are using RMI, the URI syntax is as follows:

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

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

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

For example:

```
deployer:oc4j:myserver:23791
deployer:oc4j:rmis:myserver:23943
```

Table 10–4 URI Parameters for Targeting Standalone OC4J

Parameter	Description
`rmis`	Required if the target utilizes ORMI over SSL, or ORMIS.
`host`	Required. The host name for the standalone OC4J server.
`rmiPort`	Required if RMI is 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.

Enabling Java Logging

You can enable Java logging to help troubleshoot errors that occur when running the Ant tasks. Log messages will be output to the console.

To enable logging:

1. Create an `ANT_OPTS` environment variable and set the value to `-Djava.util.logging.config.file=logging.properties` before running the Ant tasks.
2. 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*/ant/bin, you must include the path to the file in the ANT_OPTS environment variable.

You can set the value in the logging.properties file to one of the Java log-level values, which Table 10–5 describes.

Table 10–5 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 configuration-related messages or problems.
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
```

In OC4J 10*g* (10.1.3.5.0), you can set the log levels for loggers with Application Server Control, as follows:

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.

Invoking OC4J Ant Tasks

You invoke the deployment-related Ant tasks provided with OC4J through the build file (build.xml). Each task is specified in a <target> element in the build file, in a subelement formatted as <oracle:*taskName*. ... />. In the subelement, oracle is the namespace used to reference the OC4J Ant tasks.

The following sample build.xml file contains a single deploy task. This task will deploy the specified EAR to a standalone OC4J server.

```
<project name="test" default="deploy" basedir="." xmlns:oracle="antlib:oracle">
    <property name="lib.dir" value="/scratch//temp"/>
    <property name="app.name" value="hello-planet"/>
    <property name="deployer.uri" value="deployer:oc4j:localhost:23791"/>
    <property name="oc4j.admin.user" value="oc4jadmin"/>
    <property name="oc4j.admin.password" value="password"/>
    ...
```

```
  <target name="deploy-ear" depends="setup,check-oc4j-available>
   <echo message="-----> Deploying the application module deployment (ear) file"/>
   <oracle:deploy deployerUri="${deployer.uri}"
                  userid="${oc4j.admin.user}"
                  password="${oc4j.admin.password}"
                  file="${lib.dir}/${app.name}.ear"
                  deploymentName="${app.name}"
                  bindAllWebApps="default-web-site"
                  logfile="${log.dir}/deploy-ear.log"/>
  </target>
  ...
</project>
```

Adding Web Sites

You can use the `addWebSite` task 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," in the *Oracle Containers for J2EE Configuration and Administration Guide* for detailed information about OC4J Web sites and how to manually add Web sites.

Note: The `addWebSite` task cannot be used to create a Web site on multiple OC4J instances within a group.

The following example adds a secure HTTP Web site to the OC4J `home` instance in a clustered environment:

```
<oracle:addWebSite
deployerUri="deployer:oc4j:opmn://localhost:6003/home"
userId="oc4jadmin"
password="welcome"
webSiteName="test-web-site"
protocol="https" port="9443"
keystorePath="/tmp/testkeystore.jks"
keystorePassword="welcome"
logfile="${log.dir}/my.log" />
```

Table 10–6 addWebSite Task Attributes

Parameter	Description
`deployerUri`	Required.The URI specifying the deployment target.
`userid`	Required.The administrator user name for the target OC4J instance.
`password`	Required.The administrator password for the target OC4J instance.
`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.

Table 10–6 (Cont.) addWebSite Task Attributes

Parameter	Description
`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.
`logfile`	Optional. The log file to use for output.

Deploying an Archive

The following sections describe how to invoke the `deploy` task:

- Deploying a J2EE Application (EAR)
- Deploying a Standalone Web Module (WAR)
- Deploying a Standalone EJB Module (JAR)
- Deploying a Standalone Resource Adapter (RAR)

Deploying a J2EE Application (EAR)

Use the `deploy` task 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, to an OC4J instance or to a group of OC4J instances. A J2EE application's modules can be packaged or left in their directory structure as well. The following example shows the attributes typically supplied to deploy an EAR file:

```
<oracle:deploy
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
file="${lib.dir}/${app.name}.ear"
deploymentName="${app.name}"
bindAllWebApps="default-web-site"
deploymentPlan="localPath/filename"
logfile="${log.dir}/deploy-ear.log"/>
```

Table 10–7 summarizes the attributes that you can set for the `deploy` task when you deploy an EAR file.

Table 10–7 deploy task Attributes for EAR Deployment

Attribute	Description
`deployerUri`	Required.The URI specifying the deployment target.
`userid`	Required.The administrator user name for the target OC4J instance or group of instances.
`password`	Required.The administrator password for the target OC4J instance or group of instances.

Table 10–7 (Cont.) deploy task Attributes for EAR Deployment

Attribute	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.
`bindAllWebApps`	Optional. Binds all Web modules to the specified Web site. Specify the *`name`* portion of the *`name`*`_web-site.xml` file that configures the Web site.
`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.
`parent`	Optional. The parent application of this application. The default is the global, or `default`, application.
`targetPath`	Optional. The directory to deploy the EAR to. If not specified, the EAR is deployed to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications` directory by default. The deployed EAR file is also copied to this directory. Each successive deployment will cause this EAR file to be overwritten.
`deploymentDirectory`	Optional. The directory containing the OC4J-specific deployment descriptors and generated files, such as compiled JSP classes and EJB wrapper classes. The default directory is *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application-deployments/`.
`enableIIOP`	Optional. Specify to generate IIOP client stubs on the OC4J server. 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. 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`.

Table 10–7 (Cont.) deploy task Attributes for EAR Deployment

Attribute	Description
`iiopClientJarPath`	Optional. The path and file name of the JAR to output IIOP client stubs to. 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. 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. 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`.
`sequential`	Optional. Specify to deploy the archive to each OC4J instance in a group in sequence. 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. You can use the `sequentialDelay` attribute to specify a number of seconds between deployments, as described in "Specifying a Delay Between Sequential Redeployments in a Cluster" on page 10-21. If this attribute is not specified, the archive will be simultaneously deployed to all OC4J instances in the target group by default. This attribute is valid only in an Oracle Application Server environment. It is not valid for standalone OC4J.
`sequentialDelay`	Optional. Specifies a number of seconds between sequential deployments to different OC4J instances that are running an application cluster.
`logfile`	Optional. The path and file name for a log to be generated for the deployment.

Deploying a Standalone Web Module (WAR)

Use the `deploy` task to deploy a standalone Web module packaged in a WAR file to an OC4J instance or to a group of OC4J instances.

Note: The `deploy` task 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.

For example:

```
<oracle:deploy
deployerUri="${deployer.uri}"
```

```
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
file="${lib.dir}/${app.name}.war"
deploymentName="${app.name}"
contextRoot="/myapp"
bindAllWebApps="default-web-site"
logfile="${log.dir}/deploy-war.log"/>
```

Table 10–8 summarizes the WAR-specific attributes that you can set for the `deploy` task when you deploy a WAR file.

Table 10–8 deploy Task Attributes for Standalone WAR Deployment

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or group of instances.
`password`	Required. The administrator password for the target OC4J instance or group of instances.
`file`	Required. The path and file name of the archive to deploy.
`deploymentName`	Required. The user-defined module deployment name, used to identify the module within OC4J.
`bindAllWebApps`	Required. Binds the Web module to the specified Web site. Specify the *`name`* portion of the *`name`*`_web-site.xml` file that configures the Web site.
`contextRoot`	Required. The Web module's context root, which will be appended to the URL used to access the application through a Web browser. For example, if you supply `/petstore` as the context root, the module could be accessed with the following URL: `http://node1.company.com:7777/petstore`
`parent`	Optional. The parent application of this module. The default is the global, or `default`, application.
`targetPath`	Optional. The directory to deploy the archive to. If not specified, the archive is deployed to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications` directory by default. The deployed archive file is also copied to this directory. Each successive deployment will cause this file to be overwritten.
`deploymentDirectory`	Optional. The directory containing the OC4J-specific deployment descriptors and generated files, such as compiled JSP classes and EJB wrapper classes. The default directory is *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application-deployments`.
`logfile`	Optional. The path and name for a log file generated for the deployment.

Deploying a Standalone EJB Module (JAR)

Use the `deploy` task to deploy a standalone EJB module packaged as a JAR file. For example:

```
<oracle:deploy
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
```

```
password="${oc4j.admin.password}"
file="${lib.dir}/${app.name}.jar"
deploymentName="${app.name}"
logfile="${log.dir}/deploy-jar.log"/>
```

Table 10–9 summarizes the attributes that you can set for the `deploy` task when you deploy an EJB JAR file.

Table 10–9 deployTask Attributes for Standalone EJB JAR Deployment

Attribute	Description
`file`	Required. The path and file name of the archive to deploy.
`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. 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` task to deploy a standalone resource adapter packaged in an archive to an OC4J instance or to a group of OC4J instances. The following example shows the attributes typically supplied to deploy a standalone RAR file:

```
<oracle:deploy
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
file="${lib.dir}/${app.name}.rar"
deploymentName="${app.name}"
grantAllPermissions="true"
logfile="${log.dir}/deploy-rar.log"/>
```

Table 10–10 summarizes the attributes that you can set for the `deploy` task when you deploy a RAR file.

Table 10–10 deploy Task Attributes for Standalone RAR Deployment

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or group of instances.
`password`	Required. The administrator password for the target OC4J instance or group of instances.
`file`	Required. The path and file name of the archive to deploy.

Table 10–10 (Cont.) deploy Task Attributes for Standalone RAR Deployment

Attribute	Description
`deploymentName`	Required. The user-defined connector name, used to identify the connector within OC4J.
`grantAllPermissions`	Required if resource adapter needs runtime permissions. Include and set to `true` to grant all runtime permissions requested by the resource adapter, if required.
`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.
`nativeLibPath`	Optional. The path to the directory containing native libraries (such as DLLs) within the RAR file.
`logfile`	Optional. The path and name for a log file generated for the deployment.

Managing Web Bindings

You can use Ant tasks 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. The tasks 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 at the time an EAR file or WAR file is deployed using the `bindAllWebApps` attribute of the `deploy` task. However, if the `bindAllWebApps` attribute was not specified when the EAR or WAR 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` task to bind the Web modules within a previously deployed EAR to a specified Web site. For example:

```
<oracle:bindAllWebApps
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
deploymentName="${app.name}"
webSiteName="${oc4j.binding.module}"
shared="false"
loadOnStartup="true"
accessLog="true" />
```

Table 10–11 summarizes the attributes that you can set for the `bindAllWebApps` task.

Table 10–11 bindAllWebApps Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance.
`password`	Required. The administrator password for the target OC4J instance.
`deploymentName`	Required. The user-defined name of the application that the Web modules belong to, set when the application was deployed.
`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`. Web modules are bound to the default Web site (`default-web-site`) on the target OC4J instances if this attribute 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` task 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 specify the context root that will be used to access the Web module. For example:

```
<oracle:bindWebApp
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
deploymentName="${app.name}"
webModule="${web.name}"
webSiteName="${oc4j.binding.module}"
contextRoot="/${context.root}"
shared="false"
loadOnStartup="true"
accessLog="true" />
```

Table 10–12 summarizes the attributes that you can set for the `bindWebApp` task.

Table 10–12 bindWebApp Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance.
`password`	Required. The administrator password for the target OC4J instance.
`deploymentName`	Required. The user-defined name of the application the Web module belongs to, set when the application was deployed.

Table 10–12 (Cont.) bindWebApp Task Attributes

Attribute	Description
`webModule`	Required. The name of the Web module to be bound to the Web site. 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`. Web modules are bound to the default Web site (`default-web-site`) on the target OC4J instances if this attribute is not specified.
`contextRoot`	Required. The context root for the Web module, such as `/utility`. This will be appended to the URL used to access the application through a Web browser; for example `http://localhost:8888/utility`.
`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` task to remove all Web module bindings from a specific Web site in an OC4J instance or in a group of OC4J instances that are part of a cluster. For example:

```
<oracle:unbindAllWebApps
deployerUri="${connection_url}"
userId="{username}"
password="${password}"
deploymentName="${appname}"
webSiteName="default-web-site"
logFile="${log.dir}/my.log"
/>
```

Table 10–15 summarizes the attributes that you can set for the `unbindAllWebApps` task.

Table 10–13 unbindAllWebApps Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance.

Table 10–13 (Cont.) unbindAllWebApps Task Attributes

Parameter	Description
`password`	Required. The administrator password for the target OC4J instance.
`deploymentName`	Required. The user-defined name of the application the Web modules belong to, set when the application was deployed.
`webSiteName`	Optional. The Web site name from which the Web modules try to unbind. The name is the same as the Web site XML configuration file name. For example, `default-web-site`. Web modules are unbound from the default Web site (`default-web-site`) on the target OC4J instances if this attribute is not specified.
`logfile`	Optional. The log file to use for output.

Unbinding a Specific Web Module

Use the `unbindWebApp` task to remove a specific Web module binding from a Web site in an OC4J instance or in a group of OC4J instances that are part of a cluster For example:

```
<oracle:unbindWebApp
deployerUri="${connection_url}"
userId="{username}"
password="${password}"
deploymentName="hello"
webModuleName="hello-web"
webSiteName="default-web-site"
logFile="${log.dir}/my.log"
/>
```

Table 10–14 summarizes the attributes that you can set for the `unbindWebApp` task.

Table 10–14 unbindWebApp Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance.
`password`	Required. The administrator password for the target OC4J instance.
`deploymentName`	Required. The user-defined name of the application the Web modules belong to, set when the application was deployed.
`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 modules try to unbind. The name is the same as the Web site XML configuration file name. For example, `default-web-site`. The Web module is unbound from the default Web site (`default-web-site`) on the target OC4J instances if this attribute is not specified.
`logfile`	Optional. The log file to use for output.

Listing Web Bindings

Use the `listWebBindings` task 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` attribute, which is described below. For example:

```
<oracle:listWebBindings
deployerUri="${deployer.uri}"
userId="${oc4j.admin.user}"
password="${oc4j.admin.password}"
webSiteName="${oc4j.binding.module}"
verbose="true"
logfile="${log.dir}/my.log"
/>
```

Table 10–15 summarizes the attributes that you can set for the `listWebBindings` task.

Table 10–15 listWebBindings Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance.
`password`	Required. The administrator password for the target OC4J instance.
`webSiteName`	Optional. The Web site name for which to view all Web bindings. The 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.
`logfile`	Optional. The log file to use for output.

Redeploying an Archive

Use the `redeploy` task to redeploy a previously deployed archive to an OC4J instance or to a group of OC4J instances. The `isConnector="true"` attribute must be included if you are redeploying a standalone resource adapter (RAR). The previous version of the archive will be undeployed as part of this process. For example:

```
<oracle:redeploy
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
file="${lib.dir}/${app.name}.archiveType"
deploymentName="${app.name}"
keepsettings="true"
sequential="true"
logfile="${log.dir}/deploy-ear.log"/>
```

Table 10–16 summarizes the attributes that you can set for the `redeploy` task.

Table 10–16 redeploy Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or group of instances.
`password`	Required. The administrator password for the target OC4J instance or group of instances.
`file`	Required. The path and file name of the archive 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 a standalone RAR. Include and set to `true` if redeploying a standalone RAR.
`keepsettings`	Optional. If this attribute 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. If this attribute is not specified, values will be set to those in the deployment descriptors packaged with the archive.
`sequential`	Optional. Specify to redeploy the archive to each OC4J instance in a group of OC4J instances in sequence. The redeployment to each target OC4J instance must complete before redeployment begins on to the next target instance. Requests will not be routed to an OC4J instance while the archive is being redeployed to it. You can use the `sequentialDelay` attribute to specify a number of seconds between redeployments, as described in "Specifying a Delay Between Sequential Redeployments in a Cluster" on page 10-21. If this attribute is not included, the archive will be simultaneously deployed to all OC4J instances in the group by default. This option is valid only in a clustered environment. It is not valid for standalone OC4J.
`sequentialDelay`	Optional. Specifies a number of seconds between sequential redeployments to different OC4J instances that are running an application cluster.
`failureRecovery`	Optional. Enable recovery from a failed redeployment. The previous archive is redeployed if possible.
`logfile`	Optional. The path and name for a log file generated for the deployment.

Specifying a Delay Between Sequential Redeployments in a Cluster

When an application is redeployed to a group with the `sequential` attribute of the `redeploy` Ant task, the redeployment operation is serialized, with redeployment done to one OC4J instance at a time so that the application being deployed 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 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 `redeploy` task, even if you specify the `sequential` and `keepsettings` attributes.

In OC4J 10*g* (10.1.3.5.0), you can use the `sequentialDelay` attribute of the `redeploy` task 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 *`sequentialDelay`* attribute, 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.

An example of the `redeploy` Ant task with the `sequentialDelay` attribute follows:

```
<oracle:redeploy
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
file="${lib.dir}/${app.name}.archiveType"
deploymentName="${app.name}"
keepsettings="true"
sequential="true"
sequentialDelay="15"
logfile="${log.dir}/deploy-ear.log"/>
```

The `sequentialDelay` option also applies to the `deploy` Ant task.

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

Use the `undeploy` task to remove an application or module from an OC4J instance or from a group of OC4J instances. The `isConnector="true"` attribute must be included if you are undeploying a standalone resource adapter (RAR). For example:

```
<oracle:undeploy
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
deploymentName="${app.name}"
```

```
logfile="${log.dir}/filename.log"/>
```

Table 10–17 summarizes the attributes that you can set for the `undeploy` task.

Table 10–17 undeploy Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or group of instances.
`password`	Required. The administrator password for the target OC4J instance or group of instances.
`deploymentName`	Required. The user-defined name of the application or module to undeploy. This is the name set when the archive was deployed.
`isConnector`	Required for a standalone RAR. Include and set to `true` if undeploying a standalone RAR.
`logfile`	Optional. The path and name for a log file generated for the deployment.

Updating Modified Classes in a Deployed EJB Module

Use the `updateEJBModule` task to perform 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, without requiring redeployment of the entire module. For example:

```
<oracle:updateEJBModule
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
deploymentName="${app.name}"
ejbModuleName="${ejb.jar}"
file="${new.ejb.jar}"
logfile="${log.dir}/filename.log"/>
```

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" on page 3-4.

Table 10–18 summarizes the attributes that you can set for the `updateEJBModule` task.

Table 10–18 updateEJBModule Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or group of instances.

Table 10–18 (Cont.) updateEJBModule Task Attributes

Attribute	Description
`password`	Required. The administrator password for the target OC4J instance or group of instances.
`deploymentName`	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.
`logfile`	Optional. The path and name for a log file generated for the update.

Creating and Managing Shared Libraries

You can use Ant tasks 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
- 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

Use the `publishSharedLibrary` task to install a shared library in an OC4J instance or in a group of OC4J instances. Once installed, the shared library will be available for use by applications within each instance. For example:

```
<oracle:publishSharedLibrary
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
libraryName="name"
libraryVersion="version"
logfile="${log.dir}/filename.log">
  <oracle:uploadCodeSource path="path/file" />
  <oracle:addCodeSource path="path/file" />
  <oracle:sharedLibraryImport libraryname="name" min-version="version"
      max-version="version" />
</oracle:publishSharedLibrary>
```

The shared library binaries will be installed in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/shared-lib` directory within each OC4J instance. At the same time, a `<shared-library>` element declaring the shared library will be added to the `server.xml` file on each OC4J instance.

Include one element for each code source to upload or add. Do the same for each existing shared library to import.

- To upload a new code source to each OC4J server, specify the path and file name of the JAR or ZIP archive file to upload in a nested `<oracle:uploadCodeSource>` element. The path can be absolute or relative to the current working directory.
- To add a JAR or ZIP file that already exists on the server, specify the path and file name in an `<oracle:addCodeSource>` element. Specify an absolute or relative path pointing to the location of the existing file on each OC4J server. If a relative path is used, it will be interpreted as relative to *`ORACLE_HOME`*.
- To import an existing shared library into the new shared library, specify the shared library name as defined within the OC4J instance or instances in an `<oracle:sharedLibraryImport>` element. You can specify the minimum or maximum version, or both, of the library to import.

The following example uploads two JAR files to each target OC4J server:

```
<oracle:publishSharedLibrary
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
libraryName="acme.common"
libraryVersion="2.5"
logfile="${log.dir}/filename.log">
  <oracle:uploadCodeSource path="/acme/acme-apis.jar" />
  <oracle:uploadCodeSource path="/acme/acmeImpl.jar" />
</oracle:publishSharedLibrary>
```

Table 10–19 summarizes the attributes that you can set for the `publishSharedLibrary` task.

Table 10–19 publishSharedLibrary Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or group of instances.
`password`	Required. The administrator password for the target OC4J instance or group of instances.
`libraryName`	Required. The name of the shared library. In cases 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`.
`libraryVersion`	Required. The shared library version. This value should ideally reflect the code implementation version.
`parentName`	Optional. The name of the parent shared library, if applicable.
`parentVersion`	Optional. The parent shared library version, if applicable.
`logfile`	Optional. The path and name for a log file generated for the update.

Modifying an Existing Shared Library

Use the `modifySharedLibrary` task to make changes to an existing shared library installed in an OC4J instance or in a group of OC4J instances. For example:

```
<oracle:modifySharedLibrary
deployerUri="${deployer.uri}"
```

```
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
libraryName="name"
libraryVersion="version"
logfile="${log.dir}/filename.log">
  <oracle:uploadCodeSource path="path/file" />
  <oracle:removeCodeSource path="file" />
  <oracle:addCodeSource path="path/file" />
  <oracle:addImport libraryName="name" min-version="version"
      max-version="version" />
  <oracle:removeImport libraryname="name" min-version="version"
      max-version="version" />
</oracle:modifySharedLibrary>
```

Include one element for each code source to upload, add, or remove. Do the same for each existing shared library to import or remove.

- To upload a new code source to each OC4J server, specify the path and file name of the JAR or ZIP archive file to upload in a nested `<oracle:uploadCodeSource>` element. The path can be absolute or relative to the current working directory.
- To add a JAR or ZIP file that already exists on the server or servers, specify the path and file name in an `<oracle:addCodeSource>` element. Specify an absolute or relative path pointing to the location of the existing file on the OC4J server or servers. If a relative path is used, it will be interpreted as relative to *ORACLE_HOME*.
- Use `<oracle:removeCodeSource>` to remove an existing code source from the shared library. Specify the file name of the code source within the shared library to remove.
- To import an existing shared library into the shared library, specify the shared library name as defined within the OC4J instance or instances in an `<oracle:addImport>` element. You can optionally specify the minimum or maximum version, or both, of the library to import.
- To remove an imported shared library, use an `<oracle:removeImport>` element.

The following example removes a code source and an imported library from the target shared library:

```
<oracle:modifySharedLibrary
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
libraryName="acme.common"
libraryVersion="2.5"
logfile="${log.dir}/filename.log">
  <oracle:removeCodeSource path="acme-apis.jar" />
  <oracle:removeImport libraryName="foo" min-version="2.0"/>
</oracle:modifySharedLibrary>
```

Table 10–20 summarizes the attributes that you can set for the `modifySharedLibrary` task.

Table 10–20 modifySharedLibrary Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.

Table 10–20 (Cont.) modifySharedLibrary Task Attributes

Attribute	Description
`userid`	Required. The administrator user name for the target OC4J instance or group of instances.
`password`	Required. The administrator password for the target OC4J instance or group of instances.
`libraryName`	Required. The name of the shared library to modify.
`libraryVersion`	Required. The version of the shared library.
`logfile`	Optional. The path and name for a log file generated for the update.

Removing a Shared Library

Use the `removeSharedLibrary` task to remove a shared library from an OC4J instance or from a group of OC4J instances. For example:

```
<oracle:removeSharedLibrary
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
logfile="${log.dir}/filename.log"
libraryName="name"
libraryVersion="version"/>
```

Table 10–21 summarizes the attributes that you can set for the `removeSharedLibrary` task.

Table 10–21 removeSharedLibrary Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or group.
`password`	Required. The administrator password for the target OC4J instance.
`libraryName`	Required. The name of the shared library.
`libraryVersion`	Required. The version of the shared library.
`logfile`	Optional. The path and name for a log file generated for the removal.

Importing an Existing Shared Library

Use the `addImportSharedLibrary` task to import an existing shared library to an application's classloader. The task is equivalent to adding an `<import-shared-library>` element to an application's `orion-application.xml` descriptor. This task requires an application restart for the change to take effect. Refer to "Installing a Shared Library" on page 10-24 for instructions on installing a shared library. For example:

```
<oracle:addImportSharedLibrary
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
logfile="${log.dir}/filename.log"
```

```
appName="Myapp"
name="oracle.jdbc"
MinVersion="1.0"/>
```

Table 10–22 summarizes the attributes that you can set for the `addImportSharedLibrary` task.

Table 10–22 addImportSharedLibrary Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or instances.
`password`	Required. The administrator password for the target OC4J instance or instances.
`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.
`logfile`	Optional. The log file to use for output.

Deleting an Imported Shared Library

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

```
<oracle:deleteImportSharedLibrary
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
logfile="${log.dir}/filename.log"
appName="Myapp"
name="oracle.jdbc"/>
```

Table 10–23 summarizes the attributes that you can set for the `addImportSharedLibrary` task.

Table 10–23 deleteImportSharedLibrary Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or instances.
`password`	Required. The administrator password for the target OC4J instance or instances.
`appName`	Required. The name of the application, as defined at deployment time, from which the shared library is deleted.

Table 10–23 (Cont.) deleteImportSharedLibrary Task Attributes

Parameter	Description
`name`	Required. The name of the shared library to remove from the given application.
`logfile`	Optional. The log file to use for output.

Stopping a Shared Library from being Inherited

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

```
<oracle:addRemoveInheritedSharedLibrary
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
logfile="${log.dir}/filename.log"
appName="Myapp"
name="oracle.jdbc"/>
```

Table 10–24 summarizes the attributes that you can set for the `addRemoveInheritedSharedLibrary` task.

Table 10–24 addRemoveInheritedSharedLibrary Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or instances.
`password`	Required. The administrator password for the target OC4J instance or instances.
`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.
`logfile`	Optional. The log file to use for output.

Allowing a Shared Library to be Inherited

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

```
<oracle:deleteRemoveInheritedSharedLibrary
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
logfile="${log.dir}/filename.log"
appName="Myapp"
name="oracle.jdbc"/>
```

Table 10–25 summarizes the attributes that you can set for the `deleteRemoveInheritedSharedLibrary` task.

Table 10–25 deleteRemoveInheritedSharedLibrary Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or instances.
`password`	Required. The administrator password for the target OC4J instance or instances.
`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.
`logfile`	Optional. The log file to use for output.

Managing Application Lifecycle

You can use Ant tasks 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` task 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 following example starts the `ascontrol` application on `node2` within a cluster:

```
<oracle:start
deployerUri="deployer:oc4j:opmn://node2.company.com:6004/home"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
deploymentName="ascontrol"/>
```

Table 10–26 summarizes the attributes that you can set for the `start` task.

Table 10–26 -start Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or instances.
`password`	Required. The administrator password for the target OC4J instance or instances.
`deploymentName`	Required. The name of the application to start.

Stopping Applications

Use the `stop` task 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 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.

```
<oracle:stop
deployerUri="deployer:oc4j:opmn://node2.company.com:6004/home"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
deploymentName="ascontrol"
timeout="5"/>
```

Table 10–27 summarizes the attributes that you can set for the `stop` task.

Table 10–27 -stop Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or instances.
`password`	Required. The administrator password for the target OC4J instance or instances.
`deploymentName`	Required. The name of the application to stop.
`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 attribute 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`. The `graceful` attribute takes precedence over the `timeout` attribute if the value is set to `false`.

Restarting Applications

Use the `restartApp` task 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 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.

```
<oracle:restartApp
deployerUri="deployer:oc4j:opmn://node2.company.com:6004/home"
userid="${oc4j.admin.user}"
```

```
password="${oc4j.admin.password}"
deploymentName="ascontrol"
timeout="5"/>
```

Table 10–28 summarizes the attributes that you can set for the `restartApp` task.

Table 10–28 -restartApp Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or instances.
`password`	Required. The administrator password for the target OC4J instance or instances.
`deploymentName`	Required. The name of the application to restart.
`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 attribute 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`. The `graceful` attribute takes precedence over the `timeout` attribute if the value is set to `false`.

Listing Applications

Use the `listApplications` task 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` argument, which is described below. For example:

```
<oracle:listApplications
deployerUri="${deployer.uri}"
userId="${oc4j.admin.user}"
password="${oc4j.admin.password}"
verbose="true"
logfile="${log.dir}/my.log" />
```

Table 10–29 summarizes the attributes that you can set for the `listApplications` task.

Table 10–29 listApplications Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or instances.
`password`	Required. The administrator password for the target OC4J instance or instances.

Table 10–29 (Cont.) listApplications Task Attributes

Parameter	Description
`verbose`	Optional. Displays more details. The additional details include: application context root binding, routing enabled, group name, and state replication.
`logFile`	Optional. The log file to use for output.

Restarting and Stopping OC4J Instances

Use the `restartServer` or `shutdownServer` task to restart or stop a specific OC4J instance or a group of OC4J instances across an entire cluster. For example:

```
<oracle:restartServer|shutdownServer
deployerUri="${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"/>
```

Table 10–30 summarizes the attributes that you can set for the `restartServer` and `shutdownServer` tasks.

Table 10–30 restartServer and shutdownServer Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The administrator user name for the target OC4J instance or instances.
`password`	Required. The administrator password for the target OC4J instance or instances.

Managing Data Sources

You can use Ant tasks 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 Ant tasks 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` task 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. For example:

```
<oracle:addDataSourceConnectionPool
```

```
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default"
name="ScottConnectionPool"
factoryClass="oracle.jdbc.pool.OracleDataSource"
dbUser="scott"
dbPassword="tiger"
url="jdbc:oracle:thin:@localhost:1521:xe"/>
```

Table 10–31 summarizes the attributes that you can set for the `addDataSourceConnectionPool` task.

Table 10–31 addDataSourceConnectionPool Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`jndiLocation`	Required. The location to use to bind the new data source connection pool into JNDI.
`connectionPoolName`	Required. The fully qualified path of the connection factory implementation.
`dbUser`	Required. The default user name for the new data source connection pool.
`dbPassword`	Required. The default password for the new data source connection pool.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`applicationName`	Optional. The name of the application to deploy to.
`loginTimeout`	Optional. The login timeout for the new data source connection pool.
`txLevel`	Optional. The transaction level (`local` or `global`).
`dbSchema`	Optional. The database schema to use,
`manageLocalTransactions`	Optional. Indicates whether or not OC4J should manage local transactions. The default value is `true`.

Testing a Data Source Connection Pool

Use the `testDataSourceConnectionPool` task 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. For example:

```
<oracle:testDataSourceConnectionPool
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default"
connectionPoolName="ScottConnectionPool"
sqlStatement="select * from dual" />
```

Table 10–32 summarizes the attributes that you can set for the `testDataSourceConnectionPool` task.

Table 10–32 testDataSourceConnectionPool Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`connectionPoolName`	Required. The name of the connection pool.
`sqlStatement`	Required. The SQL statement to use to test the connection
`logfile`	Optional. The path and name for a log file generated for the deployment.
`applicationName`	Optional. The name of the application.
`user`	Optional. The user name to use.
`password`	Optional. The default password to use.

Listing Data Source Connection Pools

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

```
<oracle:listDataSourceConnectionPool
deployerUri=${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
applicationName="default"
logfile="${log.dir}/my.log" />
```

Table 10–33 summarizes the attributes that you can set for the `listDataSourceConnectionPool` task.

Table 10–33 listDataSourceConnectionPools Task Attributes

Parameters	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`-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.
`logfile`	Optional. The log file to use for output.

Removing a Data Source Connection Pool

Use the `removeDataSourceConnectionPool` task 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. For example:

```
<oracle:removeDataSourceConnectionPool
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default"
name="ScottConnectionPool"/>
```

Table 10–34 summarizes the attributes that you can set for the `removeDataSourceConnectionPool` task.

Table 10–34 removeDataSourceConnectionPool Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`name`	Required. The name of the connection pool.
`logfile`	Optional. The path and name for a log file generated for the removal.
`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 Ant tasks 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` task to add a managed data source for an application in an OC4J instance or in each OC4J instance of a group within a cluster. For example:

```
<oracle:addManagedDataSource
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default"
dataSourceName="ScottDataSource"
jndiLocation="jdbc/ScottDataSource"
connectionPoolName="ScottConnectionPool" />
```

Table 10–35 summarizes the attributes that you can set for the `addManagedDataSource` task.

Table 10–35 addManagedDataSource Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.

Table 10–35 (Cont.) addManagedDataSource Task Attributes

Attribute	Description
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`dataSourceName`	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.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`applicationName`	Optional. The name of the application for which to add the data source.
`dbUser`	Optional. The default user name 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`).
`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` task to remove a managed data source from an application in an OC4J instance or in each OC4J instance of a group within a cluster. For example:

```
<oracle:removeManagedDataSource
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default"
dataSourceName="ScottDataSource"/>
```

Table 10–36 summarizes the attributes that you can set for the `removeManagedDataSource` task.

Table 10–36 removeManagedDataSource Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`dataSourceName`	Required. The name of the data source to remove.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`applicationName`	Optional. The name of the application from which to remove the data source.

Adding a Native Data Source

Use the `addNativeDataSource` task to add a native data source for an application in an OC4J instance or in each OC4J instance of a group within a cluster. For example:

```
<oracle:addNativeDataSource
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default"
dataSourceName="ScottNativeDataSource"
dbUser="scott"
dbPassword="tiger"
jndiLocation="jdbc/ScottNativeDataSource"
loginTimeout="60"
dataSourceClass="oracle.jdbc.pool.OracleDataSource"
url="jdbc:oracle:thin:@localhost:1521:xe"       >
<oracle:nativeDataSourceProperty name="maxStatements" value="20"/>
<oracle:nativeDataSourceProperty name="implicitCachingEnabled" value="30"/>
</oracle:addNativeDataSource>
```

Table 10–37 summarizes the attributes that you can set for the `addNativeDataSource` task.

Table 10–37 addNativeDataSource Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`dataSourceName`	Required. The name of the new data source.
`jndiLocation`	Required. The location to use to bind the new data source into JNDI.
`dbUser`	Required. The default user for the new data source.
`dbPassword`	Required. The default password 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.
<nativeDataSourceProperty>	
name	Required. The name of a property for the new data source.
value	Required. The value of a property for the new data source.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`applicationName`	Optional. The name of the application for which to add the data source.
`loginTimeout`	Optional. The login timeout for the new data source.

Removing a Native Data Source

Use the `removeNativeDataSource` task to remove a native data source from an application in an OC4J instance or in each OC4J instance of a group within a cluster. For example:

```
<oracle:removeNativeDataSource
```

```
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default"
dataSourceName="ScottNativeDataSource"/>
```

Table 10–38 summarizes the attributes that you can set for the `removeNativeDataSource` task.

Table 10–38 removeNativeDataSource Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`dataSourceName`	Required. The name of the data source to remove.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`applicationName`	Optional. The name of the application from which to remove the data source.

Testing a Database Connection

Use the `testDatabaseConnection` task to test an application's connection to a database in an OC4J instance or in each OC4J instance of a group within a cluster. For example:

```
<oracle:testDatabaseConnection
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default"
sqlStatement="select * from dual"
factoryClass="oracle.jdbc.pool.OracleDataSource"
dbUser="scott"
dbPassword="tiger"
url="jdbc:oracle:thin:@localhost:1521:xe"/>
```

Table 10–39 summarizes the attributes that you can set for the `testDatabaseConnection` task.

Table 10–39 testDatabaseConnection Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`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`).
`dbUser`	Required. The default user name for the database.
`dbPassword`	Required. The default password for the database.

Table 10–39 (Cont.) testDatabaseConnection Task Attributes

Attribute	Description
`url`	Required. The URL to set on the JDBC factory.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`applicationName`	Optional. The name of the application for which to test the database connection.

Testing a Data Source

Use the `testDataSource` task 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. For example:

```
<oracle:testDataSource
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default"
dataSourceName="ScottDataSource"
sqlStatement="select * from dual" />
```

Table 10–40 summarizes the attributes that you can set for the `testDataSource` task.

Table 10–40 testDataSource Task Attributes

Attribute	Task
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`datasourceName`	Required. The data source to test.
`sqlStatement`	Required. The SQL statement to use to test the connection.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`applicationName`	Optional. The name of the application for which to test the data source.
`dbUser`	Optional. The default user name for the data source.
`dbPassword`	Optional. The default password for the data source.

Listing Data Sources

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

```
<oracle:listDataSources
deployerUri=${deployer.uri}"
userid="${oc4j.admin.user}"
password="${oc4j.admin.password}"
applicationName="default"
logfile="${log.dir}/my.log" />
```

Table 10–41 summarizes the attributes that you can set for the `listDataSource` task.

Table 10–41 listDataSources Task Attributes

Parameter	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`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.
`logfile`	Optional. The log file to use for output.

Getting the Data Sources Descriptor for an Application

Use the `getDataSourcesDescriptor` task to retrieve an application's data sources descriptor. For example:

```
<oracle:getDataSourcesDescriptor
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
applicationName="default" />
```

Table 10–42 summarizes the attributes that you can set for the `getDataSourcesDescriptor` task.

Table 10–42 getDataSourcesDescriptor Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`applicationName`	Optional. The name of the application to which the descriptor belongs.

Managing JMS Resources

You can use OC4J Ant tasks to manage data 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 Ant tasks 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` task to add a JMS connection factory to an OC4J instance or to each instance of a group within a cluster. For example:

```
<oracle:addJMSConnectionFactory
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
domain="Queue"
jndiLocation="jms/ExampleQueueCF" />
```

Table 10–43 summarizes the attributes that you can set for the `addJMSConnectionFactory` task.

Table 10–43 addJMSConnectionFactory Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`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.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`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).
`jmsUser`	Optional. The user name associated with this connection factory (defaults to `anonymous`).
`jmsPassword`	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 an XA connection factory (defaults to `false`).

Removing a JMS Connection Factory

Use the `removeJMSConnectionFactory` task to remove a JMS connection factory from an OC4J instance or instances. For example:

```
<oracle:removeJMSConnectionFactory
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
jndiLocation="jms/ExampleQueueCF" />
```

Table 10–44 summarizes the attributes that you can set for the `removeJMSConnectionFactory` task.

Table 10–44 removeJMSConnectionFactory Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`jndiLocation`	Required. The JNDI location of the connection factory to remove.
`logfile`	Optional. The path and name for a log file generated for the deployment.

Getting Information About JMS Connection Factories

Use the `getJMSConnectionFactories` task 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. For example:

```
<oracle:getJMSConnectionFactories
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1" />
```

Table 10–45 summarizes the attributes that you can set for the `getJMSConnectionFactories` task.

Table 10–45 getJMSConnectionFactories Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`logfile`	Optional. The path and name for a log file generated for the deployment.

Managing JMS Destinations

You can use Ant tasks 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` task to add a JMS destination. For example:

```
<oracle:addDestination
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
domain="Queue"
name="ExampleQueue"
jndiLocation="jms/ExampleQueue" />
```

Table 10–46 summarizes the attributes that you can set for the `addDestination` task.

Table 10–46 addDestination Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`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.
`logfile`	Optional. The path and name for a log file generated for the deployment.
`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` task to remove a JMS destination from an OC4J instance or from each OC4J instance of a group within a cluster. For example:

```
<oracle:removeDestination
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1"
jndiLocation="jms/ExampleQueue" />
```

Table 10–47 summarizes the attributes that you can set for the `removeDestination` task.

Table 10–47 removeDestination Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`name`	Required. The OC4J JMS provider-specific name of the destination to remove.
`logfile`	Optional. The path and name for a log file generated for the deployment.

Getting Information About JMS Destinations

Use the `getDestinations` task to return the attributes for each of the OC4J JMS destinations in an OC4J instance or in a group of OC4J instances within a cluster. For example:

```
<oracle:getDestinations
deployerUri="deployer:oc4j:localhost"
userid="oc4jadmin"
password="welcome1" />
```

Table 10–48 summarizes the attributes that you can set for the `getDestinations` task.

Table 10–48 getDestinations Task Attributes

Attribute	Description
`deployerUri`	Required. The URI specifying the deployment target.
`userid`	Required. The default user name to use to get connections.
`password`	Required. The default password to use to get connections.
`logfile`	Optional. The path and name for a log file generated for the deployment.

11 Using the admin_client.jar Utility for Deployment

OC4J provides a command-line utility, `admin_client.jar`, for performing 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 deployment 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 the OC4J Ant tasks. For more information, see Chapter 9, "Using Application Server Control for Deployment," or Chapter 10, "Using OC4J Ant Tasks for Deployment.".

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 10g 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:

- Deploying to a Group of OC4J Instances Within a Cluster
- Deploying to a Specific OC4J Instance
- Deploying to a Standalone OC4J Server
- Validating a URI

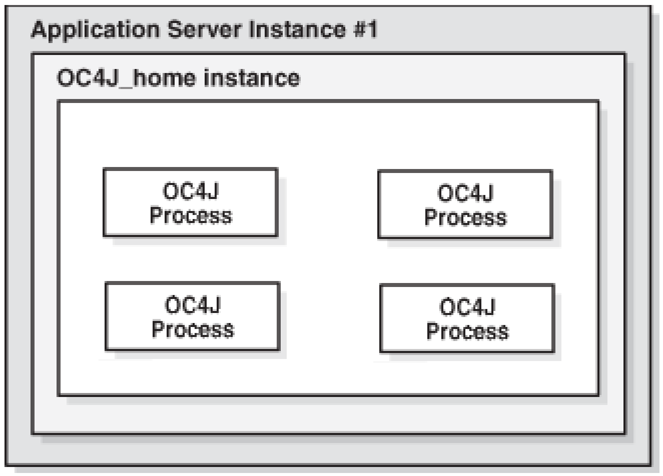
The OC4J administration user name and password are also passed to the `admin_client.jar` utility. The user name for the default administrator account is `oc4jadmin`.

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 11–1 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 11–1 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.

Deploying to a Group of OC4J Instances Within a Cluster

Use the following URI to specify all OC4J instances in a group as the deployment target. A **group** is a synchronized set of OC4J instances that belong to the same cluster topology. You can perform deployment operations simultaneously on all OC4J instances in the group. For example, you could specify `default_group` as the target to perform a deployment operation simultaneously on all OC4J instances that belong to the default group (named `default_group`) in a 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/default_group
```

Table 11–1 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.

Deploying to 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 11–2 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.

Deploying to 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 11–3 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 11–4.

Table 11–4 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," in the *Oracle Containers for J2EE Configuration and Administration Guide* 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 11–5 -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 application 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 11–6 -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 11–6 (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. 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. 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 11-15.
`-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. 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. 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. You can use the *`waitsec`* option to specify a number of seconds to wait between deployments, as follows: `-sequential 15` For more information about the *`waitsec`* option, see "Specifying a Delay Between Sequential Redeployments in a Cluster" on page 11-20. If this parameter is not specified, the archive will be simultaneously deployed to all OC4J instances in the group by default. This option is valid only in a clustered environment. It is not valid for standalone OC4J.

Table 11–6 (Cont.) -deploy Command Parameters for EAR Deployment

Parameter	Description
`-enableIIOP`	Optional. Specify this parameter to generate IIOP client stubs on the OC4J server. 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. 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. 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. 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. 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 11–7 -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. 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. 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. 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. For example, if you supply `/petstore` as the context root, the module could be accessed with the following URL: `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 11–8 -deploy Command Parameters for EJB JAR Deployment

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

Table 11–8 (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. 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 11–9 -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 Chapter 6, "Deploying Resource Adapters."

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 11–10 -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`. 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 appName -webModuleName moduleName -webSiteName siteName
-contextRoot contextRoot -shared true/false -loadOnStartup true/false -accessLog
true/false
```

Table 11–11 -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 *`siteName`* is the same as the Web site XML configuration file name. For example, `default-web-site`. Web modules are bound to the default Web site (`default-web-site`) on the target OC4J instances if this parameter is not specified.
`-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: `http://localhost:8888/petstore`. If a context root is not supplied, the context root specified in the parent application's `application.xml` deployment descriptor will be used.

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

Parameter	Description
`-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 11–12 -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 module tries to unbind. The *`web-site-name`* is the same as the Web site XML configuration file name. For example, `default-web-site`. 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 11–13 -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`. 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 11–14 -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 11–15 -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. 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. 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 11-15.
`-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. If this parameter is not specified, values will be set to those in the deployment descriptors packaged with the archive.
`-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. You can use the *`waitsec`* option to specify a number of seconds to wait between deployments, as follows: `-sequential 15` If this parameter is not specified, the archive will be simultaneously deployed to all OC4J instances in the group by default. 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.

In OC4J 10*g* (10.1.3.5.0), 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" on page 3-4.

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 11–16 -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 11–17 -publishSharedLibrary Command Parameters

Parameter	Description
`-name`	Required. The name of the shared library. 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. 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 11–18 -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. 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. 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 11–19 -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 target OC4J instance or instances. The syntax follows:

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

Table 11–20 -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 11-22 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 11–21 -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 11–22 -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 11–23 -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 11–24 -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 11–25 -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`. 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 11–26 -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.

Table 11–26 (Cont.) -restartApp Command Parameters

Parameter	Description
`-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`. 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 11–27 -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 11–28 -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 11–29 -testDataSourceConnectionPool Command Parameters

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

Table 11–29 (Cont.) -testDataSourceConnectionPool Command Parameters

Parameter	Description
`-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 11–30 -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 11–31 -removeDataSourceConnectionPool Command Parameters

Parameter	Description
`-name`	Required. The name of the connection pool.
`-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 11–32 -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`).
`-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 11–33 -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 11–34 -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.
`-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 11–35   -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 11–36   -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`).
`-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 11–37 -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 name to use to test the connection.
`-dbPassword`	Optional. The password 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 11–38 -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 11–39 -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 11–40 -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).
`-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 11–41 -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 11–42 -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 11–43 -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 11-5. 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 11-5, 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 11–44 shows examples of these URL patterns.

Table 11–44 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-D`*`name`*`=`*`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*.

12

Deploying to Standalone OC4J with admin.jar

Note: The `admin_client.jar` utility is the recommended command-line tool for deployment and management operations, and should be used in place of `admin.jar`.

This chapter provides instruction on using the `admin.jar` command-line utility provided with OC4J, which can be used to deploy or undeploy J2EE applications to a standalone OC4J instance. The following topics are covered:

- Understanding the admin.jar Syntax
- Deploying or Redeploying an Application
- Undeploying an Application
- Updating an EJB Module Within a Deployed Application
- Deploying or Redeploying a Standalone Connector
- Undeploying a Standalone Connector

Note that only those `admin.jar` options for deployment and undeployment are documented in this chapter. See the *Oracle Containers for J2EE Configuration and Administration Guide* for complete instructions on using the `admin.jar` utility.

Notes:

- `admin.jar` cannot be used to deploy 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`.

Understanding the admin.jar Syntax

The `admin.jar` utility uses the following syntax. The variables are described in Table 12–1.

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

For example, the following command will print the `admin.jar` help to the console. 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 -help
```

Table 12–1 Setting the Host and Login Information

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

Deploying or Redeploying an Application

The `-deploy` command is used to deploy or redeploy a J2EE application packaged in an EAR file to a standalone OC4J instance. OC4J must already be running before `admin.jar` can be used, except when you convert a `data-sources.xml` file before deployment.

1. Open a command console and change to the *J2EE_HOME* directory.
2. Deploy the archive to OC4J. The syntax is as follows:

```
java -jar admin.jar ormi://oc4jHost:oc4jOrmiPort adminId
    adminPassword -deploy -file path/filename
    -deploymentName appName [-targetPath path] [-parent appName]
   [-deploymentDirectory path] [-iiopClientJar path/filename]
```

   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
```

   The following table provides details on the `-deploy` parameters:

Table 12–2 -deploy Command Parameters

Parameter	Description
`-file`	Required. The path and file name of the EAR file to deploy.
`-deploymentName`	Required. The user-defined application deployment name, used to identify the application within OC4J.

Table 12–2 (Cont.) -deploy Command Parameters

Parameter	Description
`-targetPath`	Optional. The directory to deploy the EAR to. If not specified, the EAR is deployed to the *ORACLE_HOME*`/j2ee/home/applications` directory by default. 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 global or `default` application.
`-deploymentDirectory`	Optional. The directory containing the OC4J-specific deployment descriptors and generated files, such as compiled JSP classes and EJB wrapper classes. The default directory is *ORACLE_HOME*`/j2ee/home/application-deployments`.
`-iiopClientJar`	Optional. Include to generate IIOP stubs for the home, remote, and local interfaces packaged within each EJB JAR included in the EAR. 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/application-deployments`. The `GenerateIIOP` system property must be enabled at OC4J startup to use this feature. For example: `java -DGenerateIIOP=true -jar oc4j.jar`

3. Next, bind the application to the Web site that will be used to access it. The syntax is:

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

   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 admin password -bindwebapp utility
utility-web default-web-site /utility
```

Table 12–3 -bindWebApp Command Parameters

Parameter	Description
`appName`	The user-defined name of the application, which is the same name used for `-deploymentName` in the `-deploy` option.
`webAppName`	The name of the Web module. This should be the name of the WAR file contained within the EAR file, without the `.WAR` extension.
`webSiteName`	The name of the *name*`_web-site.xml` file that denotes the Web site that this Web application should be bound to.
`contextRoot`	The context root for the Web module. This will be appended to the URL used to access the application through a Web browser; for example, `http://localhost:8888/utility`.

Undeploying an Application

The following command removes an application from the OC4J runtime and removes bindings from any Web sites to which the application's Web modules were bound.

1. Open a command console and change to the *`ORACLE_HOME`* directory.
2. Undeploy the application. The syntax is as follows:

```
java -jar admin.jar ormi://oc4jHost:oc4jOrmiPort adminId adminpassword
   -undeploy appname
```

   - *`appName`* is the application name, which must match the value specified for `-deploymentName` on the `-deploy` option.

   Note: The optional `-keepFiles` parameter, which could be used to prevent files from being removed from the installed directories, has been deprecated. All files are now removed during undeployment.

   For example, the following undeploys the `utility` application:

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

Updating an EJB Module Within a Deployed Application

The `admin.jar` command-line utility provided with OC4J includes an `-updateEJBModule` option that allows an updated EJB JAR to be redeployed to an application running within an OC4J instance. Only those beans that have changed within the EJB JAR will be deployed.

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" on page 3-4.

This option is intended to be used by an application developer to redeploy the JAR file directly from a development environment. For more information on using `admin.jar`, see the *Oracle Containers for J2EE Configuration and Administration Guide*.

The syntax for `-updateEJBModule` follows:

```
java -jar admin.jar ormi://oc4j_host:oc4j_ormi_port admin_id
    admin_password -application appName -updateEJBModule ejbJarName
    [-file path/ejbJarName]
```

Usage notes:

- Specify the application the EJB is a component of as the value for *`appName`*. This name must match the name specified at deployment.
- If the updated EJB JAR file is in the working directory, and its location matches the relative module path defined in the application's `application.xml` J2EE deployment descriptor, you only need to specify the EJB JAR file name as the value for *`ejbJarName`*.
- If the updated EJB JAR is *not* in the working directory, or is in a subdirectory that does not match the relative module path defined in `application.xml`, specify the JAR file's location using the optional `-file` parameter.

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 at the root level of the `/dist` directory and the relative path defined in `application.xml` is "`customerEjb.jar`", the following command could be issued from the `/dist` directory:

```
java -jar $J2EE_HOME/admin.jar ormi://myoc4jserver:23791 oc4jadmin password
-application petstore -updateEJBModule customerEjb.jar
```

However, if the updated file is located within the `/build` directory, the optional `-file` parameter can be used to specify this location:

```
java -jar $J2EE_HOME/admin.jar ormi://myoc4jserver:23791 oc4jadmin password
-application petstore -updateEJBModule customerEjb.jar
-file build/customerEjb.jar
```

Notes:

- The example is based on the assumption that a `J2EE_HOME` environment variable pointing to the *`ORACLE_HOME`*`/j2ee/home` directory within the target OC4J host exists on the developer's machine.
- An error will occur if the EJB module name is missing or invalid, of if the updated EJB JAR cannot be found.

Deploying or Redeploying a Standalone Connector

Use the `-deployconnector` command to deploy or redeploy a standalone Java Connector Architecture-compliant resource adapter packaged in a RAR file. Redeploying a standalone resource adapter does not require a restart of the `default` application.

During redeployment of a resource adapter packaged within a RAR file, all existing application components will continue to obtain connection factories from the existing version of the resource adapter. New components, however, will obtain connection factories from the newly deployed resource adapter.

Existing JCA connections will remain open until closed by the application; new connections will be created from the original resource adapter instance.

The syntax is as follows:

```
java -jar admin.jar ormi://oc4jHost:oc4jOrmiPort adminId
    adminPassword -deployconnector -file path -name connectorName
    [-nativeLibPath path] [-grantAllPermissions]
```

Table 12–4 -deployconnector Command Parameters

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.

Table 12–4 (Cont.) -deployconnector Command Parameters

Parameter	**Description**
`-nativeLibPath`	Optional. The path to the directory containing native libraries (such as DLLs) within the RAR file.
`-grantAllPermissions`	Optional. Include to grant all runtime permissions requested by the resource adapter, if required.

Undeploying a Standalone Connector

Use `-undeployconnector` to remove a standalone connector from the OC4J runtime. Undeploying a standalone resource adapter does not require a restart of the `default` application.

The syntax is as follows. Note that the connector name must be supplied.

```
java -jar admin.jar ormi://oc4jHost:oc4jOrmiPort adminId
    adminPassword -undeployconnector -name connectorName
```

13

Deploying Web Applications from Eclipse

This chapter explains how to deploy a Web application to a standalone OC4J server directly from Eclipse 3.2 using the Web Tools Platform (WTP), which includes support for OC4J, and how to use OC4J Ant tasks for deployment through Eclipse, in these sections:

- Deploying a Web Application with the Web Tools Platform
- Using Ant Tasks from the OC4J Administration Client with Eclipse

Deploying a Web Application with the Web Tools Platform

This section provides an overview of how to create and deploy a Web application to OC4J using the Eclipse Web Tools Platform.

Connecting to OC4J from Eclipse

This section shows you how to connect to a standalone OC4J instance from Eclipse.

1. Launch Eclipse.
2. Open the J2EE perspective and select **Window > Open Perspective > Other** menu. Choose **J2EE** in the resulting dialog. This perspective includes the Servers view, which is used in the example deployment.
3. Select **Window > Show View > Console** menu. This enables you to see the server output.
4. Right click in the Servers View and select **New > Server** menu. This launches the New Server wizard.
5. In the Define a New Server panel, select **Oracle > Generic Oracle OC4J Standalone Server 10.1.3**. Click **Next**.
6. Set the following in the Define a New Generic Oracle OC4J Standalone 10.1.3 Runtime panel:
   - For JRE, select the JDK you are using.
   - For Oracle J2EE Home, browse to the `\j2ee\`*`instance`* subdirectory in the folder where you installed the Oracle Application Server.
   - Click **Next**.
7. Set the password to the password for the `oc4jadmin` administrator account you created during the OC4J installation in the Create a new Generic Oracle OC4J Standalone 10.1.3 server panel. Accept the defaults for all the other fields.
8. Click **Finish**.

Building a Web Application

Next, create a simple Web application to deploy to the OC4J server instance.

1. From the ProjectExplorer view, right-click on the **Dynamic Web Projects** folder and select **New Dynamic Web Project**.
2. Enter a name for the project.
3. Click the Show Advanced button. The Target runtime field is prepopulated with the entry for OC4J: Generic Oracle OC4J Standalone 10.1.3.
4. Accept the default values for all the fields in this dialog and click **Finish**.
5. Now you will create a JSP page within your new project. Open the **Dynamic Web Projects** folder.
6. Expand the project you created and then the **WebContent** folder.
7. Right-click on the **WebContent** folder and select the **New JSP** menu.
8. Enter a name for the file, such as `index.jsp`, and click **Finish**.
9. The file is opened in a JSP Editor. Enter this text between the <BODY> tags:

   ```
   <% out.print("Hello World!!"); %>
   ```
10. Save the file.

Deploying a Web Application

Once the Web application is ready, it can be deployed to OC4J directly from Eclipse.

1. Right-click on the JSP file, `index.jsp`, in the ProjectExplorer. Select Run As > Run on Server menu.
2. In the Run On Server dialog box, verify that the server **Oracle OC4J Standalone Server v10.1.3** is selected.
3. Click **Finish**. The Eclipse WTP tool will now do the following:
   - Package the Web application.
   - Start the OC4J server if it is not running.
   - Publish the application to the OC4J instance.
   - Launch the application in a browser.

     The console view displays the log tracking the progress of the deployment.

Using Ant Tasks from the OC4J Administration Client with Eclipse

You can use Ant tasks for deploying Web applications from Eclipse if you include `ant-oracle.jar` in the class path with the appropriate client libraries. Follow these steps to set up your environment for using Ant tasks with Eclipse:

1. If you do not have a local OC4J environment (either a standalone OC4J server or Oracle Application Server), then download or copy `oc4j_admin_client.zip` and extract the OC4J administration client, as "Downloading and Extracting the Remote Administration Client" on page 11-5 describes.
2. Copy the `ant-oracle.xml` and `ant-oracle.properties` file from the *`ORACLE_HOME`*`\j2ee\utilities` directory into the project.

3. Set the `ORACLE_HOME` environment variable to the location of the OC4J installation: OC4J, Oracle Application Server, or `oc4j_admin_client`.

   This can be done at startup, as a System variable, or from the Ant Runner mechanism in Eclipse.

For more information about OC4J Ant tasks, see Chapter 10, "Using OC4J Ant Tasks for Deployment".

14

Using Automatic Deployment in OC4J

This chapter discusses automatic deployment functionality in OC4J, which enables you to automatically reload only modified files within an application to an OC4J instance, rather than requiring that the entire application be redeployed.

This chapter includes the following topics:

- Overview of Automatic Deployment in OC4J
- Using Exploded-Directory Deployment for Application Development and Testing
- Using an Auto-Deployment Directory
- Using the Check-for-Updates Feature
- Forcing a One-Time Redeployment Using admin.jar

Overview of Automatic Deployment in OC4J

Automatic deployment, or *OC4J polling*, is a task management feature that automatically checks for changes made to currently deployed applications and modules, and reloads those files that have been modified. This functionality is a tremendous benefit for developers, eliminating the need to go through the deployment process every time code is updated.

By default, OC4J checks for files to deploy every second. This interval is configurable through the `taskmanager-granularity` attribute of the `<application-server>` element in the `server.xml` configuration file. See the *Oracle Containers for J2EE Configuration and Administration Guide* for details on task manager configuration.

Exploded-Directory Deployment

If you are developing J2EE applications with OC4J, you can use exploded-directory deployment for rapid development and test cycles. Exploded-directory deployment enables you to deploy an application to a set of distributed, expanded directories without packaging the application in an enterprise application archive (EAR) or Web application archive (WAR) file.

One exploded directory represents the application-level directory, and the rest are the modules to be deployed. You can place module directories under the application-level directory, or you can place them elsewhere and specify an alternate path to each module directory.

Redeployment of Updated Files As Needed

In addition to automatic polling, the `admin.jar` command-line utility includes an `-updateConfig` option that forces OC4J to check for updated files. You can use this feature in a production environment to check for and reload updated files on an as-needed basis. See "Forcing a One-Time Redeployment Using admin.jar" on page 14-10 for details on this feature.

When to Use Automatic Deployment

Automatic deployment is recommended only for standalone OC4J instances in a development environment. It is not recommended for use in production environments.

The reason is that the polling mechanism is invoked by the task manager on a regular schedule and uses system resources. In addition, automatic deployment carries the risk of putting OC4J in an inconsistent state, and errors may result if requests try to execute against OC4J.

Using Exploded-Directory Deployment for Application Development and Testing

Exploded-directory deployment, equivalent to deployment of a J2EE application packaged in an EAR file, facilitates application testing by supporting partial redeployment. When you deploy an EAR file, OC4J makes a copy of the EAR and then executes the application by accessing the files that are in that copy. In exploded-directory deployment, OC4J copies only the deployment plan and then executes the application by accessing the original application files in your file system.

You can deploy an application from its standard, fully expanded directory structure instead of a packaged application archive. With exploded-directory deployment, OC4J reads all Web content, class files, libraries, deployment descriptors, and so on directly from the file system. You can deploy all types of standard J2EE modules and applications in this manner, including Web modules, EJB modules, and full J2EE applications. An exploded directory must conform to the standard J2EE directory structure for applications and components.

You can specify the path to an application directory in the OC4J configuration file, `server.xml`. The modules of an application, however, do not need to be in subdirectories of the application-level directory. You can place the module directories elsewhere in the file system and specify an alternate path to each module in the `application.xml` deployment descriptor for the application. You can also specify the alternate paths in the OC4J application deployment descriptor, `orion-application.xml`.

Building and Deploying a J2EE Application from a Master Directory

While developing an application, you can modify, compile, and execute your classes quickly with exploded-directory deployment. OC4J automatically deploys your application as you develop it within an exploded-directory format.

This automatic deployment occurs whenever a timestamp changes in the application-level directory, which is a *master* directory, like *appname* in Figure 14–1. The subdirectories of the master directory, located within it or elsewhere in the file system, are application modules for OC4J to deploy.

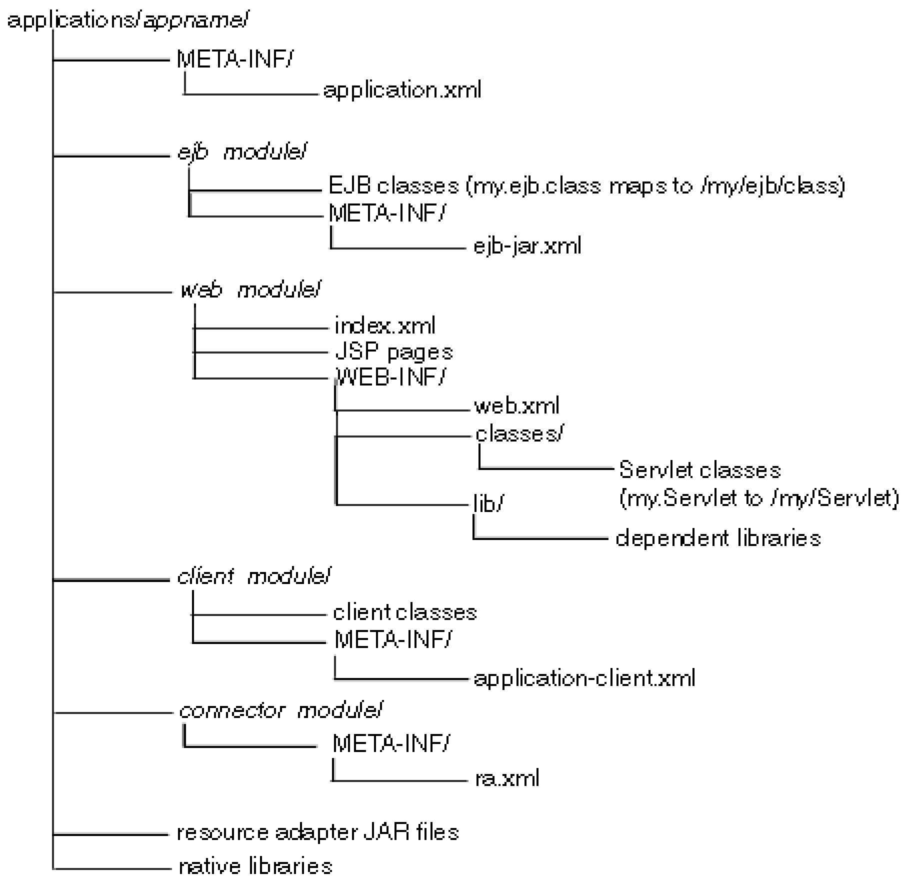
To perform an exploded-directory deployment, you need to modify OC4J configuration files for the initial deployment of the application. The location of the master directory is the value of the `path` attribute in the `server.xml` configuration file for OC4J. An alternate path for each J2EE module in `application.xml` and, optionally, in `orion-application.xml` enables OC4J to locate the module directories. The format of a path name is platform dependent. In a Linux or UNIX environment, the path name can contain forward slashes. In a Windows environment, the path name can contain back slashes. Some examples of path names follow:

```
D:\sharedfolder\My resource

/scratch/sharedfolder/my resource
```

For exploded-directory deployment, the application's master directory and module directories must have the same hierarchical format as required for an EAR, WAR, or JAR file. A set of directory paths represents the application, one path for the master directory and one or more alternate paths, each pointing to the location of an application module. Figure 14–1 shows the directory structure for a J2EE application that has the master directory *appname*. The directory structure under *appname* is similar to the structure within an EAR file.

Figure 14–1 Application Directory Structure



To build and deploy a J2EE application from a master directory:

1. Place the files in any directory or set of directories, as follows:

   a. Replace each EJB JAR, WAR, client JAR, and RAR file name with a directory name of your choice to represent a separate module.

      Figure 14–1 represents these directory names with `ejb_module/`, `web_module/`, `client_module/`, and `connector_module/`.

   b. Place the classes for each module within the appropriate directory structure that maps to their package structure.

2. Edit the `server.xml` file and add a new `<application>` element that specifies the path to the exploded directory that is to be deployed.

   For example:

```
  <application name="dwp" path="file://d:/eclipse/workspace/dwp/test-app"/>
```

3. List the absolute or relative paths of J2EE modules in `application.xml` and, optionally, in `orion-application.xml`.

   In the `j2ee/home/applications/`*`appname`*`/META-INF/application.xml` file, modify the `<web-uri>`, `<ejb>`, and `<client>` elements within `<module>` elements to designate the directory path name for each module (not the JAR or WAR file name). To designate the directories where these modules exist, the path name in each of these elements should be relative to the master directory and should be the parent of the WEB-INF or META-INF directories in each of these application types.

   If `application.xml` and `orion-application.xml` refer to the same Java EE module, the key is the path specification (absolute or relative, directory URL or archive file URL). The key must be identical in both files, if they are included.

   For example, the following `<web-uri>` element designates `myapp-web/` as the Web module directory within the `<web>` and `<module>` elements:

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

> **Note:** In OC4J 10*g* (10.1.3.5.0), the `system` application, represented by the `j2ee/home/config/system-application.xml` file, is the ultimate parent of all applications. Oracle recommends, however, that you deploy Web modules to the `default` application, represented by `j2ee/home/config/application.xml`.

4. In the `j2ee/home/config/default-web-site.xml` file, add a new `<web-app>` element for each Web application that is in the exploded directory.

   For example:

```
<web-app application="dwp" module="web" context-root="/dwp"/>
```

   The `<web-app>` element binds the application to a Web site. The value of the `application` attribute of `<web-app>` should be the same as the application name in the `server.xml` file. The value of the `name` attribute should be the directory path to the Web application, like the path in the `<web-uri>` element in the `application.xml` file.

5. If the application being deployed is a standalone Web application in exploded WAR format, you can deploy the application as a standalone Web module to the `default` application:

   a. Edit the `application.xml` file and add a new `<web-module>` element that specifies the path to the exploded directory that is to be deployed, as follows:

   ```
   <web-module id="dwp" path="file://d:/eclipse/workspace/dwp/web"/>
   ```

   b. Edit the `default-web-site.xml` file and add a new `<web-app>` element for the Web module, as follows:

   ```
   <web-app application="default" module="dwp" root="/dwp"/>
   ```

When OC4J starts, the application and Web modules will be deployed from the specified exploded-directory paths.

Cloning an Application Module to an Exploded Directory

It is possible to create an entire exploded application from an application module by cloning the `application.xml` file to an `exploded-application.xml` file, removing the unneeded entries, and then adding the application to server.xml. This keeps your J2EE module outside of the `default` application and within its own application that can be started, stopped, and managed.

To clone an application to an exploded directory:

1. Copy the `application.xml` file to `exploded-application.xml` in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` directory:

   ```
   >cd j2ee/home/config
   >cp application.xml exploded-application.xml
   ```

2. Edit `server.xml` and add the following element:

   ```
   <application name="exploded" path="exploded-application.xml" parent="default"
    start="true" />
   ```

3. Edit `exploded-application.xml`, remove the unneeded application modules (that is, most of them), and then add an entry for your Web module, like this:

   ```
   <web-module id="exp" path="d:/temp/exp/how-to-cluster-web" />
   ```

4. Bind the exploded Web module to the `exploded` Web site:

   ```
   <web-app application="exploded" name="exp" load-on-startup="true"
   context-root="/exp" />
   ```

   Instead of the `default` Web site, you use `exploded` because that is the name of the new containing application.

Reloading Modified Classes from an Exploded Directory

If you are using exploded-directory deployment and change a JavaServer Pages (JSP) module in the exploded-directory structure, the default OC4J settings cause it to notice the change and recompile the page. The implicit default setting, `with main_mode`, should cause OC4J to pick up any changes to JSP modules.

If a changed JSP does not get recompiled, try forcing the browser to not load the page from its cache by using **`SHIFT`** plus **`RELOAD`**.

If you change any compiled Java class that the JSP module uses, such as a Web bean, DTO, or DAO, OC4J does not pick up the changed class with the JSP module.

To reload modified classes from an exploded directory:

- Set the `check-for-updates` attribute to `all` in the <application-server> element of `server.xml`:

  ```
  <application-server
      application-directory="../applications"
      check-for-updates="all"
      deployment-directory="../application-deployments"
      connector-directory="../connectors"
      schema-major-version="10" schema-minor-version="0" >
  ```

  Any class changes you make in the exploded directory should then be picked up.

  For more information about the `check-for-updates` attribute, see "Using the Check-for-Updates Feature" on page 14-7.

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

Using an Auto-Deployment Directory

Automatic deployment can be initiated by dropping an EAR file into a designated *`auto-deployment`* directory within the OC4J instance. This feature should be used only in a standalone OC4J development environment.

The directory must be created on the server hosting the OC4J instance; it is not created by OC4J. An existing directory within OC4J, such as *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications`, can also be used.

The location of the directory must then be specified in the `application-auto-deploy-directory` attribute, which must be added to the root `<application-server>` element in *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config/server.xml`.

The following `server.xml` entry sets *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications` as the auto-deployment directory:

```
<application-server ...
 application-directory="../applications"
 check-for-updates="adminClientOnly"
 deployment-directory="../application-deployments"
 application-auto-deploy-directory="../applications">
...
</application-server>
```

> **Note:** If the check-for-updates feature is not enabled, OC4J must be restarted for configuration changes made in `server.xml` to take effect.

Once configured, OC4J will poll the directory for new or updated EAR files every time the task manager is executed. The server compares the timestamp on an EAR file to determine if a redeployment should be initiated. If it should, the EAR will be deployed automatically. Any Web modules packaged as WAR files within the EAR will be bound automatically to the `default` Web site.

The auto-deployment directory feature is completely independent of OC4J polling. Archives dropped in this directory will be deployed regardless of whether OC4J polling is enabled or disabled.

Using the Check-for-Updates Feature

The *check-for-updates* feature enables you to redeploy files to an OC4J instance, as the following topics describe:

- Enabling or Disabling Check for Updates
- Redeploying Configuration Files, Deployment Descriptors, and WAR Files Automatically
- Impact of Redeploying a Modified Configuration File Automatically

Note: An EAR or WAR file copied to the `ORACLE_HOME/j2ee/instance/applications` directory will be deployed or redeployed by default upon OC4J startup, regardless of whether auto-deployment is enabled.

The EAR or WAR file is also deployed when its timestamp is newer than the timestamp of the directory that contains the file.

Enabling or Disabling Check for Updates

You can enable the check-for-updates feature through one of the following methods:

- The `check-for-updates` attribute of the root `<application-server>` element in `ORACLE_HOME/j2ee/instance/config/server.xml`. For example:

```
<application-server ... check-for-updates="all" ... />
```

- Setting the `checkForUpdates` system property on the `oc4j.jar` command line. For example:

```
java -DcheckForUpdates=all -jar oc4j.jar
```

Notes: The following notes apply to the `checkForUpdates` system property:

- All system properties are prefaced on the command line with a `-D`.
- The value set for this property overrides the value set in `server.xml`.

Table 14–1 contains the values that can be set for `checkForUpdates` using either option.

Table 14–1 Valid Values for checkForUpdates

Value	Description
`all`	Enables OC4J polling, which starts automatically at the interval specified in the OC4J task manager configuration. The default interval is every 1 second. This option should not be used in Oracle Application Server or production environments. This value also allows the `-updateConfig` option to be passed on the `admin.jar` command line, which forces OC4J to perform a one-time check for updated files. See "Forcing a One-Time Redeployment Using admin.jar" on page 14-10 for details on this feature.
`adminClientOnly`	This is the default value set in both standalone OC4J and Oracle Application Server installations. It allows the `-updateConfig` option to be passed on the `admin.jar` command line, which forces OC4J to do a one-time check for updated files, and reload any that have changed. See "Forcing a One-Time Redeployment Using admin.jar" on page 14-10 for details on this feature.
`none`	Completely disables OC4J polling, including the `-updateConfig` option.

Redeploying Configuration Files, Deployment Descriptors, and WAR Files Automatically

The following files can be automatically redeployed to an OC4J instance:

- Modified OC4J-specific XML configuration files in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` directory, including `server.xml`.
- Modified deployment descriptors packaged in an updated EAR file copied to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications` directory.
- The following files packaged within an updated WAR file. The WAR can be either packaged in an EAR file copied to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications/` directory, or copied directly to the Web module's *`ORACLE_HOME`*`/j2ee/`*`instance`*`/applications/`*`webAppName`* directory.
  - Modified deployment descriptors
  - Updated files in the `WEB-INF/lib/` or `WEB-INF/classes/*` paths within the WAR
  - Updated JSP tag library (TLD) files

Note: This feature does not currently provide automatic detection of EJB-related or data-source-related configuration changes. This means, for example, that modified files in an EJB JAR file will not be automatically redeployed. OC4J must be restarted to detect such configuration changes and apply them appropriately.

Impact of Redeploying a Modified Configuration File Automatically

The following tables describe the impact of modifying or updating various files when `checkForUpdates` is set to `all`, indicating that the feature is enabled. See "Enabling or Disabling Check for Updates" on page 14-7 for instructions on enabling OC4J polling.

Table 14–2 describes the impact of modifying OC4J configuration files within the `ORACLE_HOME/j2ee/instance/config` directory in an OC4J instance.

Table 14–2 Impact of Modifying OC4J Configuration Files

Modified File	Action Initiated
`server.xml`	Modifying the OC4J server configuration file causes the OC4J server to be restarted.
`global-web-application.xml`	Modifying this file, used to configure OC4J servlet and JSP containers, forces all Web modules bound to all Web sites within the instance to be restarted.
`application.xml`	Modifying this file, which contains common settings for all applications in this OC4J instance, forces all deployed applications to be restarted.
`*-web-site.xml`	Modifying a Web site configuration file causes the Web site to be restarted. Incoming requests will not be serviced during the restart.

Impact of Redeploying a Modified Deployment Descriptor Automatically

Table 14–3 describes the impact of modifying one or more deployment descriptors within an updated EAR file copied to the `ORACLE_HOME/applications` directory.

Table 14–3 Impact of Modifying Files in an EAR

Modified File	Action Initiated
`application.xml`	Modifying the J2EE standard application deployment descriptor within an EAR forces a restart of the application.
OC4J deployment descriptors	Modifying an OC4J proprietary deployment descriptor packaged within a deployed EAR, such as `orion-application.xml`, causes the EAR to be redeployed and then restarted with the updated configuration.

Impact of Redeploying Modified Files in a War Automatically

Table 14–4 describes the impact of modifying files or deployment descriptors within an updated WAR file deployed as part of an application. The updated WAR file can either be packaged in an EAR or be copied to the Web module's `ORACLE_HOME/j2ee/instance/applications/webAppName` directory.

OC4J checks the timestamp of each file and redeploys only those that have a different timestamp from the other files.

Table 14–4 Impact of Modifying Files in a WAR

Modified File	Action Initiated
`web.xml` or `orion-web.xml`	Modifying either the J2EE standard Web deployment descriptor or the OC4J specific descriptor within an updated WAR file causes the Web module to be restarted with the new configuration.
`WEB-INF/lib/`	Updating a library JAR file in this path forces the Web module to be restarted and modified classes to be reloaded.
`WEB-INF/classes/*`	Updating a file in this path forces the Web module to be restarted and modified classes to be reloaded.

Table 14–4 (Cont.) Impact of Modifying Files in a WAR

Modified File	Action Initiated
`.tld` files	Updating a TLD file forces the Web module to be restarted.

Forcing a One-Time Redeployment Using admin.jar

The `admin.jar` command-line utility can be used to administer a standalone OC4J instance. This tool includes an `-updateConfig` option that enables OC4J polling on an as-needed basis, forcing OC4J to check its directories for updated files and reload any that have changed.

Usage Notes:

- The `admin.jar` tool can only be used against a standalone OC4J instance. It cannot be used against an OPMN-managed instance.
- `checkForUpdates` must be set to either `all` or `adminClientOnly` to use this feature.

The utility is installed in *`ORACLE_HOME`*`/j2ee/`*`instance`* by default. The OC4J server must be started before this utility can be used, except for converting a `data-sources.xml` file before deployment.

The syntax for this command follows:

```
java -jar admin.jar ormi://host:ormiPort adminId adminPassword
 -updateConfig
```

In the following example, 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`.

```
cd ORACLE_HOME\j2ee\instance
java -jar admin.jar ormi://localhost:23791 oc4jadmin password -updateConfig
```

See the *Oracle Containers for J2EE Configuration and Administration Guide* for additional instructions on using `admin.jar`.

15 Troubleshooting Deployment Errors

This chapter discusses common errors that may occur during deployment. It includes the following sections:

- Interruptions During Application Deployment
- Exceptions During Application Deployment

Interruptions During Application Deployment

If the deployment process is interrupted for any reason, you may need to clean up the temp directory, which by default is `/var/tmp`, on your system.

The Application Server Control deployment wizard uses approximately 20 MB in swap space of the temporary directory for storing information during the deployment process. At completion, the deployment wizard cleans up the additional files from the temporary directory.

However, if the wizard is interrupted, it may not have the time or opportunity to clean up the temp directory. Thus, you must clean up any additional deployment files from this directory yourself. Otherwise, this directory may fill up, which will disable any further deployment.

You can change the temp directory at OC4J startup by setting the `java.io.tmpdir` command-line option to a new location. See the *Oracle Containers for J2EE Configuration and Administration Guide* for details on setting system properties.

Exceptions During Application Deployment

This section provides details on the following types of errors that may occur during deployment:

- OC4J Out-of-Memory Errors
- Java Compiler Out-of-Memory Errors
- Stack Overflow Errors
- Errors for Number of Open Files

OC4J Out-of-Memory Errors

Deploying a large EAR file, such as a file larger than 75 MB, may cause OC4J to throw `java.lang.OutOfMemory` errors. If sufficient memory is available, you can eliminate this problem by increasing the heap size for the OC4J process at OC4J startup. For example:

```
java -Xms512m -Xmx512m -jar oc4j.jar
```

This problem might also be encountered during deployment of an application with the `admin.jar` command-line utility. Again, the solution is to increase the heap size for this utility:

```
java -Xms512m -Xmx512m -jar admin.jar ormi://localhost:23791 admin welcome -deploy
...
```

If you are running OC4J in a Linux or UNIX environment, verify that the `ulimit` settings allow the JVM process to allocate this much memory.

Java Compiler Out-of-Memory Errors

OC4J may return the following message when using the `javac` compiler in out-of-process mode when compiling EJB wrapper classes:

```
The system is out of resources.
Consult the following stack trace for details.
java.lang.OutOfMemoryError: Java heap space
```

This message indicates that the external JVM process spawned to execute the `javac` compiler has run out of memory.

The default heap size allocated to the compiler is 1024 MB. To allocate more memory to the compiler, increase the heap size by setting the `-Xmx` option through the `options` attribute of the `<java-compiler>` element in `server.xml`. For example:

```
<java-compiler name="javac" in-process="false" options="-J-Xmx2048m"/>
```

Stack Overflow Errors

The `javac` compiler may throw a `java.lang.StackOverflowError` when compiling EJB wrapper classes if your application is too large.

The thread stack size option enables the control of the `stacksize` attribute of a thread attributes object. This attribute specifies the minimum stack size to be used for the created thread.

If this occurs, you can increase the thread stack size by setting the `-Xss` option through the `options` attribute of the `<java-compiler>` element in `server.xml`. For example:

```
<java-compiler name="javac" in-process="false" options="-J-Xmx2048m -J-Xss=8m"/>
```

Errors for Number of Open Files

When deploying large applications, the OC4J JVM may throw "too many open files" exceptions. For example:

```
java.net.SocketException: Too many open files
java.io.IOException: Too many open files
```

These exceptions indicate that the operating system has run short of file descriptors, which are used by processes to identify open files of different types, including sockets or pipes.

Increasing the number of file descriptors will typically resolve this kind of problem. In a UNIX environment, you can increase the number of file descriptors with the `ulimit -n` command. The maximum number of file descriptors, as well as the maximum size that can be allocated to a process, are defined by a resource limit. Refer to your operating system-specific system administration manuals for instructions on setting the hard limit values.

A
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

License for Apache Web Server 2.0

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.

The DBI module is Copyright (c) 1994-2002 Tim Bunce. Ireland. All rights reserved.

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:

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.

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

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

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

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

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

mod_perl 1.29 License

mod_perl 1.99_16 License

(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:

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

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

Index

A

-addDataSourceConnectionPool command, 11-31
addDataSourceConnectionPool task, 10-33
-addDestination command, 11-38
addDestination task, 10-43
-addImportSharedLibrary command, 11-25
addImportSharedLibrary task, 10-27
-addJMSConnectionFactory command, 11-37
addJMSConnectionFactory task, 10-42
-addManagedDataSource command, 11-33
addManagedDataSource task, 10-36
-addNativeDataSource command, 11-34
addNativeDataSource task, 10-38
-addRemoveInheritedSharedLibrary command, 11-26
addRemoveInheritedSharedLibrary task, 10-29
-addWebSite command, 11-8
addWebSite task, 10-10
admin_client.jar tool
  deploying with, 11-1
  shut down, 11-30
administration client distribution, 11-41
admin.jar command-line utility, 12-1
admin.jar tool
  deploying with, 12-2
  undeploying with, 11-21, 12-4
admin.jar, deploying with, 12-1
Ant tasks, 1-3
  deploying with, 10-1
  OC4J, 1-3
  scripted deployment, 1-3
application mount points, configuring, 2-2
Application Server Control
  creating shared libraries, 9-18
  deploying to clusters with, 9-3
  deploying to OC4J with, 9-3
  deploying with, 9-1
  managing connection pools, 9-20
  managing data sources, 9-20
  managing JMS resources, 9-21
  managing shared libraries, 9-18
  restarting applications, 9-19
  restarting OC4J Instances, 9-20
  starting and stopping, 10-30, 11-27
  starting applications, 9-19
  stopping applications, 9-19
  stopping OC4J instances, 9-20
  undeploying an application, 9-18
applications
  auto-deployment, 14-1
  automatic redeployment, 14-1
  deploying, 10-24, 11-8
  deployment overview, 1-1
  starting, restarting, or stopping, 10-30, 11-27
  undeploying, 2-6, 10-24, 11-8
auto-deployment directory, 14-6
automatic deployment, 14-1
automatic redeployment, 14-1

B

batch deployment, script file, 11-15
-bindAllWebApps command, 11-15
bindAllWebApps task, 10-16
binding an application, 2-2
binding web modules, 10-16, 11-15
-bindWebApp command, 11-16
bindWebApp task, 10-17
build.xml file for Ant tasks, 10-4

C

cluster topology definition, 1-1
clustering configuration, 9-8
clusters, deploying to, 9-3
configuration, setting JSP parameters, 5-3
connecting to a remote Oracle Application Server, 11-39
connecting to OC4J from, 13-1

D

-deleteImportSharedLibrary command, 11-26
deleteImportSharedLibrary task, 10-28
-deleteRemoveInheritedSharedLibrary command, 11-26
deleteRemoveInheritedSharedLibrary task, 10-29
-deploy command
  EAR file, 11-9
  RAR file, 11-14
  remote client, 11-11

WAR file, 11-12
deploy task
EAR file, 10-11
RAR file, 10-15
WAR file, 10-13
deployable components, 1-1
deployment descriptors
and deployment plans, 8-2
overview, 8-2
deployment from, 13-1
deployment plans
and packaged deployment descriptors, 8-2
creating, 8-3
list of properties, 8-4
overview, 8-1
deployment process, overview, 2-1
deployment tasks, 9-5
configuring clustering, 9-8
configuring EJB modules, 9-6
managing shared libraries, 9-7
mapping roles, 9-6
resource mappings, 9-9
selecting the security provider, 9-5
deployment tools, 12-1
admin_client.jar Command-Line Utility, 1-4
admin_client.jar command-line utility, 11-1
admin.jar, 12-1
admin.jar Command-Line Utility, 1-4
Ant tasks, 1-3, 10-1
Application Server Control, 1-3, 9-1
Eclipse, 13-1
JDeveloper, 1-4
deployment, error recovery, 15-1
-describeSharedLibrary command, 11-24

E

Eclipse, 13-1
Ant tasks, 13-1, 13-2
deploying Web applications with, 13-1
Web Application deployment, 13-2
Web Tools Platform for deployment, 13-1
EJB modules
deploying, 3-1
impact of redeployment on clients, 3-5
incremental deployment, 3-1
updating, 11-21
entity beans, redeploying, 3-5

G

-getDataSourcesDescriptor command, 11-36
getDataSourcesDescriptor task, 10-41
-getDestinations command, 11-39
getDestinations task, 10-44
-getJMSConnectionFactories command, 11-38
getJMSConnectionFactories task, 10-43
groups
definition, 1-1
deploying to, 9-3, 10-7
OC4J, 1-1

H

-help command, 11-7
hot deployment, 1-2

I

incremental EJB deployment, 3-1

J

JConsole, 11-39
JDeveloper, deploying with, 1-4
JMX client for managing OC4J, 11-41
JSPs
adding to a deployed application, 5-2
modifying in a deployed application, 5-2

L

libraries
creating shared, 11-22
installing shared, 11-22
managing shared, 11-22
libraries, installing shared, 11-22
-listApplications command, 11-29
listApplications task, 10-32
-listDataSourceConnectionPools command, 11-32
listDataSourceConnectionPools task, 10-35
-listDataSources command, 11-36
listDataSources task, 10-40
-listSharedLibraries command, 11-24
-listWebBindings command, 11-18
listWebBindings task, 10-20

M

managing OC4J through a JMX client, 11-41
MBeanServer, 11-39
mod_oc4j, 2-2
-modifySharedLibrary command, 11-23
modifySharedLibrary task, 10-25
mount points, configuring for deployed applications, 2-2

O

OC4J
administration, 11-1
Ant tasks, 1-3
command-line interface, 11-1
restarting, 11-29
shutting down, 11-30
stopping, 11-29
OC4J Ant Tasks, 1-3, 10-1
OC4J polling
definition, 14-1
disabling, 14-7
Oracle HTTP Server, 2-2

P

parent application, designating, 2-1
polling, 14-1
-publishSharedLibrary command, 11-22
publishSharedLibrary task, 10-24

R

-redeploy command, 11-19
redeploy task, 10-20
-removeDataSourceConnectionPool command, 11-32
removeDataSourceConnectionPool task, 10-35
-removeDestination command, 11-39
removeDestination task, 10-44
-removeJMSConnectionFactory command, 11-37
removeJMSConnectionFactory task, 10-42
-removeManagedDataSource command, 11-33
-removeNativeDataSource command, 11-34
removeNativeDataSource task, 10-38
-removeSharedLibrary command, 11-25
removeSharedLibrary task, 10-27
resource adapters
  deploying, 12-5
  undeploying, 12-5
-restart command, 11-29
-restartApp command, 11-28
-restartApp task, 10-31
restarting
  applications, 10-30, 11-27
  OC4J, 11-29
restartServer task, 10-33

S

script file for batch deployment, 11-15
scripted deployment, 10-1, 12-1
  using admin.jar, 1-4
  using Ant tasks, 1-3
session beans, redeploying, 3-5
shared libraries
  creating, 11-22
  installing, 11-22
  managing, 11-22
-shutdown command, 11-30
shutdownServer task, 10-33
-start command, 11-27
-start task, 10-30
starting applications, 10-30, 11-27
-stop command, 11-27
-stop task, 10-31
stopping a group of OC4J instances, 11-30
stopping an OC4J instance in a managed environment, 11-30
stopping applications, 10-30, 11-27
stopping OC4J, 11-29
stopping OC4J in a standalone configuration, 11-30

T

-testDatabaseConnection command, 11-35
testDatabaseConnection task, 10-39
-testDataSource command, 11-35
testDataSource task, 10-40
-testDataSourceConnectionPool command, 11-31
testDataSourceConnectionPool task, 10-34

U

-unbindAllWebApps command, 11-17
unbindAllWebApps task, 10-18
unbinding web modules, 10-16, 11-15
-unbindWebApp command, 11-17
unbindWebApp task, 10-19
-undeploy command, 11-21
undeploy task, 10-22
undeployment, 2-6
-updateEJBModule command, 11-21
updateEJBModule task, 10-23
-usage command, 11-7

V

valid archive types, 1-1
-validateURI command, 11-5

W

WAR files
  deploying as standalone modules, 5-1
  wrapped within an EAR file, 5-1
Web module
  adding JSPs to, 5-2
  deploying, 5-1
  modifying JSPs in, 5-2
  redeploying, 5-1
web module
  binding, 10-16, 11-15
  unbinding, 10-16, 11-15
Web services
  deploying, 7-1
  redeploying, 7-1
web site
  bind/unbind web module, 10-16, 11-15
Web site, binding application to, 2-2
Web Tools Platform, 13-1

X

XML schemas, 8-2
XSDs, 8-2