Oracle® Containers for J2EE

Developer's Guide

10*g* (10.1.3.1.0)

B28952-01

October 2006

ORACLE®

Oracle Containers for J2EE Developer’s Guide, 10*g* (10.1.3.1.0)

B28952-01

Primary Author: Bonnie Vaughan

Contributing Author: Dan Hynes

Contributors: Bryan Atsatt, Steve Button, Olivier Caudron, Marcelo Goncalves, Kurt Heiss, Bruce Irvin, James Kirsch, Mike Lehmann, Philippe Le Mouel, Sheryl Maring, Kuassi Mensah, Jasen Minton, Rose Pan, Debu Panda, Gael Stevens, Charlie Shapiro, Brian Wright

Contents

Index

Preface

This document provides detailed discussions on various facets of architecting, developing, and packaging a J2EE-compliant application for deployment into Oracle Containers for J2EE (OC4J). It summarizes standard implementation but focuses primarily on Oracle implementation details and value-added features. As much as possible, the focus is on best practices and guidelines that Oracle recommends.

This preface contains the following sections:

Intended Audience

This document is intended for Java developers involved in building J2EE applications to be deployed into OC4J and for system architects designing such applications. It is based on the assumption that readers are already familiar with the following technologies:

- J2EE and Web technologies
- The Java programming language
- Web server and servlet environment configuration
- Oracle JDBC (for JSP applications accessing Oracle Database)

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

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

Related Documents

For more information, see the following Oracle resources.

Additional OC4J documents:

- *Oracle Containers for J2EE Deployment Guide*

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

- *Oracle Containers for J2EE Configuration and Administration Guide*

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

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

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

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

  This 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 provides conceptual information as well as detailed syntax and usage information for tag libraries, JavaBeans, and other Java utilities provided with OC4J.

- *Oracle Containers for J2EE Services Guide*

  This 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 provides information about Enterprise JavaBeans development 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 describes Web services development and configuration in OC4J and Oracle Application Server.

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

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

Conventions

This document uses the following text conventions.

Convention	Meaning
boldface	Boldface type indicates either 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 or code within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Getting Started with OC4J

This chapter describes Oracle Containers for J2EE 10*g* (10.1.3.1.0), or OC4J, which is part of the Oracle Application Server 10*g* Release 3 (10.1.3.1.0) installation or a standalone server.

This chapter includes the following topics:

- Introduction to OC4J
- Information in the OC4J Documentation Set
- OC4J Installation

Introduction to OC4J

Oracle Containers for J2EE 10*g* (10.1.3.1.0) provides a complete Java 2 Enterprise Edition (J2EE) 1.4-compliant environment.

OC4J provides all the containers, APIs, and services that J2EE specifies. It is based on technology licensed from Ironflare Corporation, which develops the Orion server—one of the leading J2EE containers. As such, the product and some of the documentation still contains some reference to the Orion server.

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

J2EE Support in OC4J

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

Table 1–1 OC4J J2EE Support

J2EE Standard APIs	Version Supported By OC4J
JavaServer Pages (JSP)	2.0
Servlets	2.4
Enterprise JavaBeans (EJB)	2.1, 3.0 (Complete EJB 3.0 and JPA implementation)
Java Management Extensions (JMX)	1.2
J2EE Management	1.0
J2EE Application Deployment	1.1
Java Transaction API (JTA)	1.0
Java Message Service (JMS)	1.1

Table 1–1 (Cont.) OC4J J2EE Support

J2EE Standard APIs	Version Supported By OC4J
Java Naming and Directory Interface (JNDI)	1.2
Java Mail	1.2
Java Database Connectivity (JDBC)	3.0
Oracle Application Server Java Authentication and Authorization Service (JAAS) Provider	1.0
J2EE Connector Architecture	1.5
Java API for XML-Based RPC (JAX-RPC)	1.1
SOAP with Attachments API for Java (SAAJ)	1.2
Java API for XML Processing (JAXP)	1.2
Java API for XML Registries (JAXR)	1.0.5

New Features in OC4J

Oracle Containers for J2EE 10*g* (10.1.3.*x*) includes a number of new features and enhancements, which are summarized in the following text.

Support for Web Services

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

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

Support for J2EE 1.4 Application Management and Deployment Specifications

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

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

Support for Oracle Application Server TopLink

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

OracleAS Job Scheduler

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

Two-Phase Commit Transaction Coordinator Functionality

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

Generic JMS Resource Adapter Enhancements

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

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

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

Support for the Enterprise JavaBeans 3.0

OC4J 10g (10.1.3.1.0) provides complete support for the Enterprise JavaBeans 3.0 final specification, including support for EJB annotations and dependency injections. The final specification is available at the following Web site:

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

Note: OC4J must use JDK 5.0 to enable EJB 3.0 support. This JDK is included with the current 10g (10.1.3.1.0) release, in which OC4J uses JDK 5.0 by default.

You can use following annotations and others in your EJBs:

- MessageDrivenDeployment
- StatefulDeployment
- StatelessDeployment

The *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide* describes how to use EJB 3.0 annotations and EJB 3.0 JPA extensions. The *Oracle Application Server Annotations Java API Reference* provides reference information for EJB 3.0 annotations.

Support for the <library-directory> Element

The `<library-directory>` element of the `application.xml` file can be used to specify shared libraries for OC4J instances. Directories specified in this element are scanned for archives to include at OC4J startup.

Information in the OC4J Documentation Set

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

Table 1–2 Location of Information for J2EE Subjects

J2EE Subject	The Subject is Documented in this OC4J Documentation Book
JSP	*Oracle Containers for J2EE Support for JavaServer Pages Developer's Guide*
JSP Tag Libraries	*Oracle Containers for J2EE JSP Tag Libraries and Utilities Reference*
Servlet	*Oracle Containers for J2EE Servlet Developer's Guide*
EJB	*Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide*
JTA	*Oracle Containers for J2EE Services Guide*
Data Sources	*Oracle Containers for J2EE Services Guide*
JNDI	*Oracle Containers for J2EE Services Guide*
JMS	*Oracle Containers for J2EE Services Guide*
RMI and RMI/IIOP	*Oracle Containers for J2EE Services Guide*
Security	*Oracle Containers for J2EE Security Guide*
CS*i*V2	*Oracle Containers for J2EE Security Guide*
J2CA	*Oracle Containers for J2EE Resource Adapter Administrator's Guide*
Java Object Cache	*Oracle Containers for J2EE Services Guide*
Web Services	*Oracle Application Server Web Services Developer's Guide*
HTTPS	*Oracle Containers for J2EE Services Guide*

OC4J Installation

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

2

Developing Startup and Shutdown Classes

This chapter provides guidelines on developing startup and shutdown classes that are called after OC4J initializes or before OC4J terminates. Startup classes can start services and perform functions after OC4J initiates. Shutdown classes can terminate these services and perform functions before OC4J terminates.

When you compile these classes, the `oc4j-api.jar` file must be in a path specified in the Java `CLASSPATH` environment variable, such as *`ORACLE_HOME`*`/j2ee/home/oc4j-api.jar`.

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

This chapter includes these topics:

- Developing Startup Classes
- Developing Shutdown Classes

Developing Startup Classes

Startup classes are executed only once after OC4J initializes. They are not reexecuted every time the `server.xml` file is touched. A startup class implements the `oracle.j2ee.server.OC4JStartup` interface, which contains two methods:

- `preDeploy`

  This method executes before any OC4J application initialization.

- `postDeploy`

  This method executes after all OC4J applications initialize.

In these methods, you can implement code for starting services, performing other initialization routines, ending services, and performing other termination routines.

Each method requires two arguments:

- `Hashtable`

  This argument specifies a hash table that is populated from the configuration.

- `Context`

  This argument specifies a JNDI context to which you can bind to process values contained within the context.

Both methods return a `String` value, which is currently ignored.

> **Note:** Oracle strongly recommends that you if give your startup class a constructor, you give it a public, no-argument constructor. Otherwise, `java.lang.IllegalAccessException` may be thrown when OC4J attempts to invoke a member method of this class.

After you create a startup class, you must configure it within the `<startup-classes>` element in the `server.xml` file. You can access this file through the Application Server Control Console by selecting **Advanced Properties** on the OC4J home page. Each `OC4JStartup` class is defined in a single `<startup-class>` element within the `<startup-classes>` element. Each `<startup-class>` element defines the following attributes:

- The name of the class that implements the `oracle.j2ee.server.OC4JStartup` interface
- Whether a failure is fatal

  The default is not fatal. When an exception is thrown for a failure that is not considered fatal, OC4J logs the exception and continues. When an exception is thrown for a failure that is considered fatal, OC4J logs the exception and exits.
- The order of execution

  Each startup class receives an integer. The integers designate in what order the classes are executed, starting with `0`.
- The initialization parameters, which contain key-value pairs of type `String`, that OC4J takes

  Initialization parameters are provided through the input `Hashtable` argument. The name of each key-value pair must be unique because JNDI is used to bind each value to its name.

In the `<init-library>` element in the `server.xml` file, you configure the directory where the startup class resides or the directory and JAR file where the class is archived. The `path` attribute can be fully qualified or relative to `/j2ee/`*`instance`*`/config`.

For example, the configuration for the `TestStartup` class is contained within a `<startup-class>` element in the `server.xml` file:

- The `failure-is-fatal` attribute is `true`, so an exception would cause OC4J to exit.
- The `<execution-order>` subelement contains `0`, so this is the first startup class to execute.
- Two initialization key-value pairs are defined, of type `String`. These key-value pairs will be populated in the hash table that the `Hashtable` argument specifies:

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

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

Add the following notation to the `server.xml` file to define the `TestStartup` class:

```
<startup-classes>
  <startup-class classname="test.oc4j.TestStartup" failure-is-fatal="true">
```

```
    <execution-order>0</execution-order>
    <init-param>
      <param-name>oracle.test.startup</param-name>
      <param-value>true</param-value>
     </init-param>
     <init-param>
      <param-name>startup.oracle.year</param-name>
      <param-value>2002</param-value>
     </init-param>
   </startup-class>
 </startup-classes>
```

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

The following example shows `TestStartup`, which implements the `oracle.j2ee.server.OC4JStartup` interface. The `preDeploy` method retrieves the key-value pairs from the hash table and prints them. The `postDeploy` method is a null method. The `oc4j.jar` file must be in the path that the Java `CLASSPATH` environment variable specifies when you compile `TestStartup`.

```
package text.oc4j;
import oracle.j2ee.server.OC4JStartup;

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

public class TestStartup implements OC4JStartup {

   //public, no-argument constructor
    public TestStartup() {
    }

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

        return "ok";
    }

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

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

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

When OC4J starts, the `preDeploy` method of `TestStartup` is executed before any application is initialized. OC4J populates the JNDI context with the values from the hash table. If `TestStartup` throws an exception, then OC4J exits because the `failure-is-fatal` attribute was set to `true`.

Developing Shutdown Classes

Shutdown classes are executed before OC4J terminates. A shutdown class implements the `oracle.j2ee.server.OC4JShutdown` interface, which contains two methods, `preUndeploy` and `postUndeploy`, in which you can implement code for shutting down services or perform other termination routines.

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

Each method requires two arguments: a hash table that is populated from the configuration and a JNDI context to which you can bind to process values specified in key-value pairs.

Note: Oracle strongly recommends that if you give your shutdown class a constructor, you give it a public, no-argument constructor. Otherwise, `java.lang.IllegalAccessException` may be thrown when OC4J attempts to invoke a member method of this class.

The implementation and configuration is identical to the shutdown classes as described in "Developing Startup Classes" on page 2-1 with the exception that the configuration is defined within the `<shutdown-classes>` and `<shutdown-class>` elements and there is no `failure-is-fatal` attribute. Thus, the configuration for a `TestShutdown` class would be as follows:

```
<shutdown-classes>
  <shutdown-class classname="test.oc4j.TestShutdown">
    <execution-order>0</execution-order>
    <init-param>
      <param-name>oracle.test.shutdown</param-name>
      <param-value>true</param-value>
     </init-param>
     <init-param>
       <param-name>shutdown.oracle.year</param-name>
       <param-value>2002</param-value>
     </init-param>
   </shutdown-class>
 </shutdown-classes>
```

Assuming that the `TestShutdown` class is archived in `"/j2ee/home/app1/shutdown.jar"`, add another `<init-library>` element in the `server.xml` file, as follows:

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

3

Utilizing the OC4J Class Loading Framework

This chapter provides guidelines on understanding and using the new class-loading framework in OC4J. Discussions on common class loader problems and recommendations for avoiding them are also provided.

The following topics are included:

Class Loading in OC4J

This section contains the following topics:

Overview of Class Loading

The term **class loading** refers to the process of locating the bytes for a given class name and converting them into a Java class instance. All class instances within a Java Virtual Machine (JVM) start as an array of bytes, structured in the class file format defined by the JVM specification.

Class loading is performed by the JVM during the startup process, and subsequently by **class loaders**, subclasses of the `java.lang.ClassLoader` class, which find and load class files at runtime. Class loaders provide an abstraction that enables the JVM to load classes without any knowledge of where the class bytes come from, for both local and remote storage as well as dynamic class generation.

Each class loader works with one or more **code sources**, root locations from which the class loader searches for classes. Code sources can be defined to represent physical storage of binary class files, Java sources that must first be compiled or even classes generated on the fly.

Standard class loaders are linked together in a parent-child hierarchy, with each class loader having an associated parent class loader. This hierarchy represents a tree structure, ranging in complexity from simple chains to complex, multibranched trees.

In this hierarchy, a child class loader imports a set of class loaders from its parent loader. In the OC4J context, all J2EE applications running within an OC4J instance are children of the `system` application. As a result, a class loader created at the application level imports a set of class loaders from the `system.root` class loader.

See Figure 3–1 below for a graphical representation of the class loader tree structure used in OC4J.

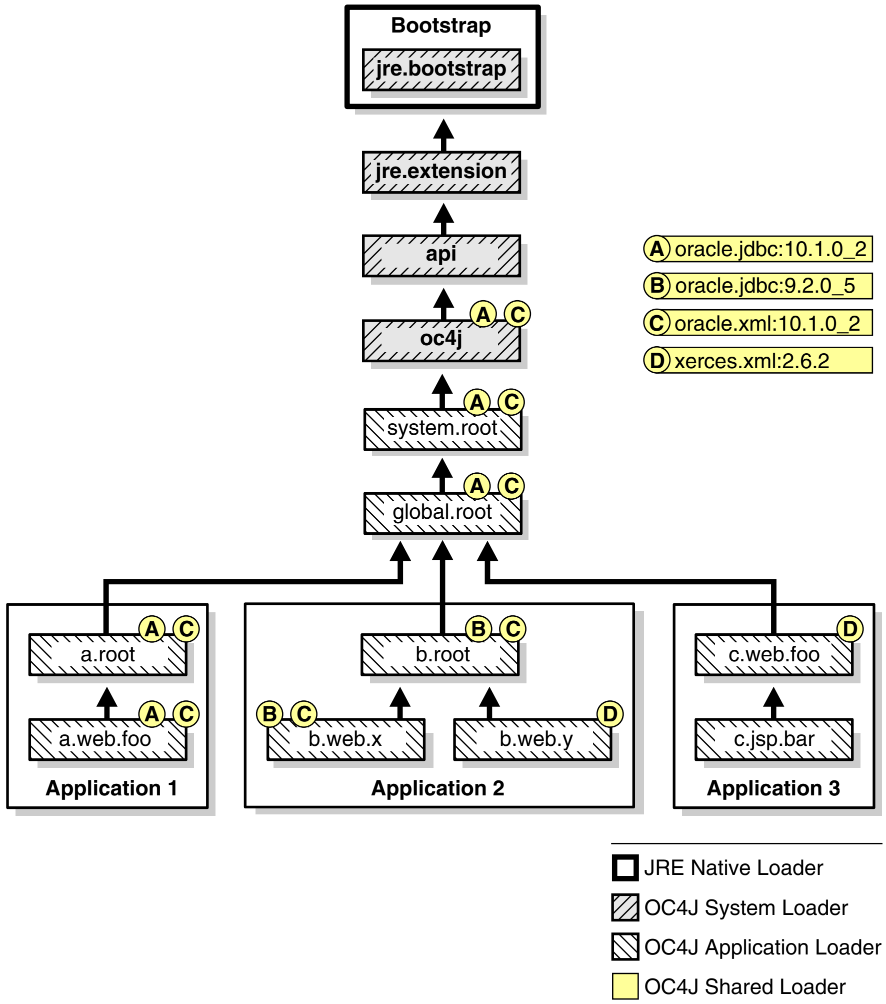
Class Versioning with Shared Libraries in OC4J

The class loader hierarchy ensures that a J2EE application deployed into the OC4J instance inherits a set of libraries by default from the `default` application. A Web module bound to this application, in turn, inherits a set of classes from the application, as well as the classes inherited from the `system` application, which sits at the root of application hierarchy in OC4J.

However, this inheritance model is not always desirable, such as when a nondefault version of a library or class is needed by an application or module. The OC4J class loading infrastructure addresses this problem by enabling class loaders created for an application or module to import a different version of a shared library than the default imported by the parent class loader, or even remove a class loader from the set of inherited class loaders entirely.

Figure 3–1 illustrates the class loader tree structure in OC4J.

Figure 3–1 The OC4J Class Loader Tree



- The `jre.bootstrap` loader is a proxy for the native bootstrap loader built into the JVM. The native bootstrap loader itself is not directly visible at runtime.
- The `jre.extension` loader is a custom replacement for the JRE-supplied “extension” loader.
- The `api` loader contains J2EE and OC4J API classes that must be visible to all applications as well as to all OC4J internal classes.
- The `oc4j` loader contains OC4J system classes.
- The `system.root` loader is created for the OC4J `system` application.

  Because `system` is at the root of the application hierarchy, the classes in this loader are inherited by default by all other applications deployed into the OC4J instance.
- `global.root` is the class loader created for the `default` application, which is the default parent of all J2EE applications deployed to the OC4J instance.
- *`app-name`*`.root` is the root loader for a deployed application.
- *`app-name`*`.web.`*`web-mod-name`* class loaders each contain Web module - classes packaged within a WAR file.

- *app-name*.jsp.*jsp name* loads a compiled JSP implementation class.

The OC4J shared class loaders, oracle.jdbc:10.1.0_2, oracle.jdbc:9.2.0_5, oracle:xml:10_1_02 and xerces:xml:2.6.2, represent *shared libraries* declared in the OC4J instance. Each shared library definition consists of these items:

- A shared library name, such as xerces.xml
- A version number that typically represents the shared library's implementation version, such as 2.6.2
- One or more code sources, JAR or ZIP files, containing the classes that comprise the library

Class Loaders are created at runtime based on the shared library definitions within the OC4J instance. Class Loaders are registered using a concatenation of the shared library name and the version number; for example, xerces.xml:2.6.2.

See "Installing and Publishing a Shared Library in OC4J" on page 3-10 for detailed instructions on creating and installing shared libraries.

Note how JDBC driver and XML parser classes are loaded by the three deployed applications: While Application 1 follows the default behavior of inheriting the classes contained in the oracle.jdbc:10_1_02 and oracle:xml:10_1_02 shared loaders from its parent, Application 2 and Application 3 each import alternative driver and parser class loaders for their use.

By default, an application inherits the same set of shared libraries present in its parent application, including libraries inherited from the system application. This means, for example, that an application will by default use the Oracle JDBC driver and Oracle XML parser, which are inherited from the system application.

However, using OC4J's class versioning capabilities, you can override an inherited library with a different version, or even remove a library from the list of inherited libraries altogether.

Shared Libraries That Applications Import by Default

The default set of shared libraries imported by *all* application class loaders within the OC4J instance is specified within the <imported-shared-libraries> element in *ORACLE_HOME*/j2ee/*instance*/system-application.xml. This is the configuration file for the system application, an internal component of Oracle Containers for J2EE that sits at the root of the application hierarchy and provides classes and configuration required at OC4J startup.

By default, an application deployed to an OC4J instance will inherit these Oracle shared libraries:

- oracle.jdbc:10.1.0_2
- oracle.xml:10.1.0_2
- oracle.cache:10.1.3
- oracle.http.client:10.1.3
- oracle.sqlj:10.1.3
- soap:10.1.3
- oracle.jwsdl:10.1.3

Configuring an Application to Import a Nondefault Version of a Shared Library

You can force an application to import a different version of a shared library than the one declared in `system-application.xml` by creating a shared library with the same name, but assigning a different version number. You will then configure the application to import this shared library.

See the following sections for details:

- "Example: Importing an Earlier Version of the Oracle JDBC Driver" on page 3-5 for an end-to-end example
- "Example: Configuring an Application to Use a DataDirect JDBC Driver" on page 3-6 for another complete example
- "Installing and Publishing a Shared Library in OC4J" on page 3-10 for additional options for creating shared libraries
- "Configuring an Application to Import a Shared Library" on page 3-14 for additional options for configuring applications to import a particular shared library

Example: Importing an Earlier Version of the Oracle JDBC Driver

The following example shows you how to configure an application to use an Oracle 9.2.0_5 JDBC driver, which is an earlier version of the Oracle JDBC driver than the version packaged with OC4J 10*g* (10.1.3.1.0). This example applies only to thin JDBC drivers and does not apply to the Oracle Call Interface (OCI) drivers.

Step 1: Create the Shared Library in OC4J

You can install a shared library for the 9.2.x JDBC driver using any of the mechanisms described in "Options for Installing and Publishing a Shared Library" on page 3-11. This example illustrates this task using Application Server Control Console.

> **Note:** To use a JDBC driver that is not packaged with OC4J, you must create a managed data source specifically for use by the application, then configure the application to use it.
>
> This is necessary because the default JDBC drivers and data sources used by applications are imported by the global `system` application's class loader. Because your application is loading a different driver, it must also load a data source for the driver to use.
>
> See the *Oracle Containers for J2EE Services Guide* for details on creating and using application-specific managed data sources.

1. Click **Administration>Shared Libraries**. Note the default JDBC driver shared library, `oracle.jdbc:10.1.0_2`.
2. Click **Create** on the Shared Libraries page.
3. Enter the name for the shared library. In this case, you will enter the same name as the existing library, which is `oracle.jdbc`.
4. Enter the shared library version, which in this case is 9.2.0_5.

5. Click **Add** to upload the library JAR file to the OC4J instance. The following shared library declaration is added to the *ORACLE_HOME*/j2ee/*instance*/server.xml file:

```
<shared-library name="oracle.jdbc" version="9.2.0_5">
  <code-source path="ojdbc14.jar"/>
</shared-library>
```

Step 2: Configure an Application to Use the Shared Library

Once the shared library has been created in OC4J, you can configure an application to use it instead of the default shared library installed with OC4J.

The following example illustrates how to do this at the time the application is deployed using Application Server Control Console.

1. Select Applications>Deploy to launch the Application Server Control Console deployment wizard.
2. Supply the path to the application in the first page of the wizard.
3. Specify the application name and supply any context URI mappings in the second page.
4. Click **Configure Class Loading** in the third page of the wizard (Deploy: Deployment Settings).
5. Note that both versions of the oracle.jdbc shared library are listed in the Import Shared Libraries frame. Specify the version number you want to use—`9.2.0_5`—in the Maximum Version To Use column.
6. Deploy the application. After the application is deployed, notethe following entry in the application's `orion-application.xml` file:

```
<imported-shared-libraries>
  <import-shared-library name="oracle.jdbc" max-version="9.2.0_05"/>
</imported-shared-libraries>
```

Example: Configuring an Application to Use a DataDirect JDBC Driver

The Oracle Application Server distribution includes several JDBC drivers to provide connectivity to non-Oracle databases. The following example shows you how to configure an application to use the DataDirect Sybase driver to connect to a Sybase database.

Step 1: Create the Shared Library in OC4J

You can install a shared library for the driver using any of the mechanisms described in "Options for Installing and Publishing a Shared Library" on page 3-11. This example will illustrate this task using Application Server Control Console.

1. Click **Administration>Shared Libraries**.
2. Click **Create** on the Shared Libraries page.
3. Enter the name for the shared library; for example, `sybase.jdbc`.
4. Enter the shared library version, for example: 1.0.
5. Click **Add** to upload the library JAR files to the OC4J instance. Note that the `YMbase.jar` and `YMutil.jar` files are required to use any of the DataDirect drivers provided with Oracle Application Server.
   - `YMsybase.jar`

- `YMbase.jar`
- `YMutil.jar`

The following shared library declaration is added to the *ORACLE_HOME*/j2ee/*instance*/server.xml file:

```
<shared-library name="sybase.jdbc" version="1.0">
  <code-source path="YMbase.jar"/>
  <code-source path="YMutil.jar"/>
  <code-source path="YMsybase.jar"/>
</shared-library>
```

Step 2: Configure an Application to Use the Shared Library

Once the shared library has been created in OC4J, you can configure an application to use it instead of the default shared library installed with OC4J.

The following example illustrates how to do this at the time the application is deployed using Application Server Control Console.

1. Select Applications>Deploy to launch the Application Server Control Console deployment wizard.
2. Supply the path to the application in the first page of the wizard.
3. Specify the application name and supply any context URI mappings in the second page.
4. Click **Configure Class Loading** in the third page of the wizard (Deploy: Deployment Settings).
5. Check the **Import** checkbox for the `sybase.jdbc` shared library. Optionally specify `1.0` as the maximum version to use.
6. Deploy the application. After the application is deployed, notethe following entry in the application's `orion-application.xml` file:

```
<imported-shared-libraries>
  <import-shared-library name="sybase.jdbc" max-version="1.0"/>
</imported-shared-libraries>
```

Removing or Replacing an Oracle Shared Library Imported by Default

The shared library framework also allows a shared library to be removed from the set of shared libraries inherited by an application from its parent, and optionally allows a different shared library to be imported in its place.

Shared libraries inherited by default can be removed using one or more `<remove-inherited>` subelements within an `<imported-shared-libraries>` in the application's `orion-application.xml` file. The name of the library to remove is specified as the value for the `name` attribute.

For example, the following entry in `orion-application.xml` will prevent the application from importing the Oracle TopLink shared library:

```
<orion-application>
  <imported-shared-libraries>
    <remove-inherited name="oracle.toplink"/>
  </imported-shared-libraries>
</orion-application>
```

See the following sections for complete examples:

- Example: Replacing the Oracle XML Parser with the Xerces Parser
- Example: Removing an Oracle Shared Library at Deployment Time

Example: Replacing the Oracle XML Parser with the Xerces Parser

The following example illustrates how to remove the Oracle XML parser from the default set of shared libraries inherited from the `system` application using Application Server Control Console. It will also force the application to use the Xerces XML parser in its place.

Step 1: Create the Shared Library in OC4J

You can install a shared library for the Xerces parser using any of the mechanisms described in "Options for Installing and Publishing a Shared Library" on page 3-11. This example will illustrate this task using Application Server Control Console.

1. Click **Administration>Shared Libraries**.
2. Click **Create** on the Shared Libraries page.
3. Enter the name for the shared library. In this case, you will enter `xerces.xml`.
4. Enter the shared library version, which in this case is 2.5.0.
5. Click **Add** to upload the library JAR files to the OC4J instance. Upload the following Apache libraries:
   - `xercesImpl.jar`
   - `xml-apis.jar`

   The following shared library declaration is added to the *ORACLE_HOME*/j2ee/*instance*/server.xml file:

```
<shared-library name="xerces.mxl" version="2.5.0">
  <code-source path="xercesImpl.jar"/>
  <code-source path="xml-apis.jar"/>
</shared-library>
```

Step 2: Configure an Application to Use the Shared Library

Once the shared library has been created in OC4J, you can configure an application to use the Xerces parser instead of the default parser installed with OC4J.

The following example illustrates how to do this at the time the application is deployed using Application Server Control Console.

1. Select **Applications>Deploy** to launch the Application Server Control Console deployment wizard.
2. Supply the path to the application in the first page of the wizard.
3. Specify the application name and supply any context URI mappings in the second page.
4. Click **Configure Class Loading** in the third page of the wizard (Deploy: Deployment Settings).
5. Check the **Import** checkbox for the `xerces.xml` shared library. Optionally specify `2.5.0` as the maximum version to use.
6. To explicitly remove the Oracle parser, un-check the **Import** checkbox for the `oracle.xml` shared library to remove it from the list of shared libraries inherited by the application.

7. Optionally click the **Save Deployment Plan** button, and save the plan for re-use.
8. Deploy the application. After the application is deployed, note the following entry in the application's `orion-application.xml` file:

```
<orion-application>
  <imported-shared-libraries>
    <remove-inherited name="oracle.xml"/>
    <import-shared-library name="xerces.xml" max-version="2.5.0/>
  </imported-shared-libraries>
</orion-application>
```

Example: Removing an Oracle Shared Library at Deployment Time

The following example illustrates how to remove the Oracle TopLink shared library at the time the application is deployed using Application Server Control Console.

1. Select **Applications>Deploy** to launch the Application Server Control Console deployment wizard.
2. Supply the path to the application in the first page of the wizard.
3. Specify the application name and supply any context URI mappings in the second page.
4. Click **Configure Class Loading** in the third page of the wizard (Deploy: Deployment Settings).
5. Uncheck the **Import** checkbox for the `oracle.toplink` shared library to remove it from the list of shared libraries inherited by the application.
6. Optionally click the **Save Deployment Plan** button, and save the plan for re-use.
7. Deploy the application. After the application is deployed, note the following entry in the application's `orion-application.xml` file:

```
<orion-application>
  <imported-shared-libraries>
    <remove-inherited name="oracle.toplink"/>
  </imported-shared-libraries>
</orion-application>
```

Using a Packaged JAR Instead of an Oracle Shared Library

The class loading infrastructure enables you to package an XML parser or JDBC driver as a JAR with your application and then force the application to use it instead of the Oracle XML parser or JDBC driver installed with OC4J, without having to declare the JAR as a shared library within OC4J.

Configuring an Application to Use Its Own Shared Library

In this case, you will specify the default inherited Oracle library in the `<remove-inherited>` tag in `orion-application.xml`, which is then packaged with the JAR in the application's EAR file. After deployment into OC4J, the application will not import the default library installed with OC4J, causing the application's class loader to find and load your packaged library instead.

The following notation in `orion-application.xml` will prevent the application's class loader from importing the Oracle XML parser:

```
<imported-shared-libraries>
  <remove-inherited name="oracle.xml"/>
```

```
</imported-shared-libraries>
```

In the case of Web applications, you can specify that classes bundled within the application's WAR file be used through a notation in the application's `orion-web.xml` descriptor file.

First, add or uncomment the `<web-app-class-loader>` element in this file. Next, set the `search-local-classes-first` attribute to `true` in this element, which causes the class loader to find and load any libraries packaged in the WAR and to use these libraries rather than the corresponding libraries packaged with OC4J.

See "Specifying search-local-classes-first at Deployment Time", which follows, for details on how you can do this at deployment time using Application Server Control Console.

The entry in `orion-web.xml` looks like this:

```
<orion-web-app ...>
  ...
  <web-app-class-loader search-local-classes-first="true"
   include-war-manifest-class-path="true" />
  ...
</orion-web-app>
```

Note that this approach is not a guaranteed solution; if an application further up the hierarchy imports a shared library that includes the same classes, a collision is likely, and such collisions are difficult to debug. Ideally, you should use the shared library mechanism documented in this chapter to ensure that your Web applications use the correct library.

Specifying search-local-classes-first at Deployment Time

The following example illustrates how to set the `search-local-classes-first` attribute in the `orion-web.xml` file generated for the Web module at deployment time, using the Application Server Control Console.

1. Select **Applications>Deploy** to launch the Application Server Control Console deployment wizard.
2. Supply the path to the application in the first page of the wizard.
3. Specify the application name and supply any context URI mappings in the second page.
4. Click **Configure Class Loading** in the third page of the wizard (Deploy: Deployment Settings).
5. Under Configure Web Module Class Loaders, check the **Search Local Classes First** checkbox next to the name of the Web module containing the local JAR file to use.
6. Optionally click the **Save Deployment Plan** button, and save the plan for reuse.

Installing and Publishing a Shared Library in OC4J

Creating a shared library within an OC4J instance is essentially a two-step process. First, the binaries composing the shared library must be installed in the appropriate directory within OC4J. The shared library must then be declared in the OC4J server configuration file (`server.xml`), essentially "publishing" it within the OC4J instance.

This section includes the following topics:

- When You Should Use a Shared Library

- Options for Installing and Publishing a Shared Library
- How a Shared Library Is Installed and Published in an OC4J Instance

When You Should Use a Shared Library

Typically, applications deployed into OC4J will use the set of shared libraries packaged with OC4J, which are inherited from the `system` application. However, there are scenarios in which replacing or removing a library inherited from the application's parent is necessary. Example use cases include:

- Using a different version of the Oracle JDBC driver then the version packaged with OC4J
- Replacing the Oracle XML parser packaged with OC4J with a different parser for use by your application
- Sharing proprietary classes across one or more specific applications, rather than across all applications
- Making an open source library, such as Struts or the Spring Framework, available to multiple Web applications

Options for Installing and Publishing a Shared Library

OC4J provides several options for installing and publishing shared libraries within one or more OC4J instances. Each of these mechanisms will install the shared library in the *ORACLE_HOME*/`j2ee/`*instance*/`shared-lib` directory and make the required entry in `server.xml`.

- Oracle Enterprise Manager 10*g* Application Server Control Console

  Enables you to install a shared library on a specific OC4J instance through the **Administration>Administration Tasks>Shared Libraries** pages.

- The `publishSharedLibrary` Ant task

  Enables you to install a shared library on a standalone OC4J server or on a single OC4J instance in an Oracle Application Server environment managed by Oracle Process Manager and Notification Server (OPMN) .

- The `-publishSharedLibrary` command in `admin_client.jar`

  Also enables you to install a shared library on a single OPMN-managed OC4J instance or on a standalone OC4J server.

You can also manually install and publish a shared library within an OC4J instance following the process outlined in the next section, "How a Shared Library Is Installed and Published in an OC4J Instance".

Note: If you are using JDK1.4, Oracle Application Server 10.1.3 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 1.5 (JDK 5), 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.

How a Shared Library Is Installed and Published in an OC4J Instance

Shared libraries are installed in the `ORACLE_HOME/j2ee/instance/shared-lib` directory in OC4J. The process includes creating the correct directory structure within this directory and then copying one or more JAR or ZIP files that compose the library into the directory.

OC4J provides several tools that automate this process. See "Options for Installing and Publishing a Shared Library" on page 3-11 for an overview.

First, the following directory structure for the library is created within this directory:

```
ORACLE_HOME/j2ee/instance/shared-lib
  /library_name
    /version
      filename.jar
      filename.zip
      ...
```

The variables in the directory structure have these values:

- *`instance`* is the name of an OC4J instance, which is `home` by default in an Oracle Application Server environment and always `home` on a standalone OC4J server.
- *`library_name`* is a directory named with the name of the shared library; for example, `acme.common`.

  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`. If the technology is implemented by a single vendor, the technology name alone is sufficient.
- *`version`* is a subdirectory named for the shared library version number; for example, `2.5`. This value should ideally reflect the code implementation version.

After these directories have been created, copy each JAR or ZIP file containing the classes that compose the shared library into the *`version`* subdirectory. For example, assume the sample library consists of `acme-apis.jar` and `acmeImpl.jar`.

Given the preceding examples, the resulting directory structure within the OC4J server would be as follows:

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

Multiple versions of the shared library are installed by creating a new subdirectory for each version's archive files within this directory structure, as the following example illustrates:

```
ORACLE_HOME/j2ee/instance/shared-lib
  /acme.common
    /2.5
      acme-apis.jar
      acmeImpl.jar
    /3.0
      acme-apis.jar
      acmeImpl.jar
```

After the code sources are installed, the shared library is defined within a `<shared-library>` element that is added to the

ORACLE_HOME/j2ee/*instance*/server.xml file, which contains the configuration data for the OC4J instance. The `<shared-library>` element takes the following attributes and subelements:

- A required `name` attribute, the value of which must match the name of the shared library directory created within the `/shared-lib` directory.
- A required `version` attribute, the value of which must match the version number that serves as the name of the subdirectory containing the shared library's archive files in the `/shared-lib/`*`library_name`* directory.
- One or more `<code-source>` subelements, each containing a `path` attribute defining the path to a JAR or ZIP file belonging to the library.

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

The following example illustrates a shared library definition in the `server.xml` configuration file for the example shared library. The code source paths are relative to the subdirectory containing the JAR or ZIP files within the `/shared-lib/acme.common` directory; therefore, only the archive file names are specified as `path` values.

```
<shared-library name="acme.common" version="2.5">
  <code-source path="acme-apis.jar">
  <code-source path="acmeImpl.jar"/>
</shared-library>
```

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

```
<shared-library name="acme.common" version="2.5">
  <code-source path="*" />
</shared-library>
```

Additionally, a shared library can be configured to import one or more other shared libraries. The `<shared-library>` element can optionally take one or more `<import-shared-library>` elements, each specifying a shared library to be imported by the shared library being configured. An imported shared library must also be installed and published on the OC4J host.

The following sample code causes the `acme.common` shared library to import the `xyz.log` shared library:

```
<shared-library name="acme.common" version="2.5">
  <import-shared-library name="xyz.log"/>
  <code-source path="acme-apis.jar"/>
  <code-source path="acmeImpl.jar"/>
</shared-library>
```

When a relative code-source path is encountered at runtime, the full path is constructed by concatenating the shared library directory (`/shared-lib`), the shared library name and the version number. For example, the preceding sample entries would resolve to these paths:

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

Configuring an Application to Import a Shared Library

Once a shared library is installed, you can configure applications to import it using one of the following options:

- Declare a dependency in an application's OC4J-specific `orion-application.xml` deployment descriptor.

  See "Declaring Dependencies in an Application's OC4J Deployment Descriptor" on page 3-14 for details.

- Declare a dependency using an application's `MANIFEST.MF` file.

  This is the standard J2EE mechanism for declaring dependencies on *installed* libraries. See "Declaring Dependencies in an Application's Manifest File" on page 3-15 for details.

- Force all applications deployed to the OC4J instance to use the shared library by declaring a dependency in `application.xml`, the configuration file for the `default` application. Because `default` is the parent of all other applications deployed to the instance, the shared library will be imported by these applications.

  See "Configuring All Deployed Applications to Import a Specific Shared Library" on page 3-15 for details.

- Configure the application to import the shared library at deployment time via the deployment tasks provided in Application Server Control Console. The deployment tasks mechanism updates the application's `orion-application.xml` file as outlined in this section.

  See the *Oracle Containers for J2EE Deployment Guide* for details on using this feature.

> **Note:** Declaring a dependency using any of these options makes the shared library required by the application. An error will result if the shared library has not already been installed and published within the OC4J instance.

Declaring Dependencies in an Application's OC4J Deployment Descriptor

A dependency can be declared by adding notations in the dependent application's `orion-application.xml` deployment descriptor.

Dependencies are declared by adding an `<imported-shared-libraries>` element in the application-specific `orion-application.xml` configuration file. This element takes one or more `<import-shared-library>` subelements, each specifying a shared library to import.

The `<import-shared-library>` element has the following attributes:

- `name`: The name of the shared library.
- `min-version` and `max-version`: These are optional attributes that enable you to specify a minimum or maximum version of the library to be specified for inclusion. To use the latest installed version of the library, do not specify a version number.

The following entry in `orion-application.xml` will import the `acme.common:2.5` shared library for use by the application:

```
<imported-shared-libraries>
  <import-shared-library name="acme.common" max-version="2.6"/>
</imported-shared-libraries>
```

Declaring Dependencies in an Application's Manifest File

The standard Java extension mechanism, also known as optional packages, can be utilized to declare an application dependency on a JAR or ZIP file within a shared library. This is the standard J2EE mechanism for declaring dependencies on installed libraries.

To use this mechanism, the dependent JAR or ZIP file must be declared as a named extension in its `MANIFEST.MF` file. This name is specified as the value of the `Extension-Name` attribute. For example, the following manifest entry defines `acme.common` as an extension:

```
Extension-Name: acme.common
Specification-Vendor: Acme, Inc
Specification-Version: 2.5
Implementation-Vendor-Id: com.acme
Implementation-Vendor: Acme, Inc
Implementation-Version: 2.5
```

The application that is dependent on the shared library then declares the dependency in its own manifest file. The following manifest attributes will cause the application to import the `acme.common:2.5` shared library. Note that the value of the *`name`*`-Extension-Name` attribute exactly matches the `Extension-Name` value specified for the JAR or ZIP file's manifest file:

```
Extension-List: acme
acme-Extension-Name: acme.common
acme-Implementation-Version: 2.5
```

For more information on declaring dependencies using manifest files, see the following Web site:

`http://java.sun.com/j2se/1.4.2/docs/guide/extensions/versioning.html`

Configuring All Deployed Applications to Import a Specific Shared Library

You can ensure that all applications deployed to an OC4J instance use the same version of a shared library by configuring the default application to import it. Because `default` is the parent of all applications deployed to OC4J, any shared libraries that `default` imports will also be imported by these applications.

The configuration is managed by adding the XML notations described in "Declaring Dependencies in an Application's OC4J Deployment Descriptor" on page 3-14 to the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application.xml` file, the configuration file for the `default` application.

The following entry in `application.xml` will ensure that all deployed applications use version 2.0 of the `acme.common` shared library:

```
<imported-shared-libraries>
  <import-shared-library name="acme.common" max-version="2.0"/>
</imported-shared-libraries>
```

Note that an application will use the version of a shared library imported from the `default` application—*even if the application includes its own version of the shared library*. If this is not desirable, you can "remove" the version imported from default using the process described in "Removing or Replacing an Oracle Shared Library Imported by Default" on page 3-7.

Sharing Libraries Using the applib Directory

The legacy mechanism for sharing libraries across applications within OC4J, in which JAR or ZIP files to be shared are installed in the *ORACLE_HOME*/j2ee/*instance*/applib directory, is still supported in the current release of OC4J. All JAR and ZIP files within this directory will be included in the global.libraries shared library, and will be available to all applications within the OC4J instance.

Support for this legacy functionality will be removed in a future release of OC4J. Oracle recommends that you use the new shared-library mechanism documented in this chapter whenever possible.

Specifying a Library Directory in application.xml

The <library-directory> element of the application.xml file 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.

If the application.xml file for an application has the version="5" attribute set (JavaEE 5 application), the <library-directory> element of the .ear file's deployment descriptor can contain the name of a library directory through with application components can share libraries.

```
<application version="5">
    <library-directory>app2lib</library-directory>
    <module>
        <ejb>ejb.jar</ejb>
    </module>
</application>
```

If a <library-directory> element is not specified, or if the .ear file does not contain a deployment descriptor, the directory named lib is used.

You can use an empty <library-directory> element to specify that there is no library directory. For example:

```
<application version="5">
    <library-directory></library-directory>
    <module>
        <ejb>ejb.jar</ejb>
    </module>
</application>
```

The <library-directory> element provides a standard mechanism to define class path dependencies, as defined in the JavaEE 5 specification. In OC4j 10*g* (10.1.3.1.0), all files in a library directory with a .jar extension (but not in subdirectories) are available to all components packaged in the EAR file except application clients. Libraries in these JAR files can reference other libraries, either bundled with the application or installed separately.

OC4j 10*g* (10.1.3.1.0) also supports the <library-directory> element in J2EE 1.4 applications, with the following caveats:

- There will be no default lib directory for 1.4 applications, to prevent unexpected changes in behavior to existing applications. A user must explicitly add the <library-directory> element to an application.xml file to enable this support.

- A 1.4 application that adds the `<library-directory>` element cannot have XML validation turned on if it is using the 1.4 schema.

The proprietary `<library>` element in the `orion-application.xml` file provides the same functionality as `<library-directory>`.

Applications that use the `<library-directory>` element or JavaEE 5 applications that do not use an empty `<library-directory>` element will have an extra deployment step that iterates over files in the library directory and adds them to the class path.

Best Practices for Class Loading

This section provides guidelines for avoiding class loading issues.

Declare Class Dependencies

Make dependencies explicit in the application's `MANIFEST.MF` file or `orion-application.xml` file. Hidden or unknown dependencies will be left behind when you move your application to another environment.

Group Dependencies Together

Ensure that all dependencies are visible at the same level or above. If you must move a library, make sure all dependencies are still visible. Ensure that application resources, dependent third-party libraries and other enterprise modules are packaged in a self-contained manner.

Share Rather Than Duplicate Libraries

Avoid duplicating libraries, which increases both the disk and memory footprints and can lead to version problems.

Minimize Library Visibility

Dependency libraries should be placed at the lowest visibility level that satisfies all dependencies.

Keep Your Configurations Portable

Choose configuration options in the following order:

1. Standard J2EE options
2. Options that can be expressed within your EAR file
3. Server-level options
4. J2SE extension options

Be Sure to Use the Correct Class Loader

If you use reflection by calling `Class.forName()`, always explicitly pass the class loader returned by `Thread.currentThread().getContextClassLoader`. If you are loading a properties file, use the following:

```
Thread.currentThread().getContextClassLoader().getResourceAsStream()
```

Troubleshooting Class-Loading-Related Problems in OC4J

OC4J provides features to help you troubleshoot class loading-related issues.

- Overview of Troubleshooting

Overview of Troubleshooting

Most class loading errors in Java are related to visibility—either not enough or, more rarely, too much. Visibility in this case refers to the set of classes and resources that are available on the *class path*, which is the search path across a set of loaders and the code sources they contain (for example, JAR files, ZIP files, and directories).

In JavaSE, the class path is usually thought of as the list of code sources specified on the command line or in the manifest file for the main JAR file. These code sources are all deployed in a single loader, usually referred to as the *system* loader. This loader is in turn wired up to two other loaders: the JRE extensions loader (normally for any JAR files in the `jre/lib/ext` directory) and the JRE bootstrap loader (containing `rt.jar` and so on)

The idea that there is *a* class path is often misleading. For each class lookup, the search begins at a specific loader, and (normally) can visit only loaders *above* the class. This means that a search starting at the extensions loader would have a very different class path than one starting at the system loader.

In JavaEE, the situation is substantially more complicated, as multiple class loaders are required in place of the single system loader. Each deployed application has at least one loader, separate from all other applications, so each application has a distinct class path. There are frequently different class paths even within a single application, as each Web module is deployed in a separate loader.

This can easily be seen by making the following call from within different modules, as OC4J ensures that the correct class path is returned for the calling module:

```
System.getProperty("java.class.path");
```

There are many configuration options that affect class visibility for applications. Table 3–1 lists the most common options within OC4J. Given the possible combinations of loaders and code sources in a JavaEE environment, it is easy to see how visibility errors can arise. Often, a simple configuration change is all that is required to eliminate the problem, but understanding what change to make can be tricky.

Table 3–1 Configuration Options Affecting Class Visibility

Class Loader	Configuration Option
Configured shared library	`<code-source>` in `server.xml`
	`<import-shared-library>` in `server.xml`
app-name`.root`	`<import-shared-library>` in `orion-application.xml`
	`<library>` JAR files, ZIP files, and directories in `orion-application.xml`
	`<library-directory>` JAR files, ZIP files, and directories in `application.xml`
	`<ejb>` JAR files and ZIP files in `orion-application.xml`

Table 3–1 (Cont.) Configuration Options Affecting Class Visibility

Class Loader	Configuration Option
	RAR file: all JAR and ZIP files at the root.
	RAR file: `<native-library>` directory paths
	Manifest class path of preceding JAR and ZIP files
app-name.web.web-mod-name	WAR file: Manifest class path
	WAR file: `WEB-INF/` classes
	WAR file: `WEB-INF/lib/` all JAR and ZIP files
	<classpath> all JAR files, ZIP files, and directories in `orion-web.xml`
	Manifest class path of preceding JAR files
	`search-local-classes-first` attribute in `orion-web.xml`
	Shared libraries are inherited from the application root.

Class Loading Errors and Exceptions

Most class loading errors in J2EE often surface as one of a small set of exceptions. The OC4J class-loading infrastructure provides "annotated" subclasses for each of the following standard class-loading configuration exceptions:

- ClassNotFoundException
- NoClassDefFoundError
- ClassFormatError
- LinkageError
- ClassCastException

The subclasses enhance the output of the `getMessage()`,`printStackTrace()` and `toString()` methods with information to help you understand and correct the configuration. Often, this extra information is all that is needed to fix the problem; when it isn't, the logging, tracing, and query features can be used to dig further.

For each error, a diagnostic search may be performed to provide additional information. This search is not restricted by standard loader visibility rules, and can visit any loader or code source known to the system. Results are reported as part of the error message.

Under some conditions it would be possible for the loader runtime to act on the results and attempt recovery (such as automatically find and load a missing class). However, this might lead to further problems that would then be more difficult to diagnose.

ClassNotFoundException

This exception can occur during dynamic loading via any method that explicitly loads a class by name, such as `Class.forName()` or `ClassLoader.loadClass()`. It indicates that a required class is not visible from the initiating class loader.

How to troubleshoot and resolve

`ClassNotFoundException` is nearly identical to `NoClassDefFoundError`. One important difference is that the initial loader can be selected by the calling code, rather than by the JVM, and it is relatively easy to get this wrong.

If the `Class.forName(String)` method is used, the JRE code will select the initial loader and will always choose the caller's loader. This is rarely correct; it is nearly always better to explicitly pass the thread context loader:

```
Thread thread = Thread.currentThread();
ClassLoader loader = thread.getContextClassLoader();
Class cls = loader.loadClass(name);
```

Note that calling the loader directly is preferred to using the `Class.forName()` variant:

```
Class cls = Class.forName(name, true, loader);
```

While these two calls both try to load from the specified loader, the direct call is easier to understand. It also ensures that tracing works as expected, since some VM implementations of `forName()` can bypass the loader (only calling the loader if the class is not already cached—the loader always consults the cache itself, so calling the loader directly is safe and efficient).

The `LoadClass` query can be useful to experiment with loading classes from different initial loaders.

If the correct initial loader was used, then a code source is likely missing in the search path. See "NoClassDefFoundError" on page 3-20 for more detail.

NoClassDefFoundError

This exception can occur when the VM attempts to resolve a static dependency from one class to another. It is a type of LinkageError that indicates that the required class is not visible from the initiating loader. Since static dependency resolution is often deferred until first use, this error can occur at unexpected times.

Example Error Message

```
Missing class: acme.Dynamite

Dependent class: acme.RoadRunner
Loader: acme.root:0.0.0
Code-Source: /myapps/acme/acme.jar
Configuration: <ejb> in /myapps/acme/application.xml

The missing class is available from the following locations:

1.Code-Source: /shared/bang/0.0.0/bang.jar (from <code-source>
in j2ee/home/server.xml)

This code-source is available in loader bang:0.0.0. This
shared-library can be imported by the "acme" application.
```

How to troubleshoot and resolve

Examine the error message. The first line names the missing class. The next four lines describe the class that has the dependency: its name, the loader that defined it (also the "initiating" loader, selected by the VM), the code source from which it came, and the configuration option that caused the code source to be added to that loader. Subsequent lines describe the result of the diagnostic search, and will likely provide enough detail to resolve the issue.

Some common conditions reported by the diagnostic search are listed below, each with specific suggestions for resolution:

- The missing class is not present in any code source visible to the system.

  This often simply means that another code source must be added (which in turn may require others). Use the dependent class information to choose the right level at which to add the configuration, and then select a convenient option (see Table 3–1, " Configuration Options Affecting Class Visibility"). Consider creating a new shared-library if other applications are likely to need the same classes. Note that once created, the new shared-library must also be imported.

  This result can also mean that an existing code source declaration is invalid. When a path is encountered that does not point to a valid file or directory, this fact is logged, but at a level that is normally masked. To see all messages, start the system with the following setting:

  ```
  -Dclass.load.log.level=finest
  ```

  Look for messages about *nonexistent* code sources, find the relevant one and correct the configuration. There are often may of these messages, so it may be helpful to direct the output to a file (for example, "`loader.log`") that can be more easily searched:

  ```
  -Dclass.load.log.file=loader.log
  ```

- The missing class is present in a shared library, but that shared library was not imported.

  Import the shared library, either through Application Server Control Console or by adding an `<import-shared-library>` element to the application.

- The missing class is present, but the loader configured to use it is a child of the initiating loader.

  This most often occurs when a class deployed in an application root loader (for example, an EJB) has a dependency on a class deployed in a Web module. Resolving requires either refactoring to eliminate the dependency or moving the classes into the same loader. Often, the simplest solution is to move such classes from the Web module up to the application root, but the reverse may also be possible. See Table 3–1, " Configuration Options Affecting Class Visibility" on page 3-18 for options to add code sources to the root.

  Note that copying rather than moving such classes can lead to a `ClassCastException` when the `search-local-classes-first` option in the web module is enabled.

- The missing class is present, but the loader configured to use it is in an unrelated application.

  An ideal solution in this case is to move the relevant code sources out of the existing application into a new shared library and reconfigure both applications to import it. When this is not practical, the code source may be copied into the current application. While it is possible to use configuration to point directly at the code source in the other application, this will cause failures if that application is ever undeployed.

  The `ClassPath` query can be useful to examine code-source search order from a specific loader:

  ```
  -Doc4j.start.query="ClassPath(acme.root)+Exit"
  ```

  The `SharedLibraries` query may also be used to list all shared libraries and the loaders that import them.

ClassFormatError

This error can occur when a class is first loaded (during definition). It is a type of LinkageError, and often indicates that the class was compiled for a different version of the JVM.

How to troubleshoot and resolve

The error message will specify the version with which the class was compiled, and the version supported by the current runtime.

To correct the failure, either switch to the correct version of JVM, or re-compile the class for the current one.

LinkageError

This error can occur when a class is first loaded (during definition). The more common LinkageError subtypes are processed as special cases by the OC4J class loading system (see NoClassDefFoundError and ClassFormatError) -- the remaining cases generally indicate one of the following:

- A mismatch between the versions of a class used at compile time and that found at runtime
- A native library is required but cannot be found.

How to troubleshoot and resolve

The error message will specify the actual failure, and will also provide the name, defining loader, code source, and configuration for two different classes: the one that failed during definition, and the class that has the dependency that caused the definition to occur.

If the message indicates a version mismatch of some sort, such as `NoSuchMethodError`, source level changes and re-compilation will be required to resolve the failure. Frequently, the mismatch occurs between a superclass or interface and a subclass, and should be relatively easy to discern.

An `UnsatisfiedLinkError` means that a native library could not be found. Generally, OC4J only supports configuring native libraries within RAR modules (see Table 3–1, " Configuration Options Affecting Class Visibility" on page 3-18). If the library is specified within a RAR, it may be an invalid path. See "NoClassDefFoundError" on page 3-20 for a discussion of using the `class.load.log.level` property to detect this case.

ClassCastException

This exception usually occurs for obvious reasons, such as:

```
Object source = new Integer(0);
String target = (String) source; // Exception
```

However, this exception can also be class loading related. Unfortunately the error message created by the VM is normally empty, and even when this exception is loading related, it cannot be intercepted and annotated.

When more than one loader defines a class with the same name, a cast between the two types will fail with a `ClassCastException`:

```
import com.acme.Foo;
...
// Get the loader that resolves our static dependency
// on class Foo
```

```
ClassLoader expected = Foo.class.getClassLoader();

// Dynamically load Foo from another loader and
// create an instance

Class fooClass = aLoader.loadClass("com.acme.Foo");
Object source = fooClass.newInstance();
ClassLoader actual = fooClass.getClassLoader();

// Compare and cast

System.out.println(actual == expected); // "false"
Foo target = (Foo) source; // Exception
```

In this case, the loader instances are different, and so the JVM considers the two classes to be unrelated.

A relatively common example of this problem happens within a single application when an EJB interface is packaged both in the EJB module and in a Web module that uses the EJB. If the `search-local-classes-first` option is enabled for the Web module, the EJB classes will be loaded twice.

How to troubleshoot and resolve

First, determine the target type of the cast by looking at the code described at the top of the stack trace. If the cast is to a primitive type (such as int or long) or to a class defined by the JRE (for example, String, HashMap, or any other java.* class), then it is almost certainly not class loading related and is a (usually simple) developer error.

Next, use tracing to see definition(s) of the target class. For example, if the target of the cast is `com.acme.Foo`, use:

```
-Dclass.load.trace=class-defined:com.acme.Foo
```

The output will describe the loader and the code source from which the class is defined. If there is only one, then the problem is unlikely to be class loading related (though it is still possible if custom class loaders are in use, as they won't participate in tracing).

If there is more than one definition of the target class, then it is very likely that they are coming into contact and causing the exception. If the trace messages list the same code source for both definitions, then it is being shared across loaders and should be easy to re-arrange.

However, it is far more common that the code sources will be different, perhaps because classes were repackaged for convenience (nearly always a bad idea), or because they are different versions. If one of the loaders is from a web module, disabling search-local-classes-first may be sufficient.

If the code can be instrumented, duplication can be confirmed by adding code just before the cast (where `obj` is the object being cast):

```
ClassLoader expected = Foo.class.getClassLoader();
System.out.println("Expected: " + expected);
ClassLoader actual = obj.getClass().getClassLoader();
System.out.println("  Actual: " + actual);
```

The output should agree with the loaders named in the tracing messages. Note that if `obj` is a subclass of the expected type, then walking the hierarchy with `Class.getSupertype()` may be required.

If duplication is confirmed as the cause, it must be eliminated. This can be accomplished by arranging for the class to be shared: either move the class to a common parent of the two loaders or to a shared-library. To help determine which loader should continue to load the class, it may be helpful to see the call stack at the point of each class definition:

```
-Dclass.load.trace=class-defined:com.acme.Foo+stack
```

It may also be useful to see the relationships between loaders using the `LoaderTree` query:

```
-Doc4j.start.query=LoaderTree
```

Queries can also be executed on a running instance using the ASControl system MBean browser. The `-verbose` option can be used with this query to see lots of detail:

```
-Doc4j.start.query="LoaderTree(-verbose)+Exit"
```

The `DuplicateClasses` query can also be used to search for potential duplicates (same class names in different code sources).

Using Queries to Troubleshoot Class Loading Issues

OC4J provides a number of built-in queries that can be run to troubleshoot class loading problems. Queries can be executed at OC4J startup by setting the `oc4j.start.query` system property, or at runtime through the `ClassLoading` MBean.

- Summary of Available Class Loader Queries
- Executing Queries at OC4J Startup
- Executing Queries at Runtime Through the ClassLoading MBean

> **Note:** This feature is subject to change in future releases of OC4J.

Summary of Available Class Loader Queries

The following built-in queries are provided with OC4J to assist in troubleshooting class loading problems.

Note that the examples used illustrate usage of the `oc4j.start.query` system property, which is set on the `oc4j.jar` command line at OC4J startup.

The available queries are as follows:

- AuditLoader
- Callers
- ClassLoadMetrics
- ClassPath
- Dependencies
- Depends
- DuplicateClasses
- DuplicateCodeSources
- Exit
- FindResource

- GetResource
- HttpSessions
- LeakedLoaders
- ListQueries
- LoadClass
- LoadedClasses
- LoaderTree
- Packages
- SharedLibraries
- SystemProperties
- ThreadPools
- Threads
- UnusedCodeSources
- Uptime
- VMStat

AuditLoader

Performs various diagnostics on the contents of one or more specified class loaders or on all shared class loaders. For example, to audit all shared class loaders with detailed output, specify the `AuditLoader` query as follows:

```
AuditLoader * -verbose
```

Arguments: *`loaderName`* [*`loaderName`*]... `[-verbose] [-includeJREClasses]`

Table 3–2 AuditLoader Query Arguments

Argument	Description
`loaderName`	The name of a class loader to audit. To audit all shared class loaders, specify '*' for *`loaderName`*.
`-verbose`	Optional. Specify to output detailed information.
`-includeJREClasses`	Optional. Specify to include JRE classes.

Callers

Reports all classes that call the specified method(s). The query optionally takes the name of a class loader to limit the query to.

For example, to find all classes that call `System.currentTimeMillis()`, execute the following:

```
-Doc4j.start.query=Callers(java.lang.System.currentTimeMillis()long)
```

The next example will find the same method in classes visible from the `MyApp.root` class loader:

```
-Doc4j.start.query=Callers(-MyApp.root,java.lang.System.currentTimeMillis()long)
```

To find calls to either version of the overloaded `Class.forName()` method, pass in both method signatures as arguments:

```
-Doc4j.start.query=Callers(java.lang.Class.forName(java.lang.
String)java.lang.Class,java.lang.Class.forName(java.lang.String;boolean;java.lang.
ClassLoader)java.lang.Class)
```

Note that you can include "<init>" in the <signature> argument to specify a constructor:

```
-Doc4j.start.query=Callers(java.util.Date.<init>())
```

Arguments: [-*loaderName*] *signature*...

Table 3–3 Callers Query Arguments

Argument	Description
-loaderName	Optional. If specified, all classes visible from the class loader will be checked for the specified methods. If not specified, all classes visible to all class loaders down the tree from the `api` class loader, the default parent of all application-specific class loaders, are checked.
signature	A method to check for. Multiple methods may be specified, each as a separate argument. The syntax is: *class-type.method-name*[parameter-type]*)*return-type* Separate multiple parameter types using semicolons (`;`). Types must be either fully qualified class names or primitive names. For `void` return types, the *return-type* may be omitted.

ClassLoadMetrics

Reports metrics for one or more specified class loaders or for all class loaders if none is specified. For example:

```
-Doc4j.start.query=ClassLoadMetrics(MyApp.root,MyOtherApp.root)
```

Arguments: `[-verbose] [<loaderName>]...`

> **Note:** The verbose (multiline) output of the `PolicyClassLoader.toString()` method has been turned off by default in OC4J 10*g* (10.1.3). Only the loader name and version are now returned.
>
> You can reenable verbose output by setting either of these system properties:
>
> - `-Dclass.load.trace=`*any value*
> - `-Dverbose.loader.tostring=true`

Table 3–4 ClassLoadMetrics Query Arguments

Argument	Description
`-verbose`	Optional. Supply to generate detailed metrics.
`loaderName`	Optional. The name(s) of the class loader(s) to report metrics for.

ClassPath

Reports the code sources in use.

Arguments: `[-list] [loaderName]`

Table 3–5 ClassPath Query Arguments

Argument	Description
`-list`	Optional. Include to generate a line-separated, numbered list of code sources.
`loaderName`	Optional. If specified, the class loader is used as the starting point from which the classpath is computed. Otherwise, the classpath defaults to the internal `oc4j` class loader, which loads all OC4J system classes.

Dependencies

Reports all dependencies of the specified class.

Arguments: `<className> [loaderName] [-r]`

Table 3–6 Dependencies Query Arguments

Argument	Description
`className`	The fully qualified name of the class to report dependencies for.
`loaderName`	Optional. If specified, the class loader is used as the starting point from which dependencies are determined. Otherwise, the the internal `oc4j` class loader, which loads system classes, is used.
`-r`	Optional. Set to search classes recursively. Note that use of this option can cause long execution times.

Depends

Reports on all classes that are dependent on the specified class.

Arguments: `[-loaderName] <className>`

Table 3–7 Depends Query Arguments

Argument	Description
`className`	The fully qualified name of the class to check for dependencies. To specify an entire package, use an asterisk (`*`) as the leaf name.
`loaderName`	Optional. If specified, all classes visible to the class loader will be checked. Otherwise, all classes visible to all class loaders from the `api` class loader downwards are checked.

DuplicateClasses

Reports the existence of classes and resources with the same name in different code sources.

Arguments: `[-loaderName] [-systemCodeSources]`

Table 3–8 DuplicateClasses Query Arguments

Argument	Description
`-loaderName`	Optional. If specified, the class loader is used as the starting point for the code-source search. Otherwise, all classes visible to all class loaders from the `api` class loader downwards are checked.
`-systemCodeSources`	Optional. If specified, includes system code sources in the search.

DuplicateCodeSources

Reports code sources that have the same name or have more than one subscriber.

Arguments: `[-digest]`

Table 3–9 DuplicateCodeSources Query Arguments

Argument	Description
`-digest`	Optional. If specified, bit-wise comparisons of code sources will be performed.

Exit

Exits the process and shuts down OC4J if running. This is useful if you only want to execute the query without leaving the OC4J server running. For example:

```
-Doc4j.start.query=LoaderTree+Exit
```

Arguments: `[-force]`

Table 3–10 Exit Query Arguments

Argument	Description
`-force`	Optional. Forces a `System.exit()` call instead of a normal shutdown.

FindResource

Reports the code sources containing the specified resource, identified either by classname or resource path.

A leading or trailing asterisk * can be used for a simple wildcard search, or a leading ~ may be used to indicate that the argument should be treated as a regular expression.

The `resourcePath` argument must be passed to search for classes with no package. For example, to find class foo, the query would be:

```
-Doc4j.start.query=FindResource(Foo.class)
```

To search for resources with no package (`/`), where the resource name contains a period (`.`), a leading asterisk must be used. For example, use the following query to find the `myconfig.xml` file at the root of a code source:

```
-Doc4j.start.query=FindResource(*myconfig.xml)
```

Arguments: `[-list] <resourcePath>|<className>`

Table 3–11 FindResource Query Arguments

Argument	Description
`-list`	Optional. Can be used with wildcard or regular expressions to list all matching resources.
`resourcePath`	The fully qualified path to the resource. Either `resourcePath` or `className` must be supplied.
`className`	The fully qualified classname of the resource to search for. Either `resourcePath` or `className` must be supplied.

GetResource

Calls `getResource()` or `getResources()` on a specified loader and reports the results.

Arguments: `resourcePath [loaderName] [-all]`

Table 3–12 GetResource Query Arguments

Argument	Description
`resourcePath`	The fully qualified path to the resource.
`loaderName`	Optional. If specified, the class loader is used as the starting point for the resource search. Otherwise, all classes visible to all class loaders from the `api` class loader downwards are checked.
-all	If specified, uses `getResources()`. Otherwise, uses `getResource()`.

HttpSessions

Reports a summary of active HTTP sessions for deployed applications.

Arguments: `[details]`

Table 3–13 HttpSessions Query Arguments

Argument	Description
`details`	Optional. If specified, lists details of each HTTP session.

LeakedLoaders

Controls detection of class loader leaks and lists results.

Arguments: `[activate|list|deactivate]`

Table 3–14 LeakedLoaders Query Arguments

Argument	Description
`activate`	Optional. If specified, activates detection of class loader leaks.
`list`	Optional. If specified, lists results of class loader leak detection.
`deactivate`	Optional. If specified, deactivates detection of class loader leaks.

ListQueries

Lists all of the available queries in OC4J, by `oracle.oc4j.query.Query` subclass name.

Arguments: `[-l] [queryclass]...`

Table 3–15 ListQueries Query Arguments

Argument	Description
`-l`	Optional. If specified, lists full descriptions.
`queryclass`	Optional. If specified, lists each query with a single-line description. Pass one or more query class name to list only those queries.

LoadClass

Attempts to load the specified class using the specified class loader, or using the internal `oc4j` class loader if none is specified, and reports the result. The query in effect performs a class-loading *test run*.

Arguments: `<className> [loaderName] [-forName] [-depends] [-r] [-sort]`

Table 3–16 LoadClass Query Arguments

Argument	Description
`className`	Required. The fully qualified name of the class to report on.
`loaderName`	Optional. If specified, attempts to load the class using this class loader are reported. If not specified, the `oc4j` class loader is used by default.
`-forName`	Optional. The class loader method to record attempts for. If not specified, `loader.loadClass()` is used by default.
`-depends`	Optional. Specify to force the class loader to load and report on all dependencies of the specified class.
`-r`	Optional. Specify to load all dependencies recursively. Note that recursion does not include classes in `java.*` packages.
`-sort`	Optional. Include to sort the list of dependent classes by class name.

LoadedClasses

Lists the names of all loaded classes (if available).

Arguments: None.

LoaderTree

Reports the contents of the class loader tree for the specified root class loader, or for the JRE's bootstrap class loader if none specified. By default, only the names of the class loaders within the tree are reported.

This query is useful if you want to focus on a specific part of the class loader tree, such as from the root of a specific application downwards. For example:

```
-Doc4j.start.query=LoaderTree(MyApp.root)
```

To retrieve the entire class loader tree, do not include a root class loader name:

```
-Doc4j.start.query=LoaderTree
```

Arguments: `[rootLoaderName] [-verbose]`

Table 3–17 LoaderTree Query Arguments

Argument	Description
`rootLoaderName`	Optional. The root class loader name. If not specified, reports the contents of the JRE's bootstrap class loader tree.
`-verbose`	Optional. Specify to output detailed information. If not specified, only the names of the class loaders within the tree are reported.

Packages

Lists the package names contained within one or more code sources.

Arguments: `[loaderName | codeSourcePath]`

Table 3–18 Packages Query Arguments

Argument	Description
`loaderName`	Optional. If specified, only code sources within that class loader are searched.
`codeSourcePath`	Optional. If specified, searches the code source path. Otherwise, searches all available code sources.

SharedLibraries

Lists all installed shared libraries and the class loaders that import each.

Arguments: `[loaderName]`

Table 3–19 SharedLibraries Query Arguments

Argument	Description
`loaderName`	Optional. The name of a class loader. If specified, lists shared libraries imported by the specified class loader. Otherwise, the shared libraries imported by all class loader instances are listed.

SystemProperties

Lists or sets system properties.

Arguments: `[<key>=<value>]...`

Table 3–20 SystemProperties Query Arguments

Argument	Description
`<key>=<value>`	Optional. The name and value of a system property. If specified, sets the value of the system property. If no key-value pairs are specified, lists the current values of system properties.

ThreadPools

Lists information about application-server-related thread pools for the current JVM.

Arguments: `(list [<sys|req|cx>]|pool [<sys|req|cx>]|state [<sys|req|cx>][<id>)`

Table 3–21 ThreadPools Query Arguments

Argument	Description
`list [<sys|req|cx>]|`	If specified, lists all threads in all thread pools or all thread in a specified thread pool.
`pool [<sys|req|cx>]|`	If specified, lists all thread pools or details for a specified thread pool.
`state [<sys|req|cx>][<id>)`	If specified, dumps thread state for all thread pools, for a specified thread pool, or for a specified thread ID.

Threads

Lists thread information for the current JVM. For JDK 1.5, checks for thread deadlocks or gets thread memory usage for the current JVM.

Arguments: `[list|groups|deadlock|memory]`

Table 3–22 Threads Query Arguments

Argument	Description
`list`	Optional. If specified, lists all threads for the current JVM.
`groups`	Optional. If specified, lists all thread groups for the current JVM.
`deadlocks`	Optional, for JDK 1.5 only. If specified, checks for thread deadlocks in the current JVM.
`memory`	Optional, for JDK 1.5 only. If specified, gets thread memory usage for the current JVM.

UnusedCodeSources

Reports code sources that have never retuned any data.

Arguments: None

Uptime

Reports the length of time the current OC4J server instance has been running.

Arguments: None

VMStat

Lists statistics information for the current JVM.

Arguments: `[jps|pid|mem]`

Table 3–23 VMStat Query Arguments

Argument	Description
`jps`	Optional. If specified, Lists the running Java processes.
`pid`	Optional. If specified, returns the PID of the current JVM.
`mem`	Optional. If specified, returns memory information about the current JVM.

Executing Queries at OC4J Startup

Queries can be executed at OC4J startup by setting the `oc4j.start.query` system property on the `oc4j.jar` command line.

Query results are written to the file specified as the value of the `class.load.log.file` property, or to the console (`System.out`) if no file is specified. See "Setting Class Loader Log Levels" on page 3-38 for details on the `class.load.log.file` property.

Note that system properties must be prefaced on the command line with a `-D`. The syntax is as follows:

```
java -Doc4j.start.query=<queryName>(arg0,arg1,...) -jar oc4j.jar
```

Arguments can be passed in with the query by appending `(arg0,arg1,...)` to the query name. Note that arguments are enclosed in parentheses. Multiple arguments are separated by a comma.

Note: For some UNIX shells, the property value string—everything after the = symbol—will need to be quoted.

For example, the `DuplicateCodeSources` query can be invoked with a `-digest` argument as follows:

```
java -Doc4j.start.query=DuplicateCodeSources(-digest) -jar oc4j.jar
```

Multiple queries may be specified by separating them with the + character:

```
-Doc4j.start.query=DuplicateCodeSources(-digest)+UnusedCodeSources
```

Executing Queries at Runtime Through the ClassLoading MBean

Queries can be executed on a running OC4J instance by calling the `executeQuery` operation on the `ClassLoading` MBean.

This MBean is accessible through the Web-based Application Server Control Console interface. See the *Oracle Containers for J2EE Configuration and Administration Guide* for details on accessing and using the MBeans packaged with OC4J.

1. Click the **Administration** link in the Application Server Control Console.
2. Click **System MBean Browser**.
3. Expand the `ClassLoading` node in the navigation pane, then select the `singleton` MBean instance.
4. Click the **Operations** tab in the right-hand pane, then click the `executeQuery` operation.

Note: Two versions of the `executeQuery` operation are exposed. Click the version that takes *two* parameters. (The `queryClassData` parameter cannot be set through the System MBean Browser.)

---

5. Enter the name of the query you want to execute as the value for **queryClassName**. For example, `LoaderTree`.
6. Click the **queryArguments** icon, then add a new row for each argument you want to specify. Do NOT enclose arguments in parentheses; these are added automatically when the operation is invoked. Click **OK** when finished specifying arguments.
7. Click the **Invoke** button to call the operation.

Tracing Class Loading Events to Help Troubleshoot Issues

OC4J provides the `class.load.trace` system property that can be set to trace class loading, class loader lifecycle, and code-source lifecycle events. The tracing output generated can be extremely useful in troubleshooting class loading-related issues.

Note: This feature is subject to change in future releases of OC4J.

The `class.load.trace` property is set at OC4J startup. The syntax follows:

```
class.load.trace=<event>[[:<string-filter>[,<string-filter>]] |
 [~<pattern-filter>]]
```

- `<event>` is the event to trace. See Table 3–24 on page 3-35 for valid values.

  Multiple event values can be strung together with a + character.

  ```
  -Dclass.load.trace=class+loader
  ```

- `<string-filter>` is a filter applied to manage trace output. Note that a colon (`:`) separates the initial filter from the event. See "Using Filters to Manage Trace Output" on page 3-36 for details on the types of filters that can be applied to manage trace output.

  Multiple filters can be separated by a comma:

  ```
  -Dclass.load.trace=class:com.acme.Foo,com.acme.Bar
  ```

- `<pattern-filter>` is a single filter that is interpreted as a regular expression. Note that a ~ character precedes the filter.

Notes:

- System properties must be prefaced on the command line with a `-D`. For example:

  ```
  java -Dclass.load.trace=loader -jar oc4j.jar
  ```

- In a standalone OC4J configuration, system properties are set directly on the `oc4j.jar` command line, as shown in the example above.
- In an Oracle Application Server configuration, system properties are set in the `<data>` element where the `id` attribute is `"java-options"` in the `opmn.xml` file for the OC4J instance. For example:

  ```
  <data id="java-options" value="-Dejb3=true
   -Dclass.load.trace=class+loader"/>
  ```

Trace output is written to the console by default, but can be written to a file specified using the `class.load.log.file` system property. For example:

```
java -Dclass.load.log.file=C:\logs\logfile.txt -jar oc4j.jar
```

Table 3–24 class.load.trace System Property Values

Value	Description
`all`	Activate all tracing modes. Note that setting this value may slightly impact OC4J performance, due to the volume of output.
`none`	Disable all tracing modes.
`class`	Trace all class loading search events.
`class-defined`	Trace all events in which the specified class is initially loaded by a class loader. The object is then cached for subsequent use by class loaders.
`class-found`	Trace all search events where the specified class was found.
`class-not-found`	Trace all search events where the specified class was not found.
`code-source`	Trace all code source lifecycle events.
`code-source-create`	Trace events where a code source object is first initialized. Only one instance of a code source is created in memory; this object is then shared by all class loaders that need it.
`code-source-dependency`	Trace events for all code sources where extension dependencies declared in the `MANIFEST.MF` file packaged within the archive were found or not found.
`code-source-dependency-satisfied`	Trace events for all code sources where all extension dependencies declared in the `MANIFEST.MF` file packaged within the archive were found.
`code-source-dependency-not-satisfied`	Trace events for all code sources where all extension dependencies declared in the `MANIFEST.MF` file packaged within the archive were not found.
`code-source-manifest`	Trace events for all code sources where a `MANIFEST.MF` file is packaged within the archive and any class paths, extension declarations, etc. are being processed.
`code-source-state`	Trace events for two the following code source states: ■ Open: Code source is actively being searched for or used by a class loader. ■ Closed: Code source is not currently being used by a class loader and is basically in a passivated state.
`code-source-destroy`	Trace events where a code source object is being destroyed.
`loader`	Trace all class loader lifecycle events.
`loader-create`	Trace class loader object instantiation events.
`loader-commit`	Trace events where a class loader object has been created and populated with classes from code sources.
`loader-finalize`	Trace events where the `finalize()` method has been called on a class loader object and the object is no longer accessible.
`loader-close`	Trace events where a class loader object is in the process of being garbage collected by the JVM.
`loader-destroy`	Trace events where a class loader object is being destroyed.
`resource`	Trace all resource search events.
`resource-found`	Trace all resource search events where the resource was found.
`resource-not-found`	Trace all resource search events where the resource was not found.
`stack`	Add a stack trace to all events.
`help` or `?`	Print the help text to the console.

Using Filters to Manage Trace Output

The `class.load.trace` system property supports the use of filters to make event tracing output more manageable. The syntax is as follows. Note that a colon (:) separates the initial filter from the event to trace:

```
<event>[[:<string-filter>[,<string-filter>]] | [~<pattern-filter>]]
```

Table 3–25 describes the supported event filter types.

Table 3–25 Supported Trace Output Filters

Event	Supported Filters

`class`	■ **Exact match**
	Specify the fully qualified name of the class to trace events for. The following example will only trace loading of the `com.acme.Dynamite` class:
	`-Dclass.load.trace=class:com.acme.Dynamite`
	■ **Prefix or suffix match**
	Use a leading or trailing asterisk (`*`) to treat the string as a prefix or suffix. For example:
	`-Dclass.load.trace=class:com.acme.*`
	`-Dclass.load.trace=code-source:*foo.jar`
	■ **Regular expression match**
	Use a tilde (~) to treat the string as a regular expression. Note that the `.*` syntax indicates that any number of characters can match the expression.
	The following example will trace class loading events for class names containing "`util`":
	`-Dclass.load.trace=class~.*util.*`
	■ **Class loader match**
	Begin the filter string with `loader.` to treat the remainder of the string as a class loader name.
	The following example will trace loading only of classes performed by the `api` class loader, the default parent of all application-specific class loaders:
	`-Dclass.load.trace=class:loader.api`

Table 3–25 (Cont.) Supported Trace Output Filters

Event	Supported Filters

`code-source`	■ Full path
	Specify the full path for the code source to trace events for. For example:
	`-Dclass.load.trace=code-source:/C:/oc4j/xdk/lib`
	
	■ Partial path
	Use a leading or trailing asterisk (`*`) to treat the string as a prefix or suffix. This example will trace loading only of classes in the `com.acme` package:
	`-Dclass.load.trace=code-source:*/acme.jar`
	`-Dclass.load.trace=code-source:/C:/oc4j/*`
	
	■ **Regular expression match**
	Use a tilde (~) to treat the string as a regular expression. Note that the `.*` syntax indicates that any number of characters can match the expression.
	The following example will trace `code-source-create` events for the "`foo`" application:
	`-Dclass.load.trace=code-source-create~.*/foo/.*`

`loader`	■ **Exact match**
	Specify the complete name (`name.name:version`) of the class loader to trace events for. For example:
	`-Dclass.load.trace=loader:oracle.jdbc:10.1.0_2`
	
	■ **Suffix match**
	Use a trailing asterisk (`*`) to treat the string as a suffix. For example:
	`-Dclass.load.trace=loader:oracle.jdbc:*`
	
	■ **Regular expression match**
	Use a tilde (~) to treat the string as a regular expression. Note that the `.*` syntax indicates that any number of characters can match the expression.
	The following example will trace `create` events for application root class loaders:
	`-Dclass.load.trace=loader-create~.*root.*`

Table 3–25 (Cont.) Supported Trace Output Filters

Event	Supported Filters

`resource`	■ Full path
	Specify the full path for the resource. For example:
	`-Dclass.load.trace=resource:META-INF/services/`
	`javax.xml.parsers.DocumentBuilderFactory.`
	
	■ Partial path
	Use a leading or trailing asterisk (`*`) to treat the string as a prefix or suffix. This example will only trace loading of classes in the `com.acme` package:
	`-Dclass.load.trace=resource:*Messages_en.properties`
	`-Dclass.load.trace=resource:`
	`oracle/oc4j/admin/jmx/model/*`
	
	■ **Regular expression match**
	Use a tilde (~) to treat the string as a regular expression. Note that the `.*` syntax indicates that any number of characters can match the expression.
	The following example will trace all resource searches for property files with "`security`" in the path:
	`-Dclass.load.trace=resource~.*security.*properties`

Setting Class Loader Log Levels

By default, the class loader Logger messages are filtered at the `CONFIG` Java log level. If necessary, you can use the `class.load.log.level` system property to change the log level. Table 3–26 below lists the values that can be set on this property.

For example, to set the log level to `SEVERE`:

```
-Dclass.load.log.level=severe
```

Note that tracing-related messages are written at the `INFO` log level. As a result, avoid setting the log level above `INFO`, such as to `WARNING`, to prevent these messages from being filtered from the Logger output.

Table 3–26 class.load.log.level System Property Values

Value	Description
`all`	Output all log messages.
`severe`	Output messages at the `SEVERE` level only.
`warning`	Output messages at the `WARNING` level or above.
`info`	Output messages at the `INFO` level or above. If the log level is set below `info`, for example, to `finer`, trace output will be filtered from the Logger output.
`config`	Output messages at the `CONFIG` level or above. This is the default log level.
`fine`	Output messages at the `FINE` level or above.
`finer`	Output messages at the `FINER` level or above.
`finest`	Output messages at the `FINEST` level or above.
`off`	Suppress all logging messages.

4

Logging Implementation Guidelines

This chapter discusses the Oracle guidelines for implementing logging functionality in applications that will be deployed into OC4J. It enables applications that use the standard Java logging framework to integrate Java logging with Oracle Diagnostic Logging (ODL) and take advantage of log analysis tools provided by Oracle, as the following topics describe:

- Overview of the Java and Oracle Logging Frameworks
- Java Logging Guidelines
- Configuring Java Loggers to Use the ODL Framework

For information on logging configuration and usage in OC4J, see the *Oracle Containers for J2EE Configuration and Administration Guide*.

Overview of the Java and Oracle Logging Frameworks

The following section provides an overview of the Java and Oracle logging frameworks, and describes how they are integrated to enable Java log output to be generated in Oracle format.

The Java Logging Framework

The Java logging framework, introduced in JDK 1.4, provides extensive logging APIs through the `java.util.logging` package. For an overview of the `java.util.logging` package, visit `http://java.sun.com/j2se/1.4.2/docs/api/overview-summary.html`.

For an overview of the Java logging framework, visit Sun's site on the subject at `http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html`.

The Oracle Diagnostic Logging Framework

The *Oracle Diagnostic Logging* framework, or *ODL*, provides plug-in components that complement the standard Java framework to automatically integrate log data with Oracle log analysis tools. In the ODL framework, log files are formatted in XML, enabling them to be more easily parsed and reused by other Oracle Application Server and custom developed components.

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

ODL-formatted log files can be viewed through the Web-based Oracle Enterprise Manager 10*g* Application Server Control Console, allowing administrators to aggregate and view the logging output generated by all components and applications running within OC4J from one centralized location. See the *Oracle Containers for J2EE Configuration and Administration Guide* for instructions on viewing log files generated by an OC4J instance.

How Java Logging and Oracle Diagnostic Logging Work Together

In the Java logging framework, applications record events by making calls on Logger objects, which are instances of the `java.util.logging.Logger` class. A Logger is a named entity that is associated with a system or application component. Each Logger is assigned a specific log level, and records events only at that level of severity or higher.

Logging messages are forwarded to a Handler object, which can in turn forward the messages to a variety of destinations for publication. The `oracle.core.ojdl.logging` package includes a Handler class, `ODLHandler` class, which generates the Logger output in XML-based ODL format.

Java Logging Guidelines

The following topics provide guidelines for implementing Java Loggers that will integrate with the Oracle Diagnostic Logging framework.

Naming Java Loggers

Java Loggers are named entities, named using a hierarchical dot-separated namespace. The Logger namespace is global, and is shared by all applications running within OC4J. As such, ensure that each Logger name is unique to avoid potential naming conflicts.

Logger names should include the vendor name and component name, and optionally include the module or sub-module. Use the following convention for Logger names:

```
vendorName.componentName[.moduleName][.subModuleName]
```

For example:

```
acme.mycomponent.mymodule
```

Setting Log Levels

In the Java logging framework, log levels are represented by objects of the `java.util.logging.Level` class. This class defines seven standard log levels, ranging from SEVERE (the highest priority, with the highest value) to FINEST (the lowest priority, with the lowest value).

Ideally, your applications should utilize these predefined Java log levels, which Oracle diagnostic tools provided as part of OC4J map to Oracle Diagnostic Logging (ODL) message types and levels.

Table 4–1 illustrates the mapping between the predefined Java log levels and ODL message types and levels. The ODL log levels are between `1` and `32`, with a lower value indicating a higher severity or less volume of information.

Table 4–1 Mapping between Java log levels and ODL message types and log levels

Java Log Level	ODL Message Type:Log Level	ODL Description
`SEVERE.intValue()+100`	`INTERNAL_ERROR:1`	The program has experienced an error for some internal or unexpected non-recoverable exception.
`SEVERE`	`ERROR:1`	A problem requiring attention from the system administrator has occurred.
`WARNING`	`WARNING:1`	An action occurred or a condition was discovered that should be reviewed and may require action before an error occurs.
`INFO`	`NOTIFICATION:1`	A report of a normal action or event. This could be a user operation, such as "login completed" or an automatic operation such as a log file rotation.
`CONFIG`	`NOTIFICATION:16`	A configuration-related message or problem.
`FINE`	`TRACE:1`	A trace or debug message used for debugging or performance monitoring. Typically contains detailed event data.
`FINER`	`TRACE:16`	A fairly detailed trace or debug message.
`FINEST`	`TRACE:32`	A highly detailed trace or debug message.

The Oracle diagnostic tools provide some flexibility to accommodate custom log levels implemented with applications. However, containing log levels to the seven default Java levels (SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST) is recommended.

Adding Localization Support

Each `Logger` object can optionally have an associated `ResourceBundle` object which is used to localize log message strings.

If a `Logger` does not have an associated `ResourceBundle`, it will inherit the `ResourceBundle` name from its parent according to the classic class loader hierarchy, recursively up the tree.

Configuring Java Loggers to Use the ODL Framework

Enabling Java Loggers to output log messages in the ODL format is accomplished by mapping each Logger to the `ODLHandler`. This mapping is managed through a logging configuration file, `j2ee-logging.xml`, which is generated by OC4J in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` directory.

In OC4J 10*g* (10.1.3.1.0), you can set the log levels for loggers through the Application Server Control Console, 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.

You can also edit the `j2ee-logging.xml` configuration file by hand. Restart OC4J after making any changes to this file.

This configuration file contains two elements within the `<logging-configuration>` root element:

- `<log_handlers>`

  This element defines one or more Handlers within OC4J. It includes one or more `<log_handler>` elements, each defining the name of a Handler and the class that generates instances of it. By default, this element includes `<log_handler>` elements defining three different log handlers:

  – `oc4j-handler`

    This is the log handler for the `oracle` logger.

  – `oracle-webservices-management-auditing-handler`

    This is the log handler for the oracle.webservices.management.auditing logger.

  – `oracle-webservices-management-logging-handler`

    This is the log handler for the oracle.webservices.management.logging logger.

  The name of the Handler is used only within a `<logger>` element (described in the following text) to assign the Handler to a Logger.

  The Handler class can be either a subclass of `java.util.logging.Handler` or a class that implements a `HandlerFactory` interface. If the class is a `java.util.logging.Handler` subclass, the default constructor for that class will be used to create a Handler instance.

  If the class implements the `HandlerFactory` interface, additional configuration properties for the Handler can be specified. The only available `HandlerFactory` class is `oracle.core.ojdl.logging.ODLHandlerFactory`, which can be used to configure an `ODLHandler` instance.

  The `ODLHandlerFactory` class accepts the following properties, each specified in a `<property>` subelement:

  – `path`: Specifies the directory in which the Handler will generate log files. In the case of `ODLHander`, the directory specified is the destination for all ODL-formatted logs. Do not modify this value.

  – `maxFileSize`: Sets the maximum size, in bytes, that any log file in the directory will be allowed to grow to. When a file exceeds this limit, a new file is generated.

  – `maxLogSize`: Sets the maximum size, in bytes, allowed for the log file directory. When this limit is exceeded, log files are purged, beginning with the oldest files.

- <loggers>

  This element defines the mapping between each named Logger and the specific Handler that will process its messages, including ODLHandler. Each mapping is defined within a <logger> element, which includes the following:

  – name: The Logger name.

  – level: The minimum log level that this Logger acts upon. This level can be either a Java log level (FINE) or an ODL Message Type:Log Level (TRACE:1).

  – useParentHandlers: Indicates whether or not the Logger should use its parent Handlers. This value is true by default.

  – <handler>: The name of a Handler to use, as defined in a <log_handler> element. Note that only Handlers defined within a <log_handler> element can be specified.

The following example shows the definition of the ODLHandler and the mapping of the default oracle and custom acme.scheduler Loggers to the ODLHandler within j2ee-logging.xml.

```
<logging_configuration>
  <log_handlers>
    <log_handler name='oc4j-handler'
        class='oracle.core.ojdl.logging.ODLHandlerFactory'>
      <property name='path' value='%ORACLE_HOME%/j2ee/log/oc4j'/>
      <property name='maxFileSize' value='10485760'/>
      <property name='maxLogSize' value='104857600'/>
    </log_handler>
  </log_handlers>
  <loggers>
    <logger name='oracle' level='NOTIFICATION:1' useParentHandlers='false'>
      <handler name='oc4j-handler'/>
    </logger>
    <logger name='acme.scheduler' level='TRACE:1' useParentHandlers='false'>
      <handler name='oc4j-handler'/>
    </logger>
  </loggers>
</logging_configuration>
```

5

Creating MBeans to Manage Your Applications

This chapter provides guidelines on creating and deploying MBeans that will be used to manage applications that will be deployed into OC4J. It includes the following sections:

Overview of MBeans

An *MBean*, or *managed bean*, is a Java object that represents a manageable resource in a distributed environment, such as an application, a service, a component or a device. You can create MBeans for deployment with an application into OC4J, enabling the application or its components to be managed and monitored through the Web-based Oracle Enterprise Manager 10*g* Application Server Control Console user interface.

J2EE-related MBeans are defined in the *J2EE Management Specification (JSR-77)*, which is part of the J2EE 1.4 specification as published by Sun Microsystems. This JSR defines a set of managed objects and associated functionality that must be supported by J2EE-compliant containers. OC4J is fully compliant with JSR-77.

An MBean has a management interface that is exposed to enable a management client to manage a resource. The interface is composed of attributes, operations, and notifications:

- *Attributes*, name and value pairs of any type that the management client can get or set. Attributes are analogous to properties set on a JavaBean.
- *Operations*, methods with any signature and any return type that the client can invoke.
- *Notifications* that can be generated when specific events occur.

The actual management functionality is provided by the *OC4J MBeanServer*, which runs as a service within OC4J. The MBeanServer is able to discover, instantiate, and access MBeans supplied with your application, exposing the MBeans to other OC4J components and enabling administrators to manage the application. Methods called on the MBeanServer access MBean attributes and operations and control MBean instances.

All MBeans deployed with an application must be registered with the OC4J MBeanServer. See "Registering Your MBeans with the OC4J MBeanServer" on page 5-6 for details.

For more on the J2EE Management Specification (JSR-77), visit the Java Community Process site at the following link:

`http://jcp.org/en/jsr/detail?id=77`

Types of MBeans Supported by OC4J

Any of the following MBean types defined in JMX can be deployed into OC4J:

- Standard MBeans

  These are the simplest MBeans to design and implement; however, they are viable only in a static management interface.

  The attributes and operations of a Standard MBean are derived from a Java interface which includes the suffix `MBean` in its name. A Java object can be a Standard MBean simply by being of a class that has the same name as the interface, but without the `MBean` suffix. For example, an object would be of the class `Manager`, in the same Java package as the interface `ManagerMBean`.

  A Standard MBean can also be created from the `javax.management.StandardMBean` class.

- Dynamic MBeans

  These are MBeans that expose a dynamic management interface that is implemented at runtime. Metadata describing each exposed attribute and operation must be made available to the calling application, essentially providing a self-documenting interface.

  Dynamic MBeans must implement the `javax.management.DynamicMBean` interface.

- Model MBeans

  These are Dynamic MBeans that can be configured at runtime. The runtime administration of OC4J is implemented using MBeans of this type.

  A Model MBean implementation can be reused many times with different management interfaces and managed resources, and can provide common functionality such as persistence and caching.

  Model MBeans are defined by the interface `javax.management.modelmbean.ModelMBean`. A Model MBean must be implemented as an object of the `javax.management.modelmbean.RequiredModelMBean` class.

- Open MBeans

  Another type of Dynamic MBean that can be discovered and used by a client at runtime, without requiring the deployment of additional JAR files. Open MBeans are usable with remote management programs that may not have access to application-specific types, including non-Java programs.

  Open MBeans are defined by the package `javax.management.openmbean`.

MBean implementation classes that are registered at deployment time or during application startup— such as MBeans defined in `orion-application.xml`—must include a no-arguments constructor. (See "Registering Your MBeans with the OC4J

MBeanServer" on page 5-6 for guideline.) If the application creates and registers its MBeans, no such requirement exists.

The `oracle.j2ee.admin.jmx` package provides JMX state management capabilities, including localization support, that you may want to consider implementing in your MBean classes. See "Adding Localization Support to Your MBeans" on page 5-21 for details.

Unsupported Methods in JMX MBeanServer and MBeanServerConnection Interfaces

A number of methods from the JMX MBeanServer interface are not available to a J2EE application when it uses an `MBeanServer` object obtained from the following operation:

```
MBeanServer mbsrv = MBeanServerFactory.newMBeanServer();
```

The use of any of the following methods on the returned `MBeanServer` object will throw an `UnsupportedOperationException` exception:

```
public final ClassLoader getClassLoaderFor(ObjectName mbeanName)

public final ClassLoader getClassLoader(ObjectName loaderName)

public final ClassLoaderRepository getClassLoaderRepository()

public final Object instantiate(String className)

public final Object instantiate(String className, ObjectName loaderName)

public final Object instantiate(String className, Object[] params, String[]
signature)

public final Object instantiate(String className, ObjectName loaderName, Object[]
params, String[] signature)

public final ObjectInstance createMBean(String className, ObjectName name)

public final ObjectInstance createMBean(String className, ObjectName name,
ObjectName loaderName)

public final ObjectInstance createMBean(String className, ObjectName name,
Object[] params, String[] signature)

public final ObjectInstance createMBean(String className, ObjectName name,
ObjectName loader, Object[] params, String[] signature)

public final ObjectInputStream deserialize(ObjectName name, byte[] data)

public final ObjectInputStream deserialize(String className, byte[] data)

public final ObjectInputStream deserialize(String className, ObjectName
loaderName, byte[] data)
```

A number of methods from the MBeanServerConnection interface are not supported when an application uses the Oracle JMX connectors. The use of any of the following methods on the `MBeanServerConnection` object that is created will throw an `UnsupportedOperationException` exception:

```
public final ObjectInstance createMBean(String className, ObjectName name)
```

```
public final ObjectInstance createMBean(String className, ObjectName name,
ObjectName loaderName)

public final ObjectInstance createMBean(String className, ObjectName name,
Object[] params, String[] signature)

public final ObjectInstance createMBean(String className, ObjectName name,
ObjectName loader, Object[] params, String[] signature)
```

If your application uses the JMX MBeanServer or MBeanServerConnection interface, avoid using any of the unsupported methods in the application.

Packaging Your MBeans for Deployment

MBeans are packaged with the application they will manage. Package MBean classes in a JAR file and add the JAR file to the root level of the application's EAR file structure.

This section includes the following topics:

- Defining MBeans in orion-application.xml
- Initializing MBean Attributes

Defining MBeans in orion-application.xml

You can provide the configuration data needed to register your MBeans upon deployment by defining them in `orion-application.xml`, the OC4J-specific extension to the J2EE standard `application.xml` descriptor. Both of these descriptors are packaged with the MBeans in the parent application's EAR file.

MBeans defined in `orion-application.xml` will be registered automatically with the OC4J MBeanServer upon deployment or application start. If the application is undeployed, any MBeans belonging to it will also be undeployed.

Add the following XML elements to this descriptor to register the MBeans included in the EAR:

- A `<library>` or `<library-directory>` element pointing to the JAR file containing the MBean classes. Set the `path` attribute to the JAR file name, as follows:

  ```
  <library path="MyMBeans.jar"/>
  ```

- A unique `<jmx-mbean>` element for each MBean class included with the application. This element has a <description> subelement that you can use to specify a name for display in the MBean browser user interface. Each element registers an MBean class with the MBeanServer.

  The `<jmx-mbean>` element has the following attributes:

  - `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`.

    Ideally, the MBean name should include a `type` property indicating the logical MBean type, such as Servlet, Application, DisplayController, and so on.

- `class`: The MBean implementation class.

The `<jmx-mbean>` element optionally takes the following sub-elements:

– A `<description>` sub-element containing a readable name. This name will be displayed in the MBean browser user interface.

– One or more `<attribute>` elements, each defining an initial value to set for an attribute of the MBean. See "Initializing MBean Attributes" on page 5-5 for details.

The following example defines two application-specific MBeans in the `orion-application.xml` deployment descriptor packaged in the parent application's EAR file:

```
<orion-application>
  ...
  <jmx-mbean objectname=":type=Application,name=MyMBeanA"
    class="my.mbeans:MBeanTypeA">
    <description>My First MBean</description>
  </jmx-mbean>
  <jmx-mbean objectname=":type=Application,name=MyMBeanB"
    class="my.mbeans:MBeanTypeB">
    <description>My Second MBean</description>
  </jmx-mbean>
</orion-application>
```

Initializing MBean Attributes

You can pre-configure an MBean by setting initial values for one or more of its attributes in the `orion-application.xml` descriptor packaged with the MBean. The MBean attributes will be initialized with these values upon instantiation.

Attribute values must be one of the following types to be preconfigured:

- Primitive types (such as int, long, Integer, and Boolean)
- String constructors (In the current release, this value must be a `javax.management.ObjectName` value representing the object name of an MBean.)
- One-dimensional arrays of these supported types

Each attribute and its value is specified in an `<attribute>` element within the `<jmx-mbean>` element defining the MBean in `orion-application.xml`.

The actual value is specified in a `<value>` subelement. Multiple `<value>` subelements containing string values to set for the same attribute can be wrapped within a `<values>` element.

The following example illustrates how the supported value types can be set within a `<jmx-mbean>` element:

```
<orion-application>
  ...
  <jmx-mbean objectname=":type=Application,name=MyMBeanA"
     class="my.mbeans:MBeanTypeA">
    <description>My First MBean</description>
    <attribute name="attr1">
      <value>true</value>
    </attribute>
    <attribute name="attr2">
      <value>100</value>
    </attribute>
```

```
    <!-- An array of strings -->
    <attribute name="attr3">
      <value>Test string 1</value>
      <value>Test string 2</value>
      <value>Test string 3</value>
    </attribute>
    <!-- A javax.management.ObjectName representing the name of an MBean -->
    <attribute name="attr3">
     <value>MyApp:Type=Administration</value>
    </attribute>
  </jmx-mbean>
</orion-application>
```

Registering Your MBeans with the OC4J MBeanServer

The MBeans deployed with your application must be registered with the MBeanServer. Once registered, an MBean is fully manageable through the System MBean Browser component of the Application Server Control Console.

You have three options for registering MBeans:

- Define the MBeans in an `orion-application.xml` descriptor that is packaged with the MBeans in the application EAR file.
- Define the MBeans in the application's deployment plan at deployment time.
- Programmatically register the MBeans from within the code of an application.

  See "Programmatically Registering MBeans Through Application Code" on page 5-7 for implementation guidelines.

The following topics describe these options:

- Defining MBeans in an Application Descriptor
- Defining MBeans in a Deployment Plan
- Programmatically Registering MBeans Through Application Code

Defining MBeans in an Application Descriptor

The most straightforward mechanism for automatically registering MBeans at the time of deployment is to define the MBeans in an `orion-application.xml` descriptor that is packaged with the MBeans in the application EAR file. See "Defining MBeans in orion-application.xml" on page 5-4 for details.

Note: An MBean can be registered under the global `default` application, which makes it visible to all other applications deployed into the OC4J instance.

MBeans are registered with the `default` application by adding a `<jmx-mbean>` element to the OC4J-specific `application.xml` file, located in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/config` directory by default.

The formats of `application.xml` and `orion-application.xml` files are defined by the same XML schema. See "Defining MBeans in orion-application.xml" on page 5-4 for details on the `<jmx-mbean>` element.

Defining MBeans in a Deployment Plan

You can define MBean in an application's *deployment plan*, which consolidates all of the OC4J-specific configuration data that is spread among multiple deployment descriptors, including `orion-application.xml`. Data set in the deployment plan is persisted to the `orion-application.xml` descriptor created for the application within OC4J.

Deployment plans can be created or edited at deployment time using the deployment plan editor functionality provided in the Application Server Control Console and Oracle JDeveloper 10*g*. See the *Oracle Containers for J2EE Deployment Guide* for detailed guidelines on creating and working with deployment plans.

Programmatically Registering MBeans Through Application Code

MBeans can optionally be registered dynamically from within the code of an application.

When MBeans are registered programmatically, they are bound to the containing application's lifecycle. This means that if the application is undeployed, all of its MBeans are automatically unregistered from the MBeanServer.

An application that will register MBeans must import the`javax.management` package, which provides the core JMX classes, including classes needed to access the OC4J MBeanServer. The MBean implementation class must also be available to the application.

Applications gain access to the OC4J MBeanServer by creating a reference through the `javax.management.MBeanServerFactory`, as shown in the code snippet below:

```
MBeanServer mbsvr = MBeanServerFactory.newMBeanServer();
```

Note the following restrictions:

- MBeans must be registered under an `ObjectName` whose domain is the namespace under which the application was deployed. This will ensure that beans live in their own namespace within OC4J. The `getDefaultDomain()` method can be called on the `MBeanServer` object to return the correct domain for a given application.
- An application can only set attributes and call methods on MBeans that belong to it. In fact, MBeans that belong to other applications are not visible to the application.

The following code snippet registers an MBean with the OC4J MBeanServer. The MBean is an object of the `oracle.oc4j.admin.jmx.server.mbeans.Tester` class.

```
try
  {
    // Get a reference to the MBeanServer
    MBeanServer _mbeanServer = MBeanServerFactory.newMBeanServer();

      // Create the MBean instance
      Tester bean = new Tester();

      //Construct the MBean name using the default application's domain name
      ObjectName beanName= new ObjectName(mbsrv.getDefaultDomain()+
                                    ":type=Tester,name=MyMBean");

    // Register the MBean with the MBeanServer
    mbsrv.registerMBean(bean, beanName);
```

```
  }
  catch(Exception e)
  {
    // Handle exceptions; for simplicity, dump the stack trace to show any
    // errors that occur
    e.printStackTrace();
  }
```

The next example is a sample servlet - `UserMBeanServlet` - that will register an MBean with the OC4J MBeanServer in its `init()` method.

```
package web;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

import javax.management.*;

import oracle.oc4j.admin.jmx.server.mbeans.Tester;

public class UserMBeanServlet extends HttpServlet {

public void init() throws ServletException {
  try {
      // Get a reference to the MBeanServer
      MBeanServer mbsrv = MBeanServerFactory.newMBeanServer();

      // Create the MBean instance
      Tester bean = new Tester();

      //Construct the MBean name using the application's domain name
      ObjectName beanName= new ObjectName(mbsrv.getDefaultDomain()+
                          ":type=Tester,userprop1=bean,userprop2=beanProp2");

    // Register the MBean with the MBeanServer
    mbsrv.registerMBean(bean, beanName);

    // Print a success message to the console
    System.out.println("Finished registering" +beanName);

    }
catch(Exception e) {
// Dump the stack trace to show any errors that occur
e.printStackTrace();
    }
  }
//Standard servlet code for handling requests
...
}
```

Providing Management Access to Application-Defined MBeans

If you provide MBeans with your application, you will likely want to enable users to access and manage the MBeans remotely.

Application MBeans can be managed remotely by accessing the remote OC4J MBeanServer through JSR-160 compliant code. JSR-160 a standard API to connect to remote JMX-enabled applications using RMI. This is also known as *JMX remoting*.

You have two options for remotely managing MBeans:

- Remote Management Using the Management EJB (JSR-77)
- Remote Management Using the JMX Remote API (JSR-160)

Prerequisite: Add User to Security Group

To access an application's MBeans, the user must be added to the `oc4j-app-administrators` security groups. Users can be added to groups using either the JAZN Admintool or the Web-based Oracle Enterprise Manager 10*g* Application Server Control Console.

Note that the `oc4j-app-administrators` security group must have permission to login and invoke methods on the remote OC4J process. The group must also have namespace read access on the server.

See the *Oracle Containers for J2EE Security Guide* for detailed instructions on adding users to security groups.

Remote Management Using the JMX Remote API (JSR-160)

The *JMX Remote API (JSR-160)* provides another option for managing MBeans remotely. In fact, it offers a number of advantages over using the MEJB defined by JSR-77, making it the preferred method for remote management:

- More of the MBeanServer functionality is available than is exposed through the MEJB.
- Compliant code can easily be migrated to use new connection protocols as they become available.
- The OC4J `JMXConnector` implementation supports localization and HTTP tunneling, which enables clients to communicate with an MBeanServer across firewalls.

  Note that ORMI over SSL, ORMIS, is also supported. See the *Oracle Containers for J2EE Security Guide* for detailed instructions on adding users to security groups.
- The `JMXConnector` also allows the connection state to be monitored.
- You can use the OC4J `admin_client.jar` tool through the Oracle JMX Remote API to manage an OC4J instance in an Oracle Application Server environment or a standalone OC4J server.

  For information about `admin_client.jar`, what package you can use, and the client-side libraries you need to specify, see the *Oracle Containers for J2EE Configuration and Administration Guide*.

Note that because OC4J uses ORMI, and not JRMP, the Oracle JMX Remote API implementation is not compatible with other JSR-160 implementations.

The JMX Remote API defines a standard connector that provides Java clients with remote access to an MBeanServer via the RMI protocol. In the OC4J implementation, the connector is attached to the OC4J MBeanServer. Actual management is through a proxy; for each method called on the proxy instance, a corresponding method is to be called on the remote MBeanServer.

The following discussions explain how the API can be used:

- Connecting to the OC4J MBeanServer
- Connecting to an Application-Specific MBean Server

- Connecting to a Specific Application's JMX Domain
- Setting the JMX Service URI for an OPMN-Managed OC4J Instance
- Setting the JMX Service URI for a Standalone OC4J Instance
- Setting a Locale
- Enabling HTTP Tunneling

Connecting to the OC4J MBeanServer

The following sample code creates a `JMXConnector` instance and uses it to connect to a target OC4J instance defined as a `JMXServiceURL` object. An `MBeanServerConnection` instance, which serves as a proxy for the OC4J MBeanServer, is retrieved. The proxy allows management operations to be performed on the MBeanServer operations; in this case, retrieving all the MBeans registered with the MBeanServer.

Note: To connect in the manner outlined in this section, the user must be assigned to the `oc4j-administrators` role, which grants the user full access to the MBeanServer and all of the MBeans registered with it, including OC4J system and application-defined MBeans.

The default OC4J administration user, `oc4jadmin`, is a member of this role.

See "Connecting to a Specific Application's JMX Domain" on page 5-14 for guidelines on enabling a user to access a specific application's MBeans without assigning the user to the `oc4j-administrators` role.

The example code imports the following JMX classes and interfaces:

- `javax.management.remote.JMXConnector` interface

  Defines the client end of the JMX connector.
- `javax.management.remote.JMXConnectorFactory` class

  A factory containing methods to create JMX connector clients.
- `javax.management.remote.JMXServiceURL` class

  Constructs a URL defining the connection target. The constructor takes one or more `String` objects as parameters. Here a `String` variable defining the OPMN lookup URL containing the data needed to access an OPMN-managed OC4J instance is passed to the constructor:

  ```
  String url="service:jmx:rmi:///opmn://oc4jhost1:6003/home"
  ...
  JMXServiceURL serviceUrl= new JMXServiceURL(url);
  ```

  See "Setting the JMX Service URI for an OPMN-Managed OC4J Instance" on page 5-14 for instructions on connecting to an OPMN-managed OC4J instance running as a component of Oracle Application Server.

  See "Setting the JMX Service URI for a Standalone OC4J Instance" on page 5-16 for instructions on connecting in a standalone OC4J environment.

- `javax.management.MBeanServerConnection` interface

  Defines the proxy for performing operations on the OC4J MBeanServer.

In addition to these JMX classes and interfaces, the `oracle.oc4j.admin.jmx.remote.api.JMXConnectorConstant` class is imported to make constants used by OC4J JMX connector clients available. Rather than importing this class, you may want to use the keys defined in this class directly in your code to avoid introducing Oracle APIs into your code. The constants defined in the class are:

- `CREDENTIALS_LOGIN_KEY`

  Stores a server login name.

- `CREDENTIALS_PASSWORD_KEY`

  Stores a server login password.

- `LOCALE`

  Stores a `Locale` used to localize MBean metadata, attributes and methods accessed via the connection. See "Setting a Locale" on page 5-16 for details on localizing a connection instance.

- `HTTP_TUNNELING`

  Stores a Boolean value indicating whether HTTP tunneling is enabled. See "Enabling HTTP Tunneling" on page 5-17 for details on enabling HTTP tunneling.

```
// Import the JSR-160 classes and interfaces from jmx_remote_api.jar
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXServiceURL;
import javax.management.remote.JMXConnectorFactory;

// Import the JMX 1.2 class
import javax.management.MBeanServerConnection;

// Import OC4J specific constant values. You can optionally use
// the values specified in this class to avoid introducing
// any Oracle-specific code.
import oracle.oc4j.admin.jmx.remote.api.JMXConnectorConstant;

....
// Create a variable for a URL containing data needed to access
// the connection target; in this case, an OPMN-managed OC4J instance
String url="service:jmx:rmi///opmn://opmnhost1.company.com:6003/home"

JMXConnector jmxCon= null;

try {
// Define the connection target
JMXServiceURL serviceUrl= new JMXServiceURL(url);

// Use to pass environment properties to be used while
// retrieving a connection
Hashtable env= new Hashtable();

// Define the provider root package
env.put(JMXConnectorFactory.PROTOCOL_PROVIDER_PACKAGES,
 "oracle.oc4j.admin.jmx.remote");

Hashtable credentials= new Hashtable();
```

```
// Connect using the oc4jadmin super-user administrator account
credentials.put(JMXConnectorConstant.CREDENTIALS_LOGIN_KEY,"oc4jadmin");
credentials.put(JMXConnectorConstant.CREDENTIALS_PASSWORD_KEY,"password");

// Specify the login/password to use for the connection
env.put(JMXConnector.CREDENTIALS, credentials);

// Get an instance of the JMXConnector interface for OC4J's rmi protocol
// User is not yet connected
jmxCon = JMXConnectorFactory.newJMXConnector(serviceUrl, env);

// Connect to the target OC4J instance defined in the JMXServiceURL
jmxCon.connect();

// Retrieve the MBeanServerConnection instance that acts as a proxy
// for the OC4J MBeanServer we are connecting to.
MBeanServerConnection con= jmxCon.getMBeanServerConnection();

// Use the MBeanServerConnection instance to perform remote
// operations on the OC4J MBeanServer. This call retrieves
// all MBeans registered with the server.
Set mbeans= con.queryNames(null, null);

// Display each MBean's ObjectName
Iterator iter= mbeans.iterator();
while(iter.hasNext())
System.out.println(iter.next().toString());
}

// Important!!! Release the connection, ideally using a Finally block
finally {
if(jmxCon!=null)
 jmxCon.close();
}
```

Connecting to an Application-Specific MBean Server

Applications can create and connect to a generic MBeanServer instance, instead of using the OC4J MBeanServer. This is useful, for example, when creating applications that must be portable to multiple J2EE containers, not just OC4J.

In this scenario, you will supply the default domain for the MBeanServer. The MBeanServer instance will be created and registered with the MBeanServerFactory.

The example below creates and registers an MBeanServer instance under the domain "myserv". The code specific to this MBeanServer instance is highlighted as bold.

```
// Import the JSR-160 classes and interfaces from jmx_remote_api.jar
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXServiceURL;
import javax.management.remote.JMXConnectorFactory;

// Import the JMX 1.2 class
import javax.management.MBeanServerConnection;

// Import OC4J specific constant values. You can optionally use
// use the values specified in this class to avoid introducing
// any Oracle-specific code.
import oracle.oc4j.admin.jmx.remote.api.JMXConnectorConstant;
```

```
....
// Create a variable for a URL containing data needed to access
// the connection target; in this case, an OPMN-managed OC4J instance
String url="service:jmx:rmi///opmn://opmnhost1.company.com:6003/home"

JMXConnector jmxCon= null;

try {
// Define the connection target
JMXServiceURL serviceUrl= new JMXServiceURL(url);

// Use to pass environment properties to be used while
// retrieving a connection
Hashtable env= new Hashtable();

// Define the provider root package
env.put(JMXConnectorFactory.PROTOCOL_PROVIDER_PACKAGES,
 "oracle.oc4j.admin.jmx.remote");

Hashtable credentials= new Hashtable();

// Connect using the oc4jadmin administrator account
credentials.put(JMXConnectorConstant.CREDENTIALS_LOGIN_KEY,"oc4jadmin");
credentials.put(JMXConnectorConstant.CREDENTIALS_PASSWORD_KEY,"password");

// Specify the login/password to use for the connection
env.put(JMXConnector.CREDENTIALS, credentials);

// Specify the application-specific MBeanServer default domain name
// used at creation time for the MBeanServer the application will connect to.
// The domain name specified here is "myserv".
env.put(JMXConnectorConstant.PROPRIETARY_MBEANSERVER_DOMAIN_NAME, "myserv");

// Get an instance of the JMXConnector interface for OC4J RMI protocol
// User is not yet connected
jmxCon = JMXConnectorFactory.newJMXConnector(serviceUrl, env);

// Connect to the target OC4J instance defined in the JMXServiceURL
jmxCon.connect();

// Retrieve the MBeanServerConnection instance that acts as a proxy
// for the OC4J MBeanServer we are connecting to.
MBeanServerConnection con= jmxCon.getMBeanServerConnection();

// Use the MBeanServerConnection instance to perform remote
// operations on the OC4J MBeanServer. This call retrieves
// all MBeans registered with the server.
Set mbeans= con.queryNames(null, null);

// Display each MBean's ObjectName
Iterator iter= mbeans.iterator();
while(iter.hasNext())
System.out.println(iter.next().toString());
}

// Important!!! Release the connection, ideally using a Finally block
finally {
if(jmxCon!=null)
 jmxCon.close();
}
```

Connecting to a Specific Application's JMX Domain

Because users assigned to the `oc4j-administrators` role can access all MBeans registered with the MBeanServer, assigning this role to all users may not be desirable.

You can, however, enable a user that is not assigned this role to access only those MBeans registered by your application. In this case, the user connects at the application level, and will only see MBeans registered by the application.

The code is the same as that outlined in "Connecting to the OC4J MBeanServer" above, except the `url` passed to the `JMXServiceURL` constructor includes the name of the application. The following example will provide access to MBeans registered by the `hello-world` application:

```
// Create a variable for a URL containing data needed to access
// the connection target; in this case, an OPMN-managed OC4J instance
String url="service:jmx:rmi///opmn://opmnhost1.company.com:6003/home/hello-world"

JMXConnector jmxCon= null;

try {
// Define the connection target
JMXServiceURL serviceUrl= new JMXServiceURL(url);
...
```

See "Setting the JMX Service URI for an OPMN-Managed OC4J Instance" on page 5-14 for instructions on connecting to an OPMN-managed OC4J instance running as a component of Oracle Application Server.

See "Setting the JMX Service URI for a Standalone OC4J Instance" on page 5-16 for instructions on connecting in a standalone OC4J environment.

Setting the JMX Service URI for an OPMN-Managed OC4J Instance

In an Oracle Application Server environment, the RMI port for an OC4J instance, required for a JMX connection, is not fixed and can change each time the instance is started by OPMN.

To resolve this issue, an OPMN-based URL must be passed to the `JMXServiceURL` constructor. This URL provides an indirect lookup with OPMN and returns the RMI port required for the connection.

The syntax of the OPMN lookup URI used to connect to the MBeanServer on a specific OPMN-managed OC4J instance is as follows:

```
service:jmx:rmi|ormi:///opmn://opmnHost:[opmnPort]/oc4jInstanceName/[appName]
```

To connect to the cluster MBeanServer, specify the following URI. Note the inclusion of `/cluster`, indicating that the connection is with the cluster MBeanServer, rather than a specific OC4J instance's MBeanServer:

```
service:jmx:rmi:///opmn://opmnHost:[opmnPort]/cluster/[ASInstanceName
/[Oc4jCompName]
```

Note that while the `ASInstanceName` and `Oc4jCompName` parameters are optional, it is highly recommended that you do specify an Oracle Application Server and/or OC4J instance to connect to. Otherwise, a connection is made with a randomly-selected OC4J process within the cluster. This can result in the creation of a new instance of the cluster MBeanServer each time a new connection is obtained.

The recommended practice is to specify both `ASInstanceName` and `Oc4jCompName`, as a connection will be obtained with a single instance of the cluster MBeanServer.

Because OPMN will automatically restart the OC4J process if needed, the process is guaranteed to be always available.

For example, the following example connects to clusterMBeanServer within the `admin` OC4J instance on the `as101` Oracle Application Server instance:

```
service:jmx:rmis:///opmn://stadp69:6003/cluster/as101/admin
```

Table 5–1 describes the cluster MBeanServer service URI parameters and their values.

Table 5–1 URI parameters

Parameter	Value
`hostname`	The name of the OPMN host, such as `oc4jhost1`. This value is required.
`port`	The OPMN request port. This value is specified in the `request` attribute of the `<port>` element in `opmn.xml`. If not specified, the default value `6003` is used.
`oc4jInstanceName`	Valid for a specific OC4J MBeanServer only. The name of the OC4J instance. This value is required. The name of the default OC4J instance created in Oracle Application Server is `home`. Specify `/home` to connect to this instance on the target host.
`appName`	Valid for a specific OC4J MBeanServer only. The optional name of a specific application to access, such as `/petstore`. If not specified, an unrestricted connection to all applications is returned. This option is used to allow users not assigned the `oc4j-administrators` role to access a specific application's MBeans. Because the connection is made at the application level, the user has access only to those MBeans registered by the application.
`ASInstanceName`	Valid for a cluster MBeanServer only. The optional name of an Oracle Application Server instance to connect to. This value is specified in the `id` attribute of the `<ias-instance>` element in `opmn.xml`.
`Oc4jCompName`	Valid for a cluster MBeanServer only. The name of an OC4J instance to connect to. This value is specified in the `id` attribute of the `<process-type>` element in `opmn.xml`. If this parameter is specified without ASInstanceName, the port parameter must be supplied for the Oracle Application Server instance the OC4J instance is running within.

The following URL provides unrestricted JMX access to all application MBeans deployed into the `home` OC4J instance on the specified host. Note that the port value is omitted, meaning the default will be used:

```
service:jmx:rmi///opmn://opmnhost1/home
```

The next example provides access only to MBeans registered by the `petstore` application. The connection target is the `home02` instance on the specified host. Note that the port value has been specified:

```
service:jmx:rmi///opmn://opmnhost1:6008/home02/petstore
```

Setting a Secure JMX Service URI for an OPMN-Managed OC4J Instance

The following URI accesses the cluster MBeanServer:

```
service:jmx:rmis:///opmn://opmnHost:opmnPort/cluster/[ASInstanceName]/Oc4jCompName
```

Setting the JMX Service URI for a Standalone OC4J Instance

In a standalone OC4J installation, in which OC4J is installed, managed, started and stopped directly as a self-contained component, the RMI port is fixed. A URL containing connection parameters can be passed to the `JMXServiceURL` constructor to connect directly to the OC4J server.

The syntax of the lookup URL used in a standalone OC4J installation is as follows:

```
service:jmx:rmi|ormi://[hostname]:[rmiPort/]oc4jContextRoot/[appName]
```

For example:

```
service:jmx:rmi://oc4jhost:23791/oc4j/petstore
```

Table 5–2 describes the URL parameters and their values.

Table 5–2 URL parameters

Parameter	Value
`hostname`	Optional. The name of the OC4J host, such as `oc4jhost1`. Defaults to `localhost` if not specified.
`rmiPort`	Optional. The RMI port to connect to. If not specified, the value defaults to `23791`.
`oc4jcontextRoot`	Required. The URL path to the OC4J installed directory on the server (`"/oc4j"`). The `/oc4j` context URI is used to identify the OC4J instance's local MBeanServer instance. This value is required.
`appName`	Optional. The name of a specific application to access, such as `/petstore`. If not specified, an unrestricted connection to all applications is returned. This option is used to enable users not assigned the `oc4j-administrators` role to access a specific application's MBeans. Because the connection is made at the application level, the user has access only to those MBeans registered by the application.

Setting a Secure JMX Service URI for a Standalone OC4J Instance

You can use ORMI over SSL, or ORMIS, to secure the connection between the management client and the OC4J MBeanServer. To use this feature, simply replace `rmi` or `ormi` with `rmis` or `ormis` in the JMX service URI syntax illustrated in the preceding text. In the following example, the ORMIS port `23943` is specified:

```
service:jmx:rmis://oc4jhost:23943/oc4j/petstore
```

The target OC4J server must be configured to use ORMIS. See the *Oracle Containers for J2EE Security Guide* for instructions on enabling ORMIS.

Setting a Locale

A specific `Locale` can be associated with a connection by setting an additional environment property passed to either the `JMXConnectorFactory.newJMXConnector()` or `JMXConnector.connect()` method. The `LOCALE` constant of the OC4J-specific `oracle.oc4j.admin.jmx.api.JMXConnectorConstant` class can be used to set this property. For example:

```
env.put(JMXConnectorConstant.LOCALE, Locale.FRENCH)
```

Note that for localization to be used, the MBeans to be managed must support localization as outlined in "Adding Localization Support to Your MBeans" on page 5-21.

Enabling HTTP Tunneling

In scenarios where the client, the OC4J server or both are secured behind firewalls that allow only HTTP traffic, the JMX RMI connector can be configured to tunnel RMI traffic over HTTP, enabling communication across firewalls.

In HTTP tunneling, RMI calls are encapsulated within an HTTP `POST` request. Replies are similarly returned as HTTP-encapsulated data.

HTTP tunneling is enabled by setting the value of the `HTTP_TUNNELING` constant of the `oracle.oc4j.admin.jmx.api.JMXConnectorConstant` class to the path of the `rmiTunnel` servlet and passing it as an environment property to either the `JMXConnectorFactory.newJMXConnector()` or `JMXConnector.connect()` method. For example:

```
env.put(JMXConnectorConstant.HTTP_TUNNELING, "j2ee/rmiTunnel")
```

Note that the port value in the `JMXServiceURL` object used to get a `JMXConnection` instance must be set to the HTTP port of the target OC4J instance, and not the RMI port, as shown in previous connection examples. The OC4J default HTTP listener port is `8888` in OC4J standalone, or `7777` in an Oracle Application Server environment.

```
JMXServiceURL serviceUrl= new JMXServiceURL("rmi","oc4j-sun.acme.com",
8888,"/oc4j");
```

See the *Oracle Containers for J2EE Services Guide* for instructions on configuring RMI HTTP tunneling in OC4J.

Remote Management Using the Management EJB (JSR-77)

In compliance with the *J2EE Management Specification (JSR-77)*, OC4J enables users to remotely manage MBeans through the *Management EJB (MEJB)*, which is deployed with the OC4J implementation. The MEJB is a stateful session bean that provides a remote interface to the OC4J MBeanServer, allowing remote users to query and access MBeans running in an OC4J instance.

The MEJB uses JMX classes and interfaces. The `javax.management.j2ee.Management` interface is the MEJB remote interface, while the `javax.management.j2ee.ManagementHome` interface contains a single method which creates an MEJB instance. The MEJB is available under the JNDI name `ejb/mgmt/MEJB`.

The following discussions explain how the MEJB can be used:

- Accessing the MEJB from a J2EE Application Client
- Accessing the MEJB from a Servlet or EJB

Accessing the MEJB from a J2EE Application Client

The following code enables an application client to use the MEJB. Note that accessing the MEJB from an application client allows both local and remote operations to be performed.

```
import javax.naming.*;

// Import the MEJB interface
import javax.management.j2ee.Management;
import javax.management.j2ee.ManagementHome;

.....

Hashtable env = new Hashtable();

// Set the connection target
String url = "ormi://host.company.com:23791/default";

// Set the login context
env.put(Context.PROVIDER_URL, url);
env.put(Context.SECURITY_PRINCIPAL, "oc4jadmin");
env.put(Context.SECURITY_CREDENTIALS, "welcome");
env.put(Context.INITIAL_CONTEXT_FACTORY,
  "com.evermind.server.ApplicationClientInitialContextFactory");

// Look up the MEJB Home interface and create the MEJB
Context ctx = new InitialContext(environment);
Object hm= ctx.lookup("java:comp/env/ejb/mgmt/MEJB");
ManagementHome mgmtHome=
 (ManagementHome)PortableRemoteObject.narrow(hm,ManagementHome.class);
Management mejb=  mgmtHome.create();

.....

mejb.remove();
```

Accessing the MEJB from a Servlet or EJB

The following code enables a servlet or EJB running within the target OC4J instance to use the MEJB. Because the connection target is the executing container, the login and password data is not supplied, but is instead retrieved from the executing thread context. For this reason, the authenticating user must belong to the `oc4j-administrators` security group.

See the *Oracle Containers for J2EE Security Guide* for details on adding users to groups.

```
import javax.naming.*;

// Import the JSR-77 MEJB interface
import javax.management.j2ee.Management;
import javax.management.j2ee.ManagementHome;

.....

// Look up the MEJB Home interface and create the MEJB
Context ctx= new InitialContext();
Object hm= ctx.lookup("java:comp/env/ejb/mgmt/MEJB");
ManagementHome mgmtHome=
 (ManagementHome)PortableRemoteObject.narrow(hm,ManagementHome.class);
Management mejb=  mgmtHome.create();
```

```
.....

mejb.remove();
```

Sample MBean

The following is an example of a simple MBean implementation. This MBean includes operations to enable or disable a user within the application it is packaged with.

UserManagerMBean Interface

This is the Java interface for the MBean.

```
package demo.servicereq.management;

public interface UserManagerMBean {
  String[] listUsers();
  void enableUser(int userId);
  void disableUser(int userId);
}
```

UserManager Implementation Class

This is the MBean implementation class.

```
package demo.servicereq.management;

import oracle.srdemo.data.User;
import oracle.srdemo.data.UserAccess;
import java.util.Collection;
import java.util.logging.Level;
import javax.management.MBeanNotificationInfo;
import javax.management.Notification;
import javax.management.NotificationBroadcasterSupport;
import java.util.logging.Logger;
import java.util.Iterator;

public class UserManager
       extends NotificationBroadcasterSupport
       implements UserManagerMBean {

 private static final String m_classname =
 UserManager.class.getClass().getName();

 public static final String ENABLE_USER = "enableUser";
 public static final String DISABLE_USER = "disableUser";
 public static final String LIST_USERS = "listUsers";

 String[] m_users;

  // A logger
  private static final Logger m_logger = Logger.getLogger(m_classname);



 /**
   * Constructor.
   */
  public UserManager()  {
  }
```

```
  /**
   * Lists all the user names.
   * @return
   */
  public String[] listUsers() {
    m_logger.entering(m_classname,"listUsers");
    // query all users and return
    m_logger.exiting(m_classname,"listUsers");
    return m_users;
  }

/**
   * Sets the specified user to a state of enabled.
   * @param userId
   */
  public void enableUser(int userId) {
    m_logger.entering(m_classname,"enableUser");
    sendNotification(ENABLE_USER, "userId [" + userId + "] now enabled");
    // Lookup user with userId.
    // Set the status of the user to enabled.
    // throw exception if user doesn't exist.
    m_logger.exiting(m_classname,"enableUser");
  }

/**
   * Sets the specified user to a state of disabled.
   * @param userId
   */
  public void disableUser(int userId) {
    m_logger.entering(m_classname,"disableUser");
    sendNotification(DISABLE_USER, "userId [" + userId + "] now disabled");
    // Lookup user with userId.
    // Set the status of the user to disabled.
    // throw exception if user doesn't exist.
    m_logger.exiting(m_classname,"disableUser");
  }

/**
    * Informs any interested party on what notifications this MBean emits.
    * @return the notifications this MBean emits
    */
  public MBeanNotificationInfo[] getNotificationInfo() {
    m_logger.entering(m_classname,"getNotificationInfo");
    String NOTIFICATIONS[] =
    { ENABLE_USER, DISABLE_USER };
    MBeanNotificationInfo[] info =
    {
      new MBeanNotificationInfo(NOTIFICATIONS,
                           "javax.management.Notification",
                           "Notification set for UserManager") };
    m_logger.exiting(m_classname,"getNotificationInfo");
    return info;
  }

 /**
   * Sends a JMX notification using the sendNotification method
   * from the base class.
   * @param operation - the name of the operation sending the notification.
   * @param desc - the description to place within the notification.
```

```
    * @return void
    */
   public void sendNotification(String operation, String desc) {
     m_logger.entering(m_classname,"sendNotification");
     Notification notification = new Notification(operation,this,
     System.currentTimeMillis(), desc);
     super.sendNotification(notification);
     m_logger.exiting(m_classname,"sendNotification");
   }

}
```

Adding Localization Support to Your MBeans

All Dynamic MBeans - including Model and Open MBeans - must provide metadata describing the MBean as well as each of its exposed attributes and operations. If your application will be marketed internationally, you should design your MBeans to support localization of this metadata.

For additional information on localization with resource bundles, read the following article from Sun Microsystems:

`http://java.sun.com/developer/technicalArticles/Intl/ResourceBundles/`

Localization Support Provided by Oracle

The `oracle.j2ee.admin.jmx` API extends the JMX specification with functionality specific to localizing MBeans. Note that any MBeans that use this API must be specific to OC4J, and will not be portable to other J2EE containers.

The localization-specific interface and class are covered below.

JMXState

The `oracle.j2ee.admin.jmx.JMXState` interface defines the state associated with a JMX operation, specifically the state of the `Locale` instance to be used. This state can be retrieved by an MBean implementation to provide localization support.

This interface contains a single method, which returns the `Locale` instance:

```
public Locale getLocale()
```

JMXStateFactory

The `oracle.j2ee.admin.jmx.JMXStateFactory` class provides access to the state associated with a JMX operation, such as the `Locale` object to be used. It includes the following method:

- `public static JMXState getJMXState()`

  Retrieves the `oracle.j2ee.admin.jmx.JMXState` instance containing the JMX state associated with the calling JMX MBean operation. This method should only be called from within an MBean implementation.

Using Resource Bundles to Localize MBean Metadata

Localization is enabled using *resource bundles*, which are objects of the `java.util.ResourceBundle` class. A resource bundle consists of one or more Java classes or Java `properties` files containing key/value pairs, where the value is a string comprising the descriptive text. A resource bundle will typically exist for each supported language.

The following example is a snippet from the default OC4J `Messages.properties` file showing how the metadata for a sample `UserManager` MBean is stored as key/value pairs. The keys correspond to values set in the code sample that follows.

```
//Description of the MBean
usermanager_description=Manages users of the corresponding installation.

//Attribute description strings
This MBean does not have attributes.


//Operation description strings
usermanager_listUsers=List current user IDs.
usermanager_enableUser=Enables the specified user account.
usermanager_disableUser=Disables the specified user account.

//Operation parameter description strings
usermanager_userId=The user account to affect.
```

See *Localization with Resource Bundles* at the following URL for more on resource bundle implementation:

```
http://java.sun.com/developer/technicalArticles/Intl/ResourceBundles/
```

> **Note:** OC4J uses Java classes to localize the metadata for its own components, including Oracle-supplied MBeans. Do not make changes to the default `Messages.properties` file used by OC4J. Any changes made to this file will be overwritten whenever a new version of OC4J is installed.

Adding Localization Support to Your MBeans

To expose its metadata, an MBean must implement the overloaded `getMBeanInfo()` method, which returns an object of the `javax.management.MBeanInfo` object populated with generic metadata for the attributes and operations exposed by the MBean.

A Dynamic MBean must implement the generic version of this method to retrieve and localize its metadata using the `Locale` object from the JMX client. The implementation of a Model MBean - an object of the `javax.management.modelmbean.RequiredModelMBean` class - must also implement this generic signature. See "Implementing the Generic getMBeanInfo() Signature" on page 5-22 for details.

MBeans supporting localization must also expose the signature `getMBeanInfo(Locale locale)`, which retrieves and localizes the metadata based on the Locale passed in by the OC4J MBeanServer. See "Exposing the Required getMBeanInfo(Locale locale) Signature" on page 5-23

Implementing the Generic getMBeanInfo() Signature

To localize the returned metadata, the generic `getMBeanInfo()` method must get access to the `Locale` object from the JMX client. The `getLocale()` method of the `oracle.j2ee.admin.jmx.JMXState` interface can be used to accomplish this. Your bean should also import the `oracle.oc4j.admin.jmx.JMXStateFactory` class, which provides access to the state associated with a JMX operation, such as the `Locale` object to be used.

The following example illustrates implementation of this method in a Simple MBean.

```
package demo.servicereq.management;


import oracle.srdemo.data.User;
import oracle.srdemo.data.UserAccess;
import java.util.Collection;
import java.util.logging.Level;
import javax.management.MBeanNotificationInfo;
import javax.management.Notification;
import javax.management.NotificationBroadcasterSupport;
import java.util.logging.Logger;
import java.util.Iterator;

// Import packages needed for localization of MBean metadata
import java.util.Locale;
import java.util.ResourceBundle;
import javax.management.MBeanInfo;


// Import the Oracle localization APIs
import oracle.j2ee.admin.jmx.JMXStateFactory;
import oracle.j2ee.admin.jmx.JMXState;

public class UserManager
       extends NotificationBroadcasterSupport
       implements UserManagerMBean {

// Create the MBeanInfo object to allow attributes and operations exposed by the
// MBean to be retrieved
public MBeanInfo getMBeanInfo()
  {
    // Get access to the Locale instance set by the JMX client
    Locale locale = JMXStateFactory.getJMXState().getLocale();

    // Create MBean's localized meta-data using the locale from JMX client
    return createMBeanInfo(locale);
  }
```

Exposing the Required getMBeanInfo(Locale locale) Signature

All MBeans deployed into OC4J, regardless of type, are required to expose the `getMBeanInfo(Locale locale)` signature, which retrieves and localizes the MBean’s metadata based on the `Locale` object passed in as a parameter. This method will be called by clients such as the System MBean Browser component of the Application Server Control Console as well as the Oracle JSR-160 connector, which is used for remote management of MBeans.

The method allows support of localized `MBeanInfo` without requiring any modification to the default `getMBeanInfo()` method, such as introducing the Oracle specific `JMXState` class. As such, it allows you to write localized code that remains portable, although the proper localization will only work within Oracle Application Server.

The following example illustrates implementation of both signatures of `getMBeanInfo()` in a Dynamic MBean.

```
package demo.servicereq.management;

import oracle.srdemo.data.User;
import oracle.srdemo.data.UserAccess;
import java.util.Collection;
import java.util.logging.Level;
import javax.management.MBeanNotificationInfo;
import javax.management.Notification;
import javax.management.NotificationBroadcasterSupport;
import java.util.logging.Logger;
import java.util.Iterator;

// Import the Oracle localization APIs
import oracle.j2ee.admin.jmx.JMXStateFactory;
import oracle.j2ee.admin.jmx.JMXState;

public class UserManager
       extends NotificationBroadcasterSupport
       implements UserManagerMBean {

// Create the MBeanInfo object to allow attributes and operations exposed by the
// MBean to be retrieved
public MBeanInfo getMBeanInfo()
{
  // Get access to the Locale instance set by the JMX client
  Locale locale = JMXStateFactory.getJMXState().getLocale();

  // Create MBean's localized metadata using the Locale set by the JMX client
  return createMBeanInfo(locale);
}

// Expose the Dynamic MBean operation
public MBeanInfo getMBeanInfo(Locale locale)
{
  // Create MBean's metadata using the Locale from the MBeanServer
  return createMBeanInfo(locale);
}
```

In this example, both signatures of `getMBeanInfo()` pass the `Locale` instance to the private `createMBeanInfo(Locale locale)` method, which retrieves the appropriate resource bundle containing the descriptions of the MBean, its attributes and its operations. Note that this method also sets the localized description for each of the operation's parameters. It then creates the `MBeanInfo` object containing the localized metadata.

```
package demo.servicereq.management;

import java.util.Locale;

import java.util.ResourceBundle;

import javax.management.MBeanConstructorInfo;
import javax.management.MBeanInfo;

import javax.management.MBeanOperationInfo;

import javax.management.MBeanParameterInfo;

import oracle.srdemo.data.User;
import oracle.srdemo.data.UserAccess;
```

```
import java.util.Collection;
import java.util.logging.Level;
import javax.management.MBeanNotificationInfo;
import javax.management.Notification;
import javax.management.NotificationBroadcasterSupport;
import java.util.logging.Logger;
import java.util.Iterator;

// Import the Oracle localization APIs
import oracle.j2ee.admin.jmx.JMXStateFactory;
import oracle.j2ee.admin.jmx.JMXState;

public class UserManager
       extends NotificationBroadcasterSupport
       implements UserManagerMBean {

        private static final String m_classname =
 UserManager.class.getClass().getName();

 public static final String ENABLE_USER = "enableUser";
 public static final String DISABLE_USER = "disableUser";
 public static final String LIST_USERS = "listUsers";

  String[] m_users;

  // A logger
  private static final Logger m_logger = Logger.getLogger(m_classname);

 /**
   * Constructor.
   */
     public UserManager()  {
  }

  /**
   * Lists all the user names.
   * @return
   */
  public String[] listUsers() {
    m_logger.entering(m_classname,"listUsers");
    // query all users and return
    m_logger.exiting(m_classname,"listUsers");
    return m_users;
  }

/**
   * Sets the specified user to a state of enabled.
   * @param userId
   */
  public void enableUser(int userId) {
    m_logger.entering(m_classname,"enableUser");
    sendNotification(ENABLE_USER, "userId [" + userId + "] now enabled");

    // Lookup user with userId.
    // Set the status of the user to enabled.
    // throw exception if user doesn't exist.
    m_logger.exiting(m_classname,"enableUser");
  }

/**
```

```
   * Sets the specified user to a state of disabled.
   * @param userId
   */
  public void disableUser(int userId) {
    m_logger.entering(m_classname,"disableUser");
    sendNotification(DISABLE_USER, "userId [" + userId + "] now disabled");
    // Lookup user with userId.
    // Set the status of the user to enabled.
    // throw exception if user doesn't exist.
    m_logger.exiting(m_classname,"disableUser");
  }

/**
    * Informs any interested party on what notifications this MBean emits.
    * @return the notifications this MBean emits
    */
  public MBeanNotificationInfo[] getNotificationInfo() {
    m_logger.entering(m_classname,"getNotificationInfo");
    String NOTIFICATIONS[] =
    { ENABLE_USER, DISABLE_USER };
    MBeanNotificationInfo[] info =
    {
      new MBeanNotificationInfo(NOTIFICATIONS,
                               "javax.management.Notification",
                               "Notification set for UserManager") };
    m_logger.exiting(m_classname,"getNotificationInfo");
    return info;
  }

 /**
   * Sends a JMX notification using the sendNotification method
   * from the base class.
   * @param operation - the name of the operation sending the notification.
   * @param desc - the description to place within the notification.
   * @return void
   */
  public void sendNotification(String operation, String desc) {
    m_logger.entering(m_classname,"sendNotification");
    Notification notification = new Notification(operation,this,
    System.currentTimeMillis(), desc);
    super.sendNotification(notification);
    m_logger.exiting(m_classname,"sendNotification");
  }

// Create the MBeanInfo object to allow attributes and operations exposed by the
// MBean to be retrieved
public MBeanInfo getMBeanInfo()
{
  // Get access to the Locale instance set by the JMX client
    Locale locale;
    locale = JMXStateFactory.getJMXState().getLocale();

    // Create MBean's localized metadata using the Locale from JMX client
  return createMBeanInfo(locale);
}

// Create the MBeanInfo object to allow attributes and operations exposed by the
// MBean to be retrieved
public MBeanInfo getMBeanInfo(Locale locale)
{
```

```
  // Create the MBean's metadata using the Locale from the MBeanServer
  return createMBeanInfo(locale);
}

// Create the MBean's localized metadata
private MBeanInfo createMBeanInfo(Locale locale)
{
  // Get the resource bundle for the specified locale
  ResourceBundle resource =
    ResourceBundle.getBundle("mymbeans/Messages", locale);

  MBeanInfo minfo = null;
  try
   {

   // Set the MBean constructor descriptor
      consInfoArray_ = new MBeanConstructorInfo[0];

   // Set the MBean operations descriptor

   MBeanOperationInfo[] operInfoArray_ = new MBeanOperationInfo[]{ };
      MBeanParameterInfo params;
      params = new MBeanParameterInfo[];

      // Set the description for the userId parameter
      params[0] = new MBeanParameterInfo("userId",
                 resource.getString("usermanager_userId"));

      // Set the description for the listUsers operation
      operInfoArray_[0] = new MBeanOperationInfo("listUsers",
                          resource.getString("usermanager_listUsers"),
                                               params,
                                               "",
                                               MBeanOperationInfo.ACTION)

      // Set the description for the enableUser operation
      operInfoArray_[1] = new MBeanOperationInfo("enableUser",
                          resource.getString("usermanager_enableUser"),
                                               params,
                                               "",
                                               MBeanOperationInfo.ACTION)

      // Set the description for the disableUser operation
      operInfoArray_[2] = new MBeanOperationInfo("disableUser",
                          resource.getString("usermanager_disableUser"),
                                               params,
                                               "",
                                               MBeanOperationInfo.ACTION)

   // Create the MBeanInfo object containing the localized metadata
   // The null parameter is for notifications, which are not provided by this
   // MBean
   MBeanInfo minfo = new MBeanInfo("MyManager",
                         resource.getString("mymanager_description"),
                         attribInfoArray_,
                         consInfoArray_,
                         operInfoArray_,
                         null);
   }
   catch
```

```
    {
    // Handle exceptions
   }
return minfo;
}
```

6

Working with Open Source Frameworks

This chapter discusses developing applications on open source frameworks for deployment into OC4J. It contains the following sections:

Installing Open Source Libraries in OC4J

To enable a Web application built on an open source framework to be used in OC4J, the JAR file(s) containing the open source implementation as well as any dependent libraries must be available within OC4J. Note that no open source framework libraries are available at the global level in OC4J by default.

The standard approach is to package the required JAR file(s) with the Web application on the `WEB-INF/lib` path. The library can then be deployed into OC4J with the Web application. While this is the most straightforward process, it will result an instance of the library being loaded for each Web application.

If the open source library is required by multiple Web applications, or to multiple Web modules within a single J2EE application, the OC4J proprietary *shared library* mechanism may be a more practical approach. Using this mechanism is an efficient use of system resources, as OC4J creates a single class loader for the shared library, rather than loading multiple copies of the same library.

To use the shared library mechanism, you will install the shared library on the OC4J instances hosting the applications using one of the options described in "Options for Installing and Publishing a Shared Library" on page 3-11. You can then configure specific applications to import the shared library as described in "Configuring an Application to Import a Shared Library" on page 3-14.

Making the open source framework implementation available as a shared library is a practical approach in these cases:

- You want all Web modules within a J2EE application (EAR) to use the same version of the library.

In this case, you will configure the application to import the shared library in the EAR-level `orion-application.xml` deployment descriptor as described in "Declaring Dependencies in an Application's OC4J Deployment Descriptor" on page 3-14, or in the `MANIFEST.MF` file as outlined in "Declaring Dependencies in an Application's Manifest File" on page 3-15.

- You want to ensure that all Web modules deployed into OC4J use the same version of the library.

  This requires configuring the `default` application to import the shared library, as described in "Configuring All Deployed Applications to Import a Specific Shared Library" on page 3-15. Note that if this configuration is used, an application will use the version of the shared library imported from the `default` application, even if a different version of the library is packaged with the application.

You can create shared libraries using any of the mechanisms described in "Options for Installing and Publishing a Shared Library" on page 3-11. The following example illustrates how you can easily add the Xerces parser as a shared library within an OC4J server using Application Server Control Console.

1. Click **Administration>Shared Libraries**.
2. Click **Create** on the Shared Libraries page.
3. Enter the name for the shared library. In this case, you will enter `xerces.xml`.
4. Enter the shared library version, which in this case is 2.5.0.
5. Click **Add** to upload the library JAR files to the OC4J instance. Upload the following Apache libraries:
   - `xercesImpl.jar`
   - `xml-apis.jar`

   The following shared library declaration is added to the *ORACLE_HOME*/j2ee/*instance*/server.xml file:

```
<shared-library name="xerces.mxl" version="2.5.0">
  <code-source path="xercesImpl.jar"/>
  <code-source path="xml-apis.jar"/>
</shared-library>
```

Removing Imported Oracle Shared Libraries to Avoid Conflicts

When working with open-source frameworks, it may sometimes be necessary to remove or override Oracle shared libraries imported by default to avoid conflicts with corresponding open source libraries packaged with an application.

For example, the Oracle XML parser packaged with OC4J will be used by default by any application deployed into OC4J. This is not desirable, for example, with an Axis-based application, which requires the Xerces parser library.

In this scenario, you have several options:

- Specify the shared library to remove in the `orion-application.xml` file packaged with the application

  See "Removing or Replacing an Oracle Shared Library Imported by Default" on page 3-7 for an overview of using the `<remove-inherited>` tag.

- Configure the application to not import the Oracle shared library at deployment time

See "Removing or Replacing an Oracle Shared Library Imported by Default" on page 3-7 for a step-by-step example illustrating how a shared library can be removed at deployment time.

- If the required open-source JAR file is packaged within a Web module, you can specify that OC4J search for and load this local library, which will cause it to be used instead of the default Oracle library.

  See "Using a Packaged JAR Instead of an Oracle Shared Library" on page 3-9 for a step-by-step example.

Using Jakarta Struts

The following sections provide an overview of using Jakarta Struts in an OC4J environment:

Overview of Jakarta Struts

Jakarta Struts is an open source framework for building Web applications using open standards such as Java servlets, JavaServer Pages, and XML. Applications built on the latest official release of Struts, version 1.1, can be easily deployed into the latest release of OC4J.

Struts supports a modular application development model based on the Model-View-Controller (MVC) pattern. With Struts, you can create an extensible development environment for your application, based on industry standards and proven design models. No OC4J-specific configuration is required to deploy Struts MVC-enabled Web applications into OC4J.

Struts is part of the Apache Jakarta Project, sponsored by the Apache Software Foundation. See the user guide, installation guide, and other documentation on the official Struts Web site:

`http://jakarta.apache.org/struts`

Note: The Struts documentation strongly recommends *against* installing Struts as a shared library. Specifically, the documentation notes that `ClassNotFoundException` issues may occur unless all of your application classes are stored in the shared library directory.

As such, you should package `struts.jar` and any dependency libraries with your application.

Struts Support in Oracle JDeveloper

Oracle JDeveloper 10*g* provides extensive support for building Web applications on the Struts 1.1 framework.

JDeveloper includes the source for the Struts framework in the *`jdev_install`*`/jakarta-struts/` directory. This directory contains the same Struts package, including sample Web applications, that can be downloaded from the Apache Struts home page.

A sampling of the features for Struts development provided in JDeveloper follows:

- The Struts page flow diagram lets you draw the flow of your application's Web pages using icons selected from a palette. The diagram visually represents standard Struts elements, including actions and action forwards, and automatically updates those elements in the Struts configuration file.
- The Structure window and Property Inspector let you edit the attributes of any Struts element.
- The Struts Configuration Editor allows you to directly edit the Struts configuration file.
- A large set of custom JSP tag libraries that work with the Struts framework is provided, and the Struts tag libraries are accessible from the JDeveloper Component Palette.

See the online Help provided with Oracle JDeveloper for details on Struts support.

Accessing the Struts Binary Distribution

If you are not using Oracle JDeveloper, you can download the Struts 1.1 distribution directly from Jakarta at the following site:

`http://jakarta.apache.org/site/binindex.cgi`

The sample applications, packaged as WAR files, make an excellent resource for understanding generic Struts. A good example of a WAR file configured to use Struts is provided in the `/webapps` folder of the Struts archive file as `struts-blank.war`. This example serves as a useful template when you are constructing your own Web applications.

Using the Spring Framework

This section provides an overview of using the Spring Framework 1.2 in OC4J. It includes the following topics:

- Overview of the Spring Framework
- Oracle TopLink Support in Spring 1.2
- Downloading the Spring Framework Distribution

Overview of the Spring Framework

Spring is an increasingly popular open source Java/J2EE application framework based on the Dependency Injection (DI) design pattern. OC4J provides full support for applications built on Spring Framework 1.2. This latest release of Spring also provides extensive support for Oracle TopLink integration, providing you with a platform for persistent data access.

Like Struts, Spring provides a Model-View-Controller (MVC) framework, and Web applications built on Spring MVC can be deployed seamlessly into OC4J, without requiring any OC4J-specific deployment configuration.

Spring is particularly strong in the area of persistent data access, making it an attractive framework for building Web applications that will interact with a relational database. The Spring framework itself includes a persistence layer built on the J2EE Data Access Objects (DAO) design pattern. The DAO layer integrates well with object-relational mapping (ORM) tools, including the open source Hibernate and the proven Oracle TopLink.

Oracle TopLink Support in Spring 1.2

Spring 1.2 includes extensive support for Oracle TopLink releases 9.0.4 and 10.1.3. The result of this integration is a powerful, high-performance framework for persisting *plain old Java objects* (POJOs) to relational databases.

Section 11.4 in the Spring Reference Documentation, provided with the Spring distribution as `spring-refrence.pdf`, discusses TopLink integration in considerable detail. Documentation on the various TopLink classes provided with `spring.jar` is also provided.

Downloading the Spring Framework Distribution

The Spring 1.2 distribution is available as a ZIP file from the following location:

`http://www.springframework.org/download`

The latest snapshot build can also be downloaded from this location.

The Spring Reference Documentation is provided with the distribution as `spring-refrence.pdf`. Spring also ships with several sample applications that illustrate best practices. These can be used as templates for your own applications.

Spring 1.2 is organized into seven modules, each packaged as a JAR file. The "core" module is packaged as `spring.jar` and contains all of the other modules. However, the modular structure makes it possible to provide only those JAR files that are needed, if desired.

Using Apache MyFaces

The following provides an overview of how Web applications built using Apache MyFaces work with OC4J.

- Overview of MyFaces
- Accessing the MyFaces Distribution
- JDeveloper Support for MyFaces

Overview of MyFaces

MyFaces is an open source alternative implementation of JavaServer Faces (JSF), a standard Java framework for building Web applications. MyFaces has been developed under the aegis of the Apache Foundation.

Because it is simply another "flavor" of JSF, Web applications built on MyFaces can be easily deployed into OC4J; no OC4J-specific configuration changes are necessary.

Accessing the MyFaces Distribution

You can download the MyFaces distribution from the following location:

`http://myfaces.apache.org/binary.cgi`

The framework is packaged as `myfaces.jar`, which must be made available to Web applications deployed into OC4J.

You can also download a set of Web applications packaged as WAR files that are built on the MyFaces framework from this site.

Building JSPs Using MyFaces for Deployment into OC4J

The `taglib` directive for the `core` and `html` tag libraries is the same for all JSF implementations. This means, for example, that you can use either the Sun RI or MyFaces libraries, without having to update your JSPs that have been deployed to OC4J.

If you deploy a Web application that uses MyFaces, the following libraries must be made available to the application:

- `myfaces-api.jar`
- `myfaces-impl.jar`

Alternatively, you can utilize `myfaces-all.jar`, which includes all files in these two JAR files. However, using the two separate JAR files makes it easier to "swap out" a specific library.

MyFaces also offers an open-source extension library named Tomahawk that is fully compatible with any JSF 1.1 compatible implementation, including the Sun JSF RI packaged with JDeveloper. If this component extension is used, the following library must be made available to deployed applications:

- `tomahawk.jar`

In addition, you must include the following `taglib` directive in all JSPs that will use the library (note that tools such as JDeveloper will do this for you:)

```
<%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t"%>
```

JDeveloper Support for MyFaces

Oracle JDeveloper has built-in support for the Sun JavaServer FacesReference Implementation (RI). However, you can easily configure JDeveloper to use other JSF implementations, including MyFaces, through the Import Custom JSP tag library interface.

You can have both the Sun RI and MyFaces implementations installed in JDeveloper. In fact, you can even build your JSPs using one of these standard implementations, then change to using another at deployment time simply by replacing the libraries used.

See the JDeveloper page on the Oracle Technology Network for additional information and "how to's" on using JSF, MyFaces and Oracle ADF:

```
http://www.oracle.com/technology/products/jdev/index.html
```

Using Hibernate

The following provides an overview of using Hibernate with applications deployed into OC4J.

- Accessing the Hibernate Binaries
- Using Hibernate with Applications in OC4J

Accessing the Hibernate Binaries

Hibernate is an open source object-relational mapping (ORM) tool for Java environments. Hibernate is often used in Java Swing applications, Java Servlet-based applications, or J2EE applications using EJB session beans.

You can access the Hibernate libraries and documentation from the following site:

`http://hibernate.org/`

Using Hibernate with Applications in OC4J

Both Oracle TopLink, which is packaged by default with OC4J, and Hibernate use `antlr.jar`. To avoid class collisions between the library packaged with OC4J and that packaged with your Hibernate application, you must explicitly remove the `oracle.toplink` shared library from the set of libraries that will be imported by the application.

See "Removing or Replacing an Oracle Shared Library Imported by Default" on page 3-7 for details.

Using Apache Axis

The following provides an overview on using Axis-based Web services in OC4J.

- Accessing the Axis Distribution
- Using the Xerces Parser XML Parser
- Using Oracle-Based and Axis-Based Web Services in OC4J

Accessing the Axis Distribution

Apache Axis is a well-known and widely-used framework for implementing Web services. Applications that include Web services based on Axis 1.2 and 1.3 can easily be deployed into OC4J.

You can access the Axis libraries and documentation from the following Apache site:

`http://ws.apache.org/axis/`

Using the Xerces Parser XML Parser

Axis applications typically utilize the Xerces XML parser by default. However, Axis applications can safely use the Oracle XML parser packaged with OC4J if the Xerces parser is not packaged with the application.

The Oracle XML parser is configured to be used by all deployed applications by default—even the Xerces libraries are packaged within your Web module (WAR) file. As such, if you do want Axis-based applications to use the Xerces parser, you must use one of the shared library mechanisms provided by OC4J to ensure that the Xerces parser is used by a given Axis-based application. See "Removing Imported Oracle Shared Libraries to Avoid Conflicts" on page 6-2 for details.

Note that attempting to remove the Oracle XML parser will result in an error if the affected Web service includes Oracle's JAX-RPC and/or SAAJ libraries (`jaxrpc-api.jar` and `saaj-api.jar`).

Using Oracle-Based and Axis-Based Web Services in OC4J

Axis and Oracle each provide different implementations of several key Web services libraries, such as JAX-RPC and WSDL libraries. Table 6–1 below lists these common libraries.

Table 6–1 Web Services Libraries

Oracle	Axis
`jaxrpc-api.jar`	`jax-rpc.jar`
`saaj-api.jar`	`saaj.jar`
`orawsdl.jar`	`wsdl4j-1.5.1.jar`

Because these libraries currently contain the same class implementations, class loading issues are not expected when an Axis application is deployed into OC4J.

However, a single application (EAR) cannot contain one Web service that uses Axis, and another that uses Oracle. Specifically, an Axis Web service calling out to another Web service via a Web services client created using JAX-RPC implementation is not supported.

Configuring and Using Jakarta log4j

The following sections cover considerations for using Jakarta log4j in an OC4J environment:

- Overview of Jakarta log4j
- Downloading the log4j Binary Distribution
- Using log4j Configuration Files
- Enabling log4j Debug Mode in OC4J

Overview of Jakarta log4j

The log4j framework is an open source project of the Apache Jakarta Project, sponsored by the Apache Software Foundation. The framework provides an efficient and flexible API to support runtime logging operations for Java applications. It enables developers to insert log statements into their code, incorporating messages at different levels of alarm as desired. Log4j also enables system administrators to separately define the level of logging they wish to see from the application at runtime, without requiring changes to the supplied application code.

Features of log4j allow you to enable logging at runtime without having to modify the application binary file. Statements can remain in shipped code without incurring significant performance cost. Logging is controlled through a configuration file without requiring changes to the application binary.

The sections that follow describe how to install the log4j library and configure it for use with OC4J. Use of the extensive log4j API is not OC4J-specific, so is not covered in this document. See the documentation on the official log4j Web site:

`http://jakarta.apache.org/log4j/docs/index.html`

Downloading the log4j Binary Distribution

The log4j distribution is available at the following location:

`http://jakarta.apache.org/log4j/docs/download.html`

Download the archive file from this location, choosing the appropriate format (ZIP file or compressed TAR file) for your platform, and save it to your local file system.

Using log4j Configuration Files

The log4j framework enables you to control logging behavior through settings specified in an external configuration file, allowing you to make changes to the logging behavior without modifying application code.

There are three common ways to use the external configuration files. Each approach defines what the configuration files are named and how they are located by the J2EE application server at runtime.

The following sections describe the three approaches:

- Using the Default Files for Automatic log4j Configuration
- Using Alternative Files for Automatic log4j Configuration
- Programmatically Specifying External Configuration Files

Using the Default Files for Automatic log4j Configuration

By default, log4j uses a configuration file named `log4j.properties` or `log4j.xml` to determine its logging behavior. It automatically attempts to load these files from the class loaders available to it at runtime. If it finds both files, then `log4j.xml` takes precedence.

To use an automatic configuration file, place it in a directory location that falls within the classpath used by OC4J. This includes, in order of loading precedence:

1. Global application level: `/j2ee/`*`instance`*`/applib`
2. Web application level: `/WEB-INF/classes`

Note: A log4j runtime is configured only once, using the automatic configuration files, when the first call is made to the `org.apache.log4j.Logger` class.

If you install the log4j library at the global application level, by placing it in the `/j2ee/`*`instance`*`/applib` directory, then you can use only one automatic configuration file to define all the log levels, appenders, and other log4j properties for all the applications running on your server.

If you install the log4j library separately for each Web application, in each `/WEB-INF/lib` directory, then the log4j logger is initialized separately for each Web application. This enables you to use separate automatic log4j configuration files for each Web application.

Visit the following log4j Web site and see the log4j user mailing list for more information:

http://www.mail-archive.com/log4j-user@jakarta.apache.org/

Using Alternative Files for Automatic log4j Configuration

You can choose alternative file names instead of using the default names for automatic configuration of log4j. To do this, specify an additional runtime property when OC4J is started, as follows, where "`%`" is the system prompt and *`url`* designates the location of the configuration file to use:

```
java -Dlog4j.configuration=url
```

If the specified value for the `log4j.configuration` property is a fully formed URL, log4j loads the URL directly and uses that as the configuration file.

If the specified value is not a correctly formed URL, log4j uses the specified value as the name of the configuration file to load from the class loaders it has available.

For example, assume OC4J is started as follows (where this is a single wraparound command line):

```
java -Dlog4j.debug=true -Dlog4j.configuration=file:///d:\temp\foobar.xml
       -jar oc4j.jar
```

In this case, log4j tries to load the file `d:\temp\foobar.xml` as its configuration file.

As another example, assume OC4J is started as follows:

```
java -Dlog4j.debug=true -Dlog4j.configuration=foobar.xml -jar oc4j.jar
```

In this case, log4j tries to load `foobar.xml` from the class loaders it has available. This works in the same manner as using the default automatic configuration file `log4j.xml`, but using the specified file name instead.

Note: This approach, although offering an additional level of flexibility, does require all external configuration files for all deployed applications to have the same name.

Programmatically Specifying External Configuration Files

Instead of relying on the automatic configuration file loading mechanism, some applications use a programmatic approach to load the external configuration file. In this case, the path to the configuration file is supplied directly within the application code. This allows different file names to be used for each application. The log4j utility loads and parses the specified configuration file (either an XML document or a properties file) to determine required logging behavior.

Here is an example:

```
public void init(ServletContext context) throws ServletException
{
  // Load the barfoo.xml file as the log4j external configuration file.
  DOMConfigurator("barfoo.xml");
  logger = Logger.getLogger(Log4JExample.class);
}
```

In this case, log4j tries to load `barfoo.xml` from the same directory from which OC4J was started.

Using the programmatic approach provides the most flexibility to developers and system administrators. A configuration file can be of any arbitrary name and be loaded from any location. System administrators can still make changes to the logging behavior without requiring application code changes through the external configuration file.

To provide even further flexibility, and to avoid coding a specific name and location into an application, a useful technique is to supply the file name and location as parameters inside the standard `web.xml` deployment descriptor. The servlet or JSP page reads the values of the parameters specifying the location and name of the configuration file, and dynamically constructs the location from which to load the configuration file. This process enables system administrators to choose both the name and location of the configuration file to use.

Here is a sample `web.xml` entry specifying the name and location of the configuration file:

```
<context-param>
  <param-name>log4j-config-file</param-name>
  <param-value>/web-inf/classes/app2-log4j-config.xml</param-value>
</context-param>
```

The application reads the location value from the deployment descriptor, constructs a full path to the file on the local file system, and loads the file. Following is some sample code:

```
public void init(ServletContext context) throws ServletException
{
  /*
   * Read the path to the config file from the web.xml file,
   * should return something like /web-inf/xxx.xml or web-inf/classes/xxx.xml.
   */
  String configPath = context.getInitParameter("log4j-config-file");

  /*
   * This loads the file based on the base directory of the Web application
   * as it is deployed on the application server.
   */
  String realPath = context.getRealPath(configPath);
   if(realPath!=null)
   DOMConfigurator.configure(realPath);
  _logger = Logger.getLogger(Log4JExample.class);
}
```

> **Note:** It is a good practice to place files that define behavior, and that should not be accessible to clients from an HTTP request, directly into the `/WEB-INF` directory of the Web application. (Do not use a subdirectory of `/WEB-INF`.) This applies to `log4j.xml`, for example. The servlet specification requires contents of the `/WEB-INF` directory to be inaccessible to clients.

Enabling log4j Debug Mode in OC4J

When deploying an application on OC4J that uses log4j and external configuration files, it is sometimes helpful to view how log4j is trying to find and load the requested configuration files. To facilitate this, log4j provides a debug mode that displays how it is loading (or attempting to load) its configuration files.

To turn on log4j debug mode, specify an additional runtime property when you start OC4J, as follows:

```
java -Dlog4j.debug=true -jar oc4j.jar
```

OC4J displays output similar to the following:

```
Oracle Application Server (9.0.4.0.0) Containers for J2EE initialized
log4j: Trying to find [log4j.xml] using context classloader [ClassLoader:
[[D:\myprojects\java\log4j\app1\webapp1\WEB-INF\classes],
[D:\myprojects\java\log4j\app1\webapp1\WEB-INF\lib/log4j-1.2.7.jar]]].
log4j: Using URL [file:/D:/myprojects/java/log4j/app1/webapp1/WEB-INF/classes/
log4j.xml] for automatic log4j configuration.
log4j: Preferred configurator class: org.apache.log4j.xml.DOMConfigurator
log4j: System property is :null
```

```
log4j: Standard DocumentBuilderFactory search succeeded.
log4j: DocumentBuilderFactory is oracle.xml.jaxp.JXDocumentBuilderFactory
log4j: URL to log4j.dtd is [classloader:/org/apache/log4j/xml/log4j.dtd].
log4j: debug attribute= "null".
log4j: Ignoring debug attribute.
log4j: Threshold ="null".
log4j: Level value for root is [debug].
log4j: root level set to DEBUG
log4j: Class name: [org.apache.log4j.FileAppender]
log4j: Setting property [file] to [d:/temp/webapp1.out].
log4j: Setting property [append] to [false].
log4j: Parsing layout of class: "org.apache.log4j.PatternLayout"
log4j: Setting property [conversionPattern] to [%n%-5p %d{DD/MM/yyyy}
d{HH:mm:ss} [%-10c] [%r]%m%n].
log4j: setFile called: d:/temp/webapp1.out, false
log4j: setFile ended
log4j: Adding appender named [FileAppender] to category [root].
```

Note: You can also use the `debug` attribute of the `log4j:configuration` tag in an external configuration file to enable debug output. However, this does not display the loading operations that take place, so does not offer the best service for resolving problems in loading configuration files.

7

Packaging an Application for Deployment into OC4J

This chapter provides guidelines for packaging J2EE-compliant applications and modules for deployment into OC4J. It includes the following sections:

- Overview of J2EE Application Packaging
- Packaging Deployment Descriptors

Overview of J2EE Application Packaging

The proper packaging of J2EE applications and modules is critical to successful deployment of your creations into OC4J. As a fully J2EE 1.4-compliant container, OC4J supports the packaging of J2EE applications into different modules which can then be deployed together or separately. Modules can be categorized as belonging to one of two types:

- Application modules containing a complete J2EE application or a Web application
- Standalone modules containing Enterprise JavaBeans (EJBs), application clients, or resource adapters

A complete J2EE application is assembled and delivered as an Enterprise Archive (EAR) file, a standard Java Archive (JAR) file ending in a `.ear` extension. The file includes a J2EE application descriptor and may optionally include an OC4J-specific descriptor and, with JDK 5.0, a `lib` directory. An EAR is essentially a metacontainer that includes one or more of the following J2EE modules:

- A Web application module is packaged as a Web Application Archive (WAR) file. A WAR typically contains the servlet class files, JSP files, any supporting class files (such as resource bundles), images, and HTML files that comprise a Web application. A J2EE Web deployment descriptor and an optional OC4J-specific Web descriptor are also included.

  Note that Web services are packaged in a WAR file for deployment.
- EJB modules, which contain class files for enterprise beans, are packaged as JAR files. An EJB JAR will also include an EJB deployment descriptor.
- Resource adapter modules containing all Java interfaces, classes, native libraries, and the resource adapter deployment descriptor are packaged as JAR files with a `.rar` (Resource Adapter Archive) extension.
- Application client modules containing class files and an application client deployment descriptor are packaged as JAR files.

Note that any of the preceding modules can be deployed individually into OC4J, rather than having to be packaged within an EAR file as part of a complete J2EE application. This makes it possible to deploy updated modules without having to redeploy the entire application.

> **Note:** Web services can be packaged as a WAR file or as an EJB JAR file containing stateless session beans.

The proper packaging of J2EE applications and modules is critical to successful deployment of your creations into OC4J. The following sections review the structure of these files:

- J2EE Application Structure Within OC4J
- Application Module (EAR File and WAR File) Structures

J2EE Application Structure Within OC4J

The following is the standard J2EE application structure, which you can use as your development and packaging structure as appropriate.

This discussion also shows the relative locations of optional OC4J-specific descriptors, such as `orion-application.xml`. As noted previously, if you do not include the OC4J-specific descriptors in your deployment, OC4J generates them for you when you deploy a J2EE application, using values inherited from corresponding OC4J configuration files and J2EE descriptors as defaults.

```
J2EEApplicationName.ear
   WAR file(s)
   JAR file(s)
   RAR file(s)
   lib/
      JAR and ZIP files for shared classes
   META-INF/
      MANIFEST.MF
      application.xml (standard J2EE application descriptor)
      orion-application.xml (optional OC4J application descriptor)

   WebModuleName.war
      static HTML files, such as index.html
      JSP pages
      images
      WEB-INF/
         web.xml (standard J2EE Web descriptor)
         orion-web.xml (optional OC4J Web descriptor)
         classes/
             servlet classes, according to package
         lib/
             JAR and ZIP files for shared classes

   EJBModuleName.jar
      EJB classes, according to package
      META-INF/
         ejb-jar.xml (standard J2EE descriptor)
         orion-ejb-jar.xml (optional OC4J descriptor)

   WebServiceWebModuleName.war
      WEB-INF/
        web.xml (standard J2EE Web descriptor)
```

```
      orion-web.xml (optional OC4J Web descriptor)
      webservices.xml (standard J2EE descriptor)
      oracle-webservices.xml (optional OC4J descriptor)
      mapping_file.xml
    wsdl/
      wsdl_file.wsdl
    classes/
      class files
    lib/
      JAR and ZIP files for dependency classes

ApplicationClientModuleName.jar
    client classes, according to package
    META-INF/
       application-client.xml (standard J2EE descriptor)
       orion-application-client.xml (optional OC4J descriptor)

ResourceAdapterModuleName.rar
    JAR files for required classes
    required static files or other files
    META-INF/
       ra.xml (standard J2EE descriptor)
       orion-ra.xml (optional OC4J descriptor)
```

Note: This structure is defined in the J2EE specification and related specifications. The J2EE specification is at the following location:

http://java.sun.com/j2ee/docs.html

Application Module (EAR File and WAR File) Structures

In J2EE, a WAR file is typically contained within an EAR file. In the example in the preceding section, the EAR file, *J2EEAppName*.ear, would have its /META-INF directory at the top level, along with Web module WAR files, EJB module JAR files, client application JAR files, and resource adapter RAR files (zero or more of each, as applicable):

```
META-INF/
   application.xml
   orion-application.xml (optional)
EJBModuleName.jar
WebModuleName.war
ClientModuleName.jar
ResourceAdapterModuleName.rar
```

The following examples show the structure of the archive files for a simple Web application. The EAR file contains a WAR file, which contains a single servlet.

Note: This document assumes you have some J2EE development experience and a means of creating EAR and WAR files, either by using the JAR utility directly, through an IDE such as Oracle JDeveloper, or by using the `ant` utility and a `build.xml` file. For information about using the OC4J Ant tasks, see the *Oracle Containers for J2EE Deployment Guide*. For information about `ant`, see the following Web site :

http://ant.apache.org

Sample EAR File

Following are the contents of `utility.ear`. If there were EJB, client application, or resource adapter modules, the associated JAR files would be at the same level as the WAR file. Optionally, you could also include an `orion-application.xml` file in the `/META-INF` directory. Instead, in this example, one would be generated by OC4J during deployment.

```
META-INF/MANIFEST.MF
META-INF/application.xml
utility_web.war
```

Sample WAR File

Here are the contents of `utility_web.war`. Optionally, you could also include an `orion-web.xml` file in the `/WEB-INF` directory. Instead, in this example, one would be generated by OC4J during deployment.

```
META-INF/MANIFEST.MF
WEB-INF/classes/TestStatusServlet.class
WEB-INF/web.xml
index.html
```

> **Note:** The `MANIFEST.MF` files are created automatically by the JAR utility.

Packaging Deployment Descriptors

The initial configuration data required to deploy an application or module into a J2EE container is specified in an XML-formatted configuration file known as a *deployment descriptor*. The format of a deployment descriptor is defined in a Document Type Definition (DTD) document or an XML Schema Definition (XSD).

Each deployable module has a standard J2EE deployment descriptor that is required for deployment into a J2EE-compatible server. In addition, J2EE containers such as OC4J utilize 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.

You can create and package the appropriate OC4J-specific descriptor with a deployable archive. However, this is not required; if OC4J does not find a packaged descriptor at deployment time, a *deployment plan* is automatically generated using default values inherited from corresponding OC4J configuration files and J2EE descriptors as defaults. See the *Oracle Containers for J2EE Deployment Guide* for more on deployment plans.

Deployment Descriptors Overview

Table 7–1 provides a description of each J2EE standard deployment descriptor and its corresponding OC4J extension. 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
```

Note that OC4J now provides the ability to create a *deployment plan* that consolidates all of the OC4J-specific configuration data that is persisted within the various OC4J descriptor files. You can use the deployment plan editor functionality to set or edit configuration data at deployment time. See the *Oracle Containers for J2EE Deployment Guide* for more on working with deployment plans.

Table 7–1 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 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 EJBs, 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 are enabled. The format of this file is defined by `orion-web-10_0.xsd`.
`ejb-jar.xml`	`orion-ejb-jar.xml`
Defines the specific structural characteristics and dependencies of the Enterprise JavaBeans within a JAR, and provides instructions for the EJB container about how the beans expect to interact with the container. If you are using EJB 3.0, this deployment descriptor file is optional; you can use annotations instead. In this release, OC4J supports the use of both EJB 3.0 annotations and `ejb-jar.xml` for all options of session and message-driven beans. The `ejb-jar.xml` file is not used for EJB 3.0 entities. Configuration in the `ejb-jar.xml` file overrides annotations. For EJB 30. entities, you must either use annotations or TopLink JPA persistence provider deployment XML files (`toplink-ejb-jar.xml` and `ejb3-toplink-sessions.xml`). For more information, see the *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide*.	Defines OC4J-specific configuration data for all EJBs within an archive, including EJB pool settings, time-out and retry settings, JNDI mappings and finder method specifications. Also includes properties for the TopLink persistence manager. The format of this file is defined in `orion-ejb-jar-10_0.xsd`.
`persistence.xml`	
Defines one or more persistence units in an EJB 3.0 application that uses entities. In this release, you can define `persistence.xml` in an EJB JAR, WAR, or EAR. This deployment descriptor file can be packaged in the `META-INF` directory of an EJB-JAR file, in the `WEB-INF/classes/META-INF` directory of a Web module, in any standard JAR packaged in the `lib` directory of an EAR, or in a `WEB-INF/lib` directory that packages entities. For more information, see the *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide*.	

Table 7–1 (Cont.) J2EE and OC4J Deployment Descriptors

J2EE Standard Descriptors	OC4J Proprietary Descriptors
`application-client.xml`	`orion-application-client.xml`
Describes the EJB modules and other resources used by a J2EE application client packaged in an archive.	Contains OC4J deployment data, including JNDI mappings to an EJB's home interface or to external resources such as a data source, JMS queue or mail session. The file format is defined in `orion-application-client-10_0.xsd`.
`ra.xml`	`oc4j-ra.xml`
Contains information on implementation code, configuration properties and security settings for a resource adapter packaged within a RAR file.	Contains deployment configurations for deploying resource adapters into OC4J. It contains EIS connection information, JNDI name to be used, 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`	`oracle-webservices.xml`
Describes a Web service, including WSDL information and JAX-RPC mapping data, for a Web Service application packaged within a WAR file.	Defines properties used by the OC4J Web services container, such as whether to expose the WSDL file. It also defines endpoint addresses and data specific to EJBs implemented as Web services. The file can be packaged in a WAR or an EJB JAR. This file is formatted according to `oracle-webservices-10_0.xsd`.

In addition to the extensions noted above, additional OC4J-specific descriptors are provided to allow data sources and security role mappings to be defined. The J2EE specification does not provide standard descriptors for defining such resources. As with the OC4J specific descriptors described above, these descriptors can optionally be packaged with an application for deployment into OC4J.

Table 7–2 Additional OC4J-Specific Descriptors

Descriptor	Overview
`data-sources.xml`	Defines one or more data sources to be used by the application to connect to one or more databases. Data sources offer a portable, vendor-independent method for creating connections to databases. A data source's properties are set so that it represents a particular database. The format of this file is defined by `data-sources-10_1.xsd`.
`jazn-data.xml`	Can optionally be supplied with an application or module when the XML provider type is specified. Stores JAAS (Java Authentication and Authorization Service) data on users and roles For more information about the `jazn-data.xml` file, see the *Oracle Containers for J2EE Security Guide*.

Table 7–2 (Cont.) Additional OC4J-Specific Descriptors

Descriptor	Overview
`system-jazn-data.xml`	Contains the security configuration for the OC4J instance. The `jazn-data.xml` descriptor can be specified, however, at the application level to define users and roles. For more information about the `system-jazn-data.xml` file, see the *Oracle Containers for J2EE Security Guide*.

Packaging a J2EE Standard Application Descriptor (application.xml)

The J2EE standard defines the concept and format of an application descriptor, called `application.xml`, that you must include in the `/META-INF` directory of the EAR file containing a J2EE application. The application descriptor acts as a manifest of the modules contained in the application, possibly with additional configuration information as well, and in some development environments can be created automatically for you.

See the J2EE specification for more information.

Here is an example for an application with an EJB module, a Web module, and an application client module:

```
<?xml version="1.0" ?>
<!DOCTYPE application (View Source for full doctype...)>
<application>
  <display-name>stateful, application:</display-name>
  <description>
    A sample J2EE application that uses a remote stateful session
    bean to call a local entity bean.
  </description>
  <module>
    <ejb>stateful-ejb.jar</ejb>
  </module>
  <module>
    <web>
      <web-uri>stateful-web.war</web-uri>
      <context-root>/stateful</context-root>
    </web>
  </module>
  <module>
    <java>stateful-client.jar</java>
  </module>
</application>
```

Packaging an OC4J-Specific Application Descriptor (orion-application.xml)

The J2EE standard application descriptor, `application.xml`, is extended by an OC4J-specific application-level application descriptor, `orion-application.xml`, which contains additional OC4J-specific configuration data. You can optionally provide an `orion-application.xml` file with the `application.xml` file in the `/META-INF` directory of your EAR file.

If you include an `orion-application.xml` file in your EAR file, OC4J initializes the *deployment plan* created during the application deployment process with the values specified in the file. For details on creating or editing deployment plans at deployment time, see the *Oracle Containers for J2EE Deployment Guide*.

This data can optionally be edited at deployment time using the deployment plan editor functionality provided by the Oracle Enterprise Manager 10*g* Application Server

Control Console and JDeveloper. The finalized data is then written out to a generated `orion-application.xml` file in the *`ORACLE_HOME`*`/j2ee/`*`instance`*`/application-deployments` directory.

If the EAR file does not contain an `orion-application.xml` file, OC4J generates the file in the deployment directory as noted above, using default settings inherited from the OC4J global application descriptor (assuming the OC4J default application is the parent application, as is the case by default) and the `application.xml` file is within the EAR file.

See "J2EE Application Structure Within OC4J" on page 7-2 for information about where `orion-application.xml` fits in the application structure.

Note: When OC4J generates `orion-application.xml`, it may make changes to the file but these changes are transparent. For example, an attribute setting that specifies the default value may be ignored or removed.

In most circumstances, you should use `orion-application.xml` only to define OC4J-specific configuration such as security role mappings. Also note that when OC4J generates the file, it creates `<web-module>` elements to reflect the modules specified in the J2EE `application.xml` file.

The following example shows some OC4J-specific configuration and defines the same EJB, Web, and client modules as defined in the example of the standard `application.xml` file in "Packaging a J2EE Standard Application Descriptor (application.xml)" on page 7-7:

```
<orion-application default-data-source="jdbc/OracleDS">
  <ejb-module remote="false" path="stateful-ejb.jar" />
  <web-module id="stateful-web" path="stateful-web.war" />
  <client-module path="stateful-client.jar" auto-start="false" />
  <persistence path="persistence" />
  <log>
    <file path="application.log" />
  </log>
  <namespace-access>
    <read-access>
      <namespace-resource root="">
        <security-role-mapping name="&lt;jndi-user-role&gt;">
          <group name="users" />
        </security-role-mapping>
      </namespace-resource>
    </read-access>
    <write-access>
      <namespace-resource root="">
        <security-role-mapping name="&lt;jndi-user-role&gt;">
          <group name="users" />
        </security-role-mapping>
      </namespace-resource>
    </write-access>
  </namespace-access>
</orion-application>
```

8

Using J2EE Best Practices

This chapter provides recommended best practices to consider when developing J2EE applications for deployment into OC4J. It includes the following sections:

- JavaServer Pages Best Practices
- Class Loading Best Practices
- Sessions Best Practices
- Enterprise JavaBeans Best Practices

JavaServer Pages Best Practices

The following sections discuss best practices to consider when developing JSP pages for deployment into OC4J.

- Beware of HTTP Sessions
- Unbuffer JSP Pages
- Forward to JSP Pages Instead of Using Redirects
- Hide JSP Pages from Direct Invocation to Limit Access
- Use JSP-Timeout for Efficient Memory Utilization
- Package JSP Files in an EAR File for Deployment

Beware of HTTP Sessions

HTTP sessions add performance overhead to your Web applications due to the amount of memory used. Sessions are enabled in JSP by default.

Avoid Using HTTP Sessions

Avoid using HTTP session objects if they are not required. If a JSP page does not require an HTTP session (essentially, does not require storage or retrieval of session attributes), then you can specify that no session is to be used. Specify this with a `page` directive such as the following:

```
<%@ page session="false" %>
```

This will improve the performance of the page by eliminating the overhead of session creation or retrieval.

Note that although servlets by default do *not* use a session, JSP pages by default *do* use a session.

Always Invalidate Sessions When No Longer in Use

If your JSPs do use HTTP sessions, ensure that you explicitly cancel each session, using the `javax.servlet.http.HttpSession.invalidate()` method to release the memory occupied.

The default session timeout for OC4J is 20 minutes. You can change this for a specific application by setting the `<session-timeout>` parameter in the `<session-config>` element of the application's `web.xml` file.

Pretranslate JSP Pages Using the ojspc Utility

You might consider using the `ojspc` utility to pretranslate JSP pages before deployment. This avoids the performance cost of translating pages as they are first accessed by users. See the *Oracle Containers for J2EE Support for JavaServer Pages Developer's Guide* for additional discussion of the advantages of pretranslation.

Unbuffer JSP Pages

Unbuffer JSP pages. By default, a JSP page uses an area of memory known as a *page buffer*. This buffer (8 KB by default) is required if the page uses dynamic globalization support content type settings, forwards, or error pages. If it does not use any of these features, you can disable the buffer in a `page` directive:

```
<%@ page buffer="none" %>
```

This will improve the performance of the page by reducing memory usage and saving the output step of copying the buffer.

Forward to JSP Pages Instead of Using Redirects

You can pass control from one JSP page to another using one of two options: Including a `<jsp:forward>` standard action tag or passing the redirect URL to `response.sendRedirect()` in a scriptlet.

The `<jsp:forward>` option is faster and more efficient. When you use this standard action, the forwarded target page is invoked internally by the JSP runtime, which continues to process the request. The browser is totally unaware that the forward has taken place, and the entire process appears to be seamless to the user.

When you use `sendRedirect()`, the browser actually has to make a new request to the redirected page. The URL shown in the browser is changed to the URL of the redirected page. In addition, all request scope objects are unavailable to the redirected page because redirect involves a new request.

Use a redirect only if you want the URL to reflect the actual page that is being executed in case the user wants to reload the page.

Hide JSP Pages from Direct Invocation to Limit Access

There are situations, particularly in an architecture such as Model-View-Controller (MVC), where you would want to ensure that some JSP pages are accessible only to the application itself and cannot be invoked directly by users.

As an example, assume that the front-end, or *view*, page is `index.jsp`. The user starts the application through a URL request that goes directly to that page. Further assume that `index.jsp` includes a second page, `included.jsp`, and forwards to a third page, `forwarded.jsp`, and that you do not want users to be able to invoke these directly through a URL request.

A mechanism for this is to place `included.jsp` and `forwarded.jsp` in the application `/WEB-INF` directory. When located there, the pages cannot be directly invoked through URL request. Any attempt would result in an error report from the browser.

The page `index.jsp` would have the following statements:

```
<jsp:include page="WEB-INF/included.jsp"/>
...
<jsp:forward page="WEB-INF/forwarded.jsp"/>
```

The application structure would be as follows, including the standard `classes` directory for any servlets, JavaBeans, or other classes, and including the standard `lib` directory for any JAR or ZIP files:

```
index.jsp
WEB-INF/
   web.xml
   included.jsp
   forwarded.jsp
   classes/
   lib/
```

Use JSP-Timeout for Efficient Memory Utilization

Set the jsp-timeout attribute of the `<orion-web-app>` element to 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 where some pages are called infrequently. The default value is `0`, indicating no timeout.

The `<orion-web-app>` element is found in the OC4J `global-web-application.xml` and `orion-web.xml` files. Modify the `global-web-application.xml` file to apply the timeout to all applications in an OC4J instance. To set configuration values to a specific application, set the file in the application-specific `orion-web.xml` file.

Package JSP Files in an EAR File for Deployment

OC4J supports deployment of JSP pages by copying the files directly to the appropriate location. This is very useful when developing and testing the pages.

However, this practice is not recommended for releasing your JSP-based application for production. Always package JSP files in an Enterprise Archive (EAR) file to allow deployment in a standard manner and to allow deployment across multiple application servers.

Class Loading Best Practices

See "Best Practices for Class Loading" on page 3-17 for J2EE class loading best practices.

Sessions Best Practices

The following sections discuss best practices to consider with regard to sessions.

- Persist Session State If Appropriate
- Do Not Store Shared Resources in Sessions
- Set Session Timeout Appropriately

- Monitor Session Memory Usage
- Use a Mix of Cookies and Sessions
- Use Coarse Objects Inside HTTP Sessions
- Use Transient Data in Sessions Whenever Appropriate
- Invalidate Sessions
- Miscellaneous Guidelines

Persist Session State If Appropriate

HTTP Sessions are used to preserve the conversation state with a Web browser. As such, they hold information, which if lost, could result in a client having to start the conversation over.

Hence, it is always safe to save the session state in database. However, this imposes a performance penalty. If this overhead is acceptable, then persisting sessions is indeed the best approach.

There are trade-offs when implementing state safety that affect performance, scalability, and availability. If you do not implement state-safe applications, then:

- A single JVM process failure will result in many user session failures. For example, work done shopping online, filling in a multiple page form, or editing a shared document will be lost, and the user will have to start over.
- Not having to load and store session data from a database will reduce CPU overhead, thus increasing performance.
- Having session data clogging the JVM heap when the user is inactive reduces the number of concurrent sessions a JVM can support, and thus decreases scalability.

In contrast, a state safe application can be written so that session state exists in the JVM heap for active requests only, which is typically 100 times fewer than the number of active sessions.

To improve performance of state safe applications:

- Minimize session state. For example, a security role might map to detailed permissions on thousands of objects. Rather than store all security permissions as session state, just store the role id. Maintain a cache, shared across many sessions, mapping role id to individual permissions.
- Identify key session variables that change often, and store these attributes in a cookie to avoid database updates on most requests.
- Identify key session variables that are read often, and use `HttpSession` as a cache for that session data in order to avoid having to read it from the database on every request. You must manually synchronize the cache, which requires care to handle planned and unplanned transaction rollback.

Do Not Store Shared Resources in Sessions

Objects that are stored in the session objects will not be released until the session times out (or is invalidated). If you hold any shared resources that have to be explicitly released to the pool before they can be reused (such as a JDBC connection), then these resources may never be returned to the pool properly and can never be reused.

Set Session Timeout Appropriately

Set session timeout appropriately (`setMaxInactiveInterval()`) so that neither sessions timeout frequently nor does it live for ever this consuming memory.

Monitor Session Memory Usage

Monitor the memory usage for the data you want to store in session objects. Make sure there is sufficient memory for the number of sessions created before the sessions time out.

Use a Mix of Cookies and Sessions

Typically, a cookie is set on the Web browser (automatically by the container), to track a user session. In some cases, this cookie may last a much longer duration than a single user session. (For example, one time settings, such as to determine the end-user geographic location).

Thus, a cookie that persists on the client disk could be used to save information valid for the long-term, while a server side session will typically include information valid for the short-term.

In this situation, the long-term cookie should be parsed on only the first request to the server, when a new session established. The session object created on the server should contain all the relevant information, so as not to require reparsing the cookie on each request.

A new client-side cookie should then be set that contains only an ID to identify the server-side session object. This is automatically done for any JSP page that uses sessions.

This gives performance benefit since the session object contents do not have to be re-created from the long-term cookie. The other option is to save the user settings in a database on the server, and have the user login. The unique userid can then be used to retrieve the contents from the database and store the information in a session.

Use Coarse Objects Inside HTTP Sessions

Oracle Application Server automatically replicates sessions when session object is updated. If a session object contains granular objects, (for example, a person's name), it results in too many update events to all the servers in the island. Hence, it is recommended to use coarse objects, (for example the person object, as opposed to the name attribute), inside the session.

Use Transient Data in Sessions Whenever Appropriate

Oracle Application Server does not replicate transient data in a session across servers in the island. This reduces the replication overhead (and also the memory requirements). Hence, use the transient type liberally.

Invalidate Sessions

The number of active users is generally quite small compared to the number of users logged in on the system. For example, of the 100 users on a Web site, only 10 may actually be doing something at any given time.

A session is typically established for each user on the system. Each session, of course, uses memory.

Simple things, like a logout button, provide an opportunity for quick session invalidation and removal. This avoids memory usage growth since the sessions on the system will be closer to the number of active users, as opposed to all those that have not timed out yet.

Miscellaneous Guidelines

- Use sessions as light-weight mechanism by verifying session creation state.
- Use cookies for long-standing sessions.
- Put recoverable data into sessions so that they can be recovered if the session is lost. Store non-recoverable data persistently (in file system or in database using JDBC). However, storing every data persistently is an expensive thing. Instead, one can save data in sessions and use `HttpSessionBindingListener` or other events to flush data into persistent storage during session close.
- Sticky versus Distributable Sessions
  - Distributable session data must be serialized and useful for failover. However it is expensive, as the data has to be serialized and replicated among peer processes.
  - Sticky sessions affect load-balancing across multiple JVMs, but are less expensive as there is no state replication.

Enterprise JavaBeans Best Practices

This section describes Enterprise JavaBeans (EJBs) best practices. It includes the following topics:

- Use Local, Remote, and Message-Driven EJBs When Appropriate
- Use EJBs Judiciously
- Use a Service Locator Pattern
- Cluster Your EJBs
- Index Secondary Finder Methods
- Understand the Lifecycle of EJBs
- Use Deferred Database Constraints
- Create a Cache with Read-Only EJBs
- Pick an Appropriate Locking Strategy
- Understand and Leverage Patterns
- When Using Entity Beans, Use Container-Managed Aged Persistence Whenever Possible
- Entity Beans Using Local interfaces Only
- Use a Session Bean Facade for Entity Beans
- Enforce Primary Key Constraints at the Database Level
- Use a Foreign Key for 1-1 or 1-M Relationships
- Avoid the findAll() Method on Entities Based on Large Tables
- Set prefetch-size to Reduce Round Trips to Database

- Use Lazy Loading with Caution
- Avoid Performing O-R Mapping Manually

Use Local, Remote, and Message-Driven EJBs When Appropriate

EJBs can be local or remote. If you envision calls to an EJB to originate from the same container as the one running the EJB, local EJBs are better since they do not entail the marshalling, unmarshalling, and network communication overhead. The local beans also allow you to pass an object-by-reference, thus, improving performance further.

Remote EJBs allow clients to be on different machines and/or different application server instances to talk to them. In this case, it is important to use the value object pattern to improve performance by reducing network traffic.

If you choose to write an EJB, write a local EJB over a remote EJB object. Since the only difference is in the exception on the EJB object, almost all of the implementation bean code remains unchanged.

Additionally, if you do not have a need for making synchronous calls, message driven beans are more appropriate.

Use EJBs Judiciously

An EJB is a reusable component backed by component architecture with several useful services: persistence, transactions security, naming, etc. However, these additions make it "heavy."

If you just require abstraction of some functionality and are not leveraging the EJB container services, you should consider using a simple JavaBean, or implement the required functionality using JSPs or servlets.

Use a Service Locator Pattern

Most J2EE services and resources require "acquiring" a handle to them via an initial Java Naming and Directory Interface (JNDI) call. These resources could be an EJB Home object, or, a JMS topic. This results in expensive calls to the server machine to resolve the JNDI reference, even though the same client may have gone to the JNDI service for a different thread of execution to fetch the same data.

Hence, it is recommended to have a **Service Locator**, which in some sense is a local proxy for the JNDI service, so that the client programs talk to the local service locator, which in turn talks to the real JNDI service, and that only if required.

The Java Object Cache bundled with the product may be used to implement this pattern.

This practice improves availability since the service locator can hide failures of the backend server or JNDI tree by having cached the lookup. Although this is only temporary since the results still have to be fetched.

Performance is also improved since trips to the back-end application server are reduced.

Cluster Your EJBs

Cluster your EJBs only when you require:

- Load Balancing: The EJB client or clients are load balanced across the servers in the EJB cluster.

- Fault Tolerance: The state (in case of stateful session beans) is replicated across the OC4J processes in the EJB cluster. If the proxy classes on the client cannot connect to an EJB server, they will attempt to connect to the next server in the cluster. The client does not see the failure.
- Scalability: Since multiple EJB servers behaving as one can service many more requests than a single EJB server, a clustered EJB system is more scalable. The alternative is to have stand-alone EJB systems, with manual partitioning of clients across servers. This is difficult to configure and does not have fault tolerance advantages.

In order to fully leverage EJB clustering you will need to use remote EJBs. Remote EJBs have some performance implications over local EJBs. If you use local EJBs and save a reference to them in a servlet (or JSP) session, when the session is replicated, the saved reference becomes invalid. Therefore, use EJB clustering only when you need the listed features.

Index Secondary Finder Methods

When finder methods, other than `findByPrimaryKey()` and `findAll()`, are created they may be extremely inefficient if appropriate indexes are not created that help to optimize execution of the SQL code generated by the container.

Understand the Lifecycle of EJBs

As a developer, it is imperative that you understand the lifecycle of EJBs. Many problems can be avoided by following the lifecycle and the expected actions during call backs more closely.

This is especially true with entity beans and stateful session beans. For example, in a small test environment, a bean may never get passivated, and thus a misimplementation (or nonimplementation) of `ejbPassivate()` and `ejbActivate()` may not show up until later. Moreover, since these are not used for stateless beans, they may confuse new developers.

Use Deferred Database Constraints

For those constraints that may be invalid for a short time during a transaction but will be valid at transaction boundaries, use deferred database constraints. For example, if a column is not populated during an `ejbCreate()`, but will be set prior to the completion of the transaction, then you may want to set the not null constraint for that column to be deferred. This also applies to foreign key constraints that are mirrored by EJB relationships with EJB 2.0.

Create a Cache with Read-Only EJBs

For those cases where data changes very slowly or not at all, and the changes are not made by your EJB application, read-only beans may make a very good cache. A good example of this is a country EJB. It is unlikely that it will change very often and it is likely that some degree of stale data is acceptable.

To do this:

1. Create read-only entity beans.
2. Set `exclusive-write-access="true"`.
3. Set the validity timeout to the maximum acceptable staleness of the data.

Pick an Appropriate Locking Strategy

It is critical that an appropriate locking strategy be combined with an appropriate database isolation mode for properly performing and highly reliable EJB applications.

Use optimistic locking where the likelihood of conflict in updates is low. If a lost update is acceptable or cannot occur because of application design, use an isolation mode of read-committed. If the lost updates are problematic, use an isolation mode of serializable.

Use pessimistic locking where there is a higher probability of update conflicts. Use an isolation mode of read-committed for maximum performance in this case. Use read-only locking when the data will not be modified by the EJB application.

Understand and Leverage Patterns

With the wider industry adoption, there are several common (and generally) acceptable ways of solving problems with EJBs. These have been widely published in either books or discussion forums, etc. In some sense, these patterns are best practices for a particular problem. These should be researched and followed.

Here are some examples:

- Session Façade: Combines multiple entity bean calls into a single call on a session bean, thus reducing the network traffic.
- Message Façade: Use MDBs if you do not need a return status from your method invocation.
- Value Object Pattern: A value object pattern reduces the network traffic by combining multiple data values that are usually required to be together, into a single value object.

A full discussion on the large number of patterns available is outside the scope of this document, but the references section contains some useful books, Web sites, or both on this subject.

When Using Entity Beans, Use Container-Managed Aged Persistence Whenever Possible

Although there are some limitations to container-managed persistence (CMP), CMP has a number of benefits. One benefit is portability. With CMP, decisions like persistence mapping and locking model selection become a deployment activity rather than a coding activity. This allows deployment of the same application in multiple containers with no change in code. This is commonly not true for Bean Managed Persistence (BMP) since SQL statements and concurrency control must be written into the entity bean and are therefore specific to the container and/or the data store.

Another benefit is that, in general, J2EE container vendors provide quality of service (QoS) features such as locking model variations, lazy loading, and performance and scalability enhancements, which may be controlled via deployment configuration rather than by writing code. Oracle Application Server includes features such as read-only entity beans, minimal writing of changes, and lazy loading of relations, which would have to be built into code for BMP.

A third benefit of CMP is container-managed relationships. Through declarations, not unlike CMP field mapping, a CMP entity bean can have relationships between two entity beans managed by the container with no implementation code required from application developers.

Last but least, tools are available to aid in the creation of CMP entity beans so that minimal work is required from developers for persistence. This allows developers to focus on business logic, which allows them to be more efficient. JDeveloper9i is a perfect example where, through modeling tools and wizards, very little work is required to create CMP entity beans including creation of both the generic EJB descriptors and the Oracle Application Server specific descriptors.

Overall, there are cases where CMP does not meet the requirements of an application, but the development effort saved, and the optimizations that J2EE containers like OC4J provide make CMP much more attractive than BMP.

Entity Beans Using Local interfaces Only

It is a good practice to expose your entity beans using only local interfaces because container managed relationship can only be used with local interfaces. Also local interfaces avoid expensive serialization and remote network calls.

Use a Session Bean Facade for Entity Beans

Avoid using entity beans directly from Web modules and client applications and use a session bean façade layer instead. Use of entity beans from client applications hardcodes the domain model in the client. It also introduces difficulty when managing both remote and local interfaces for entity beans.

Create a session bean façade layer by grouping together all natural use cases. This exposes operations to one or more entity beans. It provides finer grained access to the entity beans and reduces database interactions by acting as a transaction boundary. This also enables the entity beans to be accessed by Web services by exposing the stateless session bean as a Web service endpoint.

Enforce Primary Key Constraints at the Database Level

Enforce primary key constraint for the underlying table for your CMP entity beans instead of having the container execute an extra SQL statement to check for a duplicate primary key. You can switch this check by setting the `do-select-before-insert="false"` for your entity bean in the `orion-ejb-jar.xml` file.

Use a Foreign Key for 1-1 or 1-M Relationships

Use a foreign key when completing the O-R mapping for 1-1 or 1-many relationships between entity beans instead of using an association table. This enables you to avoid maintaining an extra table and an extra SQL statement generated by container to maintain the relationships.

Avoid the findAll() Method on Entities Based on Large Tables

When you use the `findAll()` method, the container tries to retrieve all rows of the table. You should try to avoid this type of operation on entities based on tables that have a large number of records. It will slow down the operations of your database.

Set prefetch-size to Reduce Round Trips to Database

Oracle JDBC drivers have extensions that allow setting the number of rows to prefetch into the client while a result set is being populated during a query. This reduces the number of round trips to the server. This can drastically improve performance of finder methods that return a large number of rows. You can specify the `prefetch-size` attribute for your finder method in the `orion-ejb-jar.xml` file.

Use Lazy Loading with Caution

If you turn on lazy loading, which is off by default, then only the primary keys of the objects retrieved within the finder are returned. Thus, when you access the object within your implementation, the OC4J container uploads the actual object based on the primary key.

You may want to turn on the lazy loading feature if the number of objects that you are retrieving is so large that loading them all into your local cache would decrease performance. If you retrieve multiple objects, but you only use a few of them, then you should turn on lazy loading. In addition, if you only use objects through the `getPrimaryKey` method, then you should also turn on lazy loading.

Avoid Performing O-R Mapping Manually

O-R mapping for CMP entity beans in the `orion-ejb-jar.xml` file is very complex and error prone. Any error in the mapping can cause deployment errors and generation of wrong SQL code for EJB-SQL statements. The following two approaches are recommended:

- Use JDeveloper 9.0.5.1 to perform the O-R mapping and to generate the mapping information in the `orion-ejb-jar.xml` file.
- Deploy the application in OC4J to generate the mappings and then modify the `orion-ejb-jar.xml` file to include the correct table name and persistence names.

A

OC4J-Specific Deployment Descriptors

This appendix provides an overview of OC4J-specific `orion-application.xml` and `orion-application-client.xml` deployment descriptor files. See the other OC4J developer guides for documentation of other OC4J-specific descriptors.

The following topics are included:

- Elements in the orion-application.xml File
- Elements in the orion-application-client.xml File

Elements in the orion-application.xml File

This section provides an overview of the OC4J-specific application deployment descriptor file.

<orion-application>

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

Attributes:

- `autocreate-tables`: Whether to automatically create database tables for CMP beans in this application. The default is `false`.
- `autodelete-tables`: Whether to automatically delete old database tables for CMP beans when redeploying in this application. The default is `false`.
- `default-data-source`: The default data source to use if other than the server default. This must point to a valid data source for this application, if specified.
- `deployment-version`: The version of OC4J that this JAR was deployed against, if it is not matching the current version then it will be redeployed. This is an internal server value; do not edit.
- `treat-zero-as-null`: Whether or not to treat read zero as null when representing a primary key. The default is `false`.

<argument>

An argument used when invoking the application client if starting it in-process; that is, if `auto-start="true"`. This element is specific to client applications.

Attribute:

- `value`: The value of the argument.

<arguments>

Contains one or more`<argument>` elements, each containing an argument used when invoking the application client if starting it in-process; that is, if `auto-start="true"`. This element is specific to client applications.

<client-module>

An application client module of the application. An application client is a GUI or console-based standalone client that interacts with the server.

Attributes:

- `auto-start`: Whether to automatically start the application in-process at OC4J server startup. The default is `false`. Note that if `true`, the `user` attribute must also be set to `true`.
- `deployment-time`: Indicates the time the client was deployed. Internal to OC4J; do not edit.
- `path`: The path- absolute or relative to the EAR file - to the application client.
- `user`: Set to `true` to run the client in-process. If `auto-start` is `true`, the attribute must also be set to `true`.

<cluster>

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

Clustering is typically enabled 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.

Subelements of `<cluster>`:

```
<property-config>
<flow-control-policy>
<replication-policy>
<protocol>
<synchronous-replication>
```

For descriptions of these subelements, see *Oracle Containers for J2EE Configuration and Administration Guide*.

Attributes:

- `enabled`: Whether clustering is enabled for the application. The default is `true`. Setting this value at the application level overrides the value inherited from the parent application, including the `default` application.
- `group-name`: The name to use when establishing the replication group channels. If not supplied, the application name as defined in `server.xml`, the OC4J server configuration file, is used by default, and new group channels are created for each enterprise application.

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

  This attribute is ignored if the `<database>` tag is included.
- `allow-collocation`: Whether to allow application state to be replicated to a group member (JVM) residing on the same host machine.

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

If multiple OC4J instances are instantiated on the same machine, different listener ports must be specified for each instance in the `default-web-site.xml`, `jms.xml`, and `rmi.xml` configuration files.

- `write-quota`: The number of other application group members (JVMs) to which the application state should be replicated. This attribute makes it possible to reduce overhead by limiting the number of JVMs to which state is written, similar to the islands concept used in previous OC4J releases.

  The default is `1` JVM.

  This attribute is ignored if the `<database>` tag is included.

- `cache-miss-delay`: The length of time, in milliseconds, to wait in-process for another group member to respond with a session if the session cannot be found locally. If the session cannot be found, the request will pause for the entire length of time specified.

  The default is `1000` milliseconds. In installations where heavy request loads are expected, this value should be increase, for example to `5000`. Setting this value higher also prevents the OC4J instance from creating a replica of session data within itself if `allow-colocation` is set to `true`.

  This attribute is ignored if the `<database>` tag is included.

<connectors>

Attribute:

- `path`: The name and path of the `oc4j-connectors.xml` file. If no `<connectors>` element is specified, then the default path is *ORACLE_HOME*/j2ee/*instance*/connectors/*rarName*./oc4j-connectors.xml.

<data-sources>

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

OC4J data sources exist in an XML file known as `data-sources.xml`. This file is installed in the /j2ee/*instance*/config/ directory with a default data source.

Attribute:

- `path`: The path to the `data-sources.xml` file. The path can be fixed or relative to the location of the `orion-application.xml` descriptor.

<description>

A string containing an optional short description of the application.

<ejb-module>

References an EJB JAR module within the application.

Attributes:

- `path`: The path (relative to the EAR or absolute) to the ejb-jar file.
- `remote`: Set to `true` to activate the EJB instances on this node or to look them up remotely from another server (remote or inside a cluster). The default is false.

<file>

A relative/absolute path to a log file.

Attribute:

- `path`: The path.

<group>

A group that this security-role mapping implies. That is, all members of the specified group are included in this role.

Attribute:

- `name`: The name of the group.

<javagroups-config>

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

Attributes:

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

<jazn>

Configures the Java Authentication and Authorization Service (JAAS) to use the XML-based configuration provider type. For a description of this element, see the description of the `<jazn>` element of the `jazn.xml` file in the *Oracle Containers for J2EE Security Guide*.

<jazn-web-app>

Defines the filter element of `JAZNUserManager`. For a description of this element, see the description of the `<jazn-web-app>` element of the `jazn.xml` file in the *Oracle Containers for J2EE Security Guide*.

<jmx-mbean>

Specifies a single MBean deployed with an application that is to be registered automatically with the OC4J MBeanServer.

Subelements:

- `<description>`: A string containing a readable name for the MBean. This name will be displayed in the MBean browser user interface.

Attributes:

- `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`.
- `class`: The MBean implementation class.

<library>

Specifies either a relative or an 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.

Attribute:

- `path`: The path.

<log>

Sets the logging configuration for the application.

Subelements:

```
<file>
<mail>
<odl>
```

<odl>

Configures Oracle Diagnostic Logging for 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 them to be more easily parsed and reused by other Oracle Application Server and custom developed components

- `maxDirectorySize`: Sets the maximum size, in bytes, allowed for the log file directory. When this limit is exceeded, log files are purged, beginning with the oldest files.
- `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`: Path and folder name of the log folder for this component. You can use an absolute path or a path relative to where the configuration XML file exists, which is normally in the `/j2ee/`*`instance`*`/config` directory. This denotes where the log files will reside for the feature that the XML configuration file is concerned with.

When you enable ODL logging, each message goes into its respective log file, named `log`*`N`*`.xml`, where N is a number starting at one. The first log message starts the log file, `log1.xml`. When the log file size maximum is reached, the second log file is opened to continue the logging, `log2.xml`. When the last log file is full, the first log file, `log1.xml`, is erased and a new file is opened for the new messages. Thus, your log files are constantly rolling over and do not encroach on your disk space.

Attributes:

- `path`: Path and folder name of the log folder for this area. You can use an absolute path or a path relative to where the configuration XML file exists, which is normally in the `/j2ee/`*`instance`*`/config` directory. This denotes where the log files will reside for the feature that the XML configuration file is concerned with. For example, modifying this element in the `server.xml` file denotes where the server log files are written.
- `max-file-size`: The maximum size in KB of each individual log file.
- `max-directory-size`: The maximum size of the directory in KB. The default directory size is 10 MB.

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

<mail>

An e-mail address to log events to. A valid mail-session also needs to be specified if this option is used.

Attribute:

- `address`: The mail-address.

<mail-session>

Defines the session SMTP server host (if using SMTP).

Attributes:

- `location`: The location in the namespace to store the session at.
- `smtp-host`: The session SMTP-server host (if using SMTP).

<namespace-access>

Specifies the namespace (naming context) security policy for RMI clients.

<namespace-resource>

Defines a resource with a specific security setting.

Attribute:

- `root`: The root of the part of the namespace that this rule applies to.

<password-manager>

Specifies the `UserManager` that is used for the lookup of hidden passwords. If omitted, the current `UserManager` is used for authentication and authorization. For example, you can use a JAZN LDAP `UserManager` for the overall `UserManager`, but use a JAZN XML `UserManager` for checking hidden passwords.

To identify a `UserManager`, provide a `<jazn>` element definition within this element, as follows:

```
<password-manager>
   <jazn ...>
</password-manager>
```

For a description of the `<jazn>` element, see the description of the `<jazn>` element of the `jazn.xml` file in the *Oracle Containers for J2EE Security Guide*

<persistence>

Contains a relative (to the application root) or absolute path to a directory where application state should be stored across restarts.

Attribute:

- `path`: The path - relative to the EAR file or absolute - to the persistence directory. For example, `./persistence`.

<principals>

Defines the path to the `principals.xml` file.

Attribute:

- `path`: The path (relative to the enterprise archive or absolute) to the principals file.

<property>

Contains a property as a name and value pair.

Attributes:

- `name`: The name of the parameter.
- `value`: The value of the parameter.

<protocol>

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

Subelements:

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

<read-access>

The read-access policy.

<resource-provider>

Define a JMS resource provider. To add a custom `<resource-provider>`, add the following to your orion-application.xml file:

```
<resource-provider class="providerClassName" name="JNDI name">
  <description> description </description>
  <property name="name" value="value" />
</resource-provider>
```

In place of the user-replaceable constructs (those in italics) in the preceding example, do the following:

- Replace the value *`providerClassName`* of the `class` attribute with the name of the resource-provider class.
- Replace the value *`JNDI name`* of the `name` attribute with a name by which to identify the resource provider. This name will be used in finding the resource provider in the application's JNDI as "`java:comp/resource/name/`".
- Replace the value *`description`* of the `description` tag with a description of the specific resource provider.
- Replace the values *`name`* and *`value`* of the corresponding attributes with the same name in any property tags that the specific resource provider needs to be given as parameters.

<security-role-mapping>

Defines the runtime mapping to groups and users of a role. Maps to a security ole of the same name in the assembly descriptor.

Attributes:

- `impliesAll`: Whether or not this mapping implies all users.
- `name`: The name of the role

<user>

Defines a user that the security-role mapping implies.

Attribute:

- `name`: The name of the user.

<user-manager>

Specifies an optional user-manager class to use. These are used to integrate existing systems and provide custom user-managers for Web applications.

Attributes:

- `class`: The fully qualified classname of the user-manager; for example `com.evermind.sql.DataSourceUserManager` or `com.evermind.ejb.EJBUserManager`.
- `display-name`: A descriptive name for the `UserManager` instance.

<web-module>

Identifies a Web application/module that is part of the application. Each Web application can be installed on any site and in any context on those sites (for instance `http://www.myserver.com/myapp/`).

Attributes:

- `id`: The name used to reference this Web application, for example when binding the module to a Web site
- `path`: The path - relative to the EAR or absolute - to the Web application.

<write-access>

The write-access policy

Elements in the orion-application-client.xml File

This file is the OC4J-specific descriptor for an application client.

<orion-application-client>

Defines an `orion-application-client.xml` file containing the deploy time information for a J2EE application client. It complements the application client assembly information found in `application-client.xml.`

<context-attribute>

Contains an attribute sent to the context. The only mandatory attribute in JNDI is the '`java.naming.factory.initial`,' which is the classname of the context factory implementation.

Attributes:

- `name`: The name of the attribute.
- `value`: The value of the attribute.

<ejb-ref-mapping>

Used for the declaration of a reference to another enterprise bean's home. The `ejb-ref-mapping` element ties this to a JNDI-location when deploying.

Attributes:

- `location`: The JNDI location to look up the EJB home from, such as `ejb/Payroll`.
- `name`: The `ejb-ref` name. Matches the name defined in an `<ejb-ref>` in `application-client.xml`.

<env-entry-mapping>

Overrides the value of an `env-entry` in the assembly descriptor. It is used to keep the EAR (assembly) clean from deployment-specific values. The body is the value.

Attributes:

- `name`: The name of the context parameter.

<lookup-context>

Specifies an optional `javax.naming.Context` implementation used for retrieving the resource. This is useful when hooking up with third party modules, such as a third party JMS server for instance. Either use the context implementation supplied by the resource vendor or if none exists write an implementation which in turn negotiates with the vendor software.

Attributes:

- `location`: The name looked for in the foreign context when retrieving the resource.

<resource-env-ref-mapping>

Declares a reference to an external resource, such as a data source, JMS queue, mail session, or similar. The `resource-env-ref-mapping` ties that element to a JNDI location during deployment.

Attributes:

- `location`: The JNDI location to bind the resource to.

<resource-ref-mapping>

Declares a reference to an external resource such as a data source, JMS queue, mail session or similar. The `resource-ref-mapping` ties this to a JNDI-location when deploying.

Attributes:

- `location`: The JNDI location to look up the resource home from.
- `name`: The `resource-ref` name. Matches the name of a `resource-ref` in `application-client.xml`

B

Third Party Licenses

This appendix includes the Third Party License for all the third party products included with Oracle Application Server.

ANTLR

This program contains third-party code from ANTLR. Under the terms of the Apache license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the ANTLR software, and the terms contained in the following notices do not change those rights.

The ANTLR License

```
Software License

We reserve no legal rights to the ANTLR--it is fully in the public domain. An
individual or company may do whatever they wish with source code distributed with
ANTLR or the code generated by ANTLR, including the incorporation of ANTLR, or its
output, into commerical software.
We encourage users to develop software with ANTLR. However, we do ask that credit
is given to us for developing ANTLR. By "credit", we mean that if you use ANTLR or
incorporate any source code into one of your programs (commercial product,
research project, or otherwise) that you acknowledge this fact somewhere in the
documentation, research report, etc... If you like ANTLR and have developed a nice
tool with the output, please mention that you developed it using ANTLR. In
addition, we ask that the headers remain intact in our source code. As long as
these guidelines are kept, we expect to continue enhancing this system and expect
to make other tools available as they are completed.
```

Apache

This program contains third-party code from the Apache Software Foundation ("Apache"). Under the terms of the Apache license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Apache software, and the terms contained in the following notices do not change those rights.

The Apache license agreements apply to the following included Apache components:

- Apache HTTP Server
- Apache JServ
- mod_jserv

- Regular Expression package version 1.3
- Apache Expression Language packaged in commons-el.jar
- mod_mm 1.1.3
- Apache XML Signature and Apache XML Encryption v. 1.4 for Java and 1.0 for C++
- log4j 1.1.1
- BCEL v. 5
- XML-RPC v. 1.1
- Batik v. 1.5.1
- ANT 1.6.2 and 1.6.5
- Crimson v. 1.1.3
- ant.jar
- wsif.jar
- bcel.jar
- soap.jar
- Jakarta CLI 1.0
- jakarta-regexp-1.3.jar
- JSP Standard Tag Library 1.0.6 and 1.1
- Struts 1.1
- Velocity 1.3
- svnClientAdapter
- commons-logging.jar
- wsif.jar
- commons-el.jar
- standard.jar
- jstl.jar

The Apache Software License

License for Apache Web Server 1.3.29

```
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. The end-user documentation included with the redistribution,
 *    if any, must include the following acknowledgment:
 *       "This product includes software developed by the
 *        Apache Software Foundation (http://www.apache.org/)."
 *    Alternately, this acknowledgment may appear in the software itself,
 *    if and wherever such third-party acknowledgments normally appear.
 *
 * 4. The names "Apache" and "Apache Software Foundation" must
 *    not be used to endorse or promote products derived from this
 *    software without prior written permission. For written
 *    permission, please contact apache@apache.org.
 *
 * 5. Products derived from this software may not be called "Apache",
 *    nor may "Apache" appear in their name, without prior written
 *    permission of the Apache Software Foundation.
 *
 * THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 * ====================================================================
 *
 * This software consists of voluntary contributions made by many
 * individuals on behalf of the Apache Software Foundation.  For more
 * information on the Apache Software Foundation, please see
 * <http://www.apache.org/>.
 *
 * Portions of this software are based upon public domain software
 * originally written at the National Center for Supercomputing
Applications,
 * University of Illinois, Urbana-Champaign.
```

License for Apache Web Server 2.0

```
Copyright (c) 1999-2004, The Apache Software Foundation
Licensed under the Apache License, Version 2.0 (the "License"); you may not use
this file except in compliance with the License.  You may obtain a copy of the
License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied.  See the License for the
specific language governing permissions and limitations under the License.
Copyright (c) 1999-2004, The Apache Software Foundation
                                Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/
```

```
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.
```

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions

```
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.
```

Apache SOAP

This program contains third-party code from the Apache Software Foundation ("Apache"). Under the terms of the Apache license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Apache software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the Apache software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Apache.

Apache SOAP License

Apache SOAP license 2.3.1

```
Copyright (c) 1999 The Apache Software Foundation.  All rights reserved.
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
```

Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

DBI Module

This program contains third-party code from Tim Bunce. Under the terms of the Tim Bunce license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Tim Bunce software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the Tim Bunce software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Tim Bunce.

You may distribute under the terms of either the GNU General Public License or the Artistic License, as specified in the Perl README file.

Perl Artistic License

The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

Definitions

"Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.

"Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or copyrights for the package.

"You" is you, if you're thinking about copying or distributing this Package.

"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.
2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.
3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a. place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b. use the modified Package only within your corporation or organization.

c. rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d. make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

a. distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

b. accompany the distribution with the machine-readable source of the Package with your modifications.

c. give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

d. make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package's interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whoever generated them, and may be sold commercially, and may be aggregated with this Package. If such scripts or library files are aggregated with this Package through the so-called "undump" or "unexec" methods of producing a binary executable image, then distribution of such an image shall neither be construed as a distribution of this Package nor shall it fall under the restrictions of Paragraphs 3 and 4, provided that you do not represent such an executable image as a Standard Version of this Package.

7. C subroutines (or comparably compiled subroutines in other languages) supplied by you and linked into this Package in order to emulate subroutines and variables of the language defined by this Package shall not be considered part of this Package, but are the equivalent of input as in Paragraph 6, provided these subroutines do not change the language in any way that would cause it to fail the regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always permitted provided that the use of this Package is embedded; that is, when no overt attempt

is made to make this Package's interfaces visible to the end user of the commercial distribution. Such use shall not be construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.
10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

FastCGI

This program contains third-party code from Open Market, Inc. Under the terms of the Open Market license, Oracle is required to license the Open Market software to you under the following terms. Note that the terms contained in the Oracle program license that accompanied this product do not apply to the Open Market software, and your rights to use the software are solely as set forth below. Oracle is not responsible for the performance of the Open Market software, does not provide technical support for the software, and shall not be liable for any damages arising out of any use of the software.

FastCGI Developer's Kit License

This FastCGI application library source and object code (the "Software") and its documentation (the "Documentation") are copyrighted by Open Market, Inc ("Open Market"). The following terms apply to all files associated with the Software and Documentation unless explicitly disclaimed in individual files.

Open Market permits you to use, copy, modify, distribute, and license this Software and the Documentation solely for the purpose of implementing the FastCGI specification defined by Open Market or derivative specifications publicly endorsed by Open Market and promulgated by an open standards organization and for no other purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions.

No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this Software and Documentation may be copyrighted by their authors and need not follow the licensing terms described here, but the modified Software and Documentation must be used for the sole purpose of implementing the FastCGI specification defined by Open Market or derivative specifications publicly endorsed by Open Market and promulgated by an open standards organization and for no other purpose. If modifications to this Software and Documentation have new licensing terms, the new terms must protect Open Market's proprietary rights in the Software and Documentation to the same extent as these licensing terms and must be clearly indicated on the first page of each file where they apply.

Open Market shall retain all right, title and interest in and to the Software and Documentation, including without limitation all patent, copyright, trade secret and other proprietary rights.

OPEN MARKET MAKES NO EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE SOFTWARE OR THE DOCUMENTATION, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL OPEN MARKET BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY DAMAGES ARISING FROM OR RELATING

TO THIS SOFTWARE OR THE DOCUMENTATION, INCLUDING, WITHOUT LIMITATION, ANY INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES OR SIMILAR DAMAGES, INCLUDING LOST PROFITS OR LOST DATA, EVEN IF OPEN MARKET HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS". OPEN MARKET HAS NO LIABILITY IN CONTRACT, TORT, NEGLIGENCE OR OTHERWISE ARISING OUT OF THIS SOFTWARE OR THE DOCUMENTATION.

Module mod_fastcgi License

This FastCGI application library source and object code (the "Software") and its documentation (the "Documentation") are copyrighted by Open Market, Inc ("Open Market"). The following terms apply to all files associated with the Software and Documentation unless explicitly disclaimed in individual files.

Open Market permits you to use, copy, modify, distribute, and license this Software and the Documentation solely for the purpose of implementing the FastCGI specification defined by Open Market or derivative specifications publicly endorsed by Open Market and promulgated by an open standards organization and for no other purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions.

No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this Software and Documentation may be copyrighted by their authors and need not follow the licensing terms described here, but the modified Software and Documentation must be used for the sole purpose of implementing the FastCGI specification defined by Open Market or derivative specifications publicly endorsed by Open Market and promulgated by an open standards organization and for no other purpose. If modifications to this Software and Documentation have new licensing terms, the new terms must protect Open Market's proprietary rights in the Software and Documentation to the same extent as these licensing terms and must be clearly indicated on the first page of each file where they apply.

Open Market shall retain all right, title and interest in and to the Software and Documentation, including without limitation all patent, copyright, trade secret and other proprietary rights.

OPEN MARKET MAKES NO EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE SOFTWARE OR THE DOCUMENTATION, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL OPEN MARKET BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY DAMAGES ARISING FROM OR RELATING TO THIS SOFTWARE OR THE DOCUMENTATION, INCLUDING, WITHOUT LIMITATION, ANY INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES OR SIMILAR DAMAGES, INCLUDING LOST PROFITS OR LOST DATA, EVEN IF OPEN MARKET HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS". OPEN MARKET HAS NO LIABILITY IN CONTRACT, TORT, NEGLIGENCE OR OTHERWISE ARISING OUT OF THIS SOFTWARE OR THE DOCUMENTATION.

Info-ZIP Unzip Package

This program contains third-party code from Info-ZIP. Under the terms of the Info-ZIP license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Info-ZIP software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the Info-ZIP software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Info-ZIP.

The Info-ZIP Unzip Package License

JSR 110

This program contains third-party code from IBM Corporation ("IBM"). Under the terms of the IBM license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the IBM software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the IBM software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or IBM.

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 in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the

Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept

this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask

```
for permission. For software which is copyrighted by the Free Software Foundation,
write to the Free Software Foundation; we sometimes make exceptions for this. Our
decision will be guided by the two goals of preserving the free status of all
derivatives of our free software and of promoting the sharing and reuse of
software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE
LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED
IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS"
WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH
YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY
SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY
COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE
LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL,
SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY
TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF
THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER
PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS
How to Apply These Terms to Your New Libraries
If you develop a new library, and you want it to be of the greatest possible use
to the public, we recommend making it free software that everyone can redistribute
and change. You can do so by permitting redistribution under these terms (or,
alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to
attach them to the start of each source file to most effectively convey the
exclusion of warranty; and each file should have at least the "copyright" line and
a pointer to where the full notice is found.

<one line to give the library's name and an idea of what it does.> Copyright (C)
<year> <name of author>

This library is free software; you can redistribute it and/or modify it under the
terms of the GNU Lesser General Public License as published by the Free Software
Foundation; either version 2.1 of the License, or (at your option) any later
version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along
with this library; if not, write to the Free Software Foundation, Inc., 59 Temple
Place, Suite 330, Boston, MA 02111-1307 USA
```

mod_mm and mod_ssl

This program contains third-party code from Ralf S. Engelschall ("Engelschall"). Under the terms of the Engelschall license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the Engelschall software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the mod_mm software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or Engelschall.

mod_mm

```
Copyright (c) 1999 - 2000 Ralf S. Engelschall.  All rights reserved.
This product includes software developed by Ralf S. Engelschall
<rse@engelschall.com> for use in the mod_ssl project (http://www.modssl.org/).
```

mod_ssl

```
Copyright (c) 1998-2001 Ralf S. Engelschall.  All rights reserved.
This product includes software developed by Ralf S. Engelschall
<rse@engelschall.com> for use in the mod_ssl project (http://www.modssl.org/).
```

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

```
/* ====================================================================
 * Copyright (c) 1998-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
```

```
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

 Original SSLeay License
 -----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
```

```
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
```

Perl

This program contains third-party code from the Comprehensive Perl Archive Network ("CPAN"). Under the terms of the CPAN license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the CPAN software, and the terms contained in the following notices do not change those rights.

Perl Kit Readme

Copyright 1989-2001, Larry Wall

All rights reserved.

This program is free software; you can redistribute it and/or modify it under the terms of either:

1. the GNU General Public License as published by the Free Software Foundation; either version 1, or (at your option) any later version, or
2. the "Artistic License" which comes with this Kit.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See either the GNU General Public License or the Artistic License for more details.

You should have received a copy of the Artistic License with this Kit, in the file named "Artistic". If not, I'll be glad to provide one.

You should also have received a copy of the GNU General Public License along with this program in the file named "Copying". If not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA or visit their Web page on the internet at http://www.gnu.org/copyleft/gpl.html.

For those of you that choose to use the GNU General Public License, my interpretation of the GNU General Public License is that no Perl script falls under the terms of the GPL unless you explicitly put said script under the terms of the GPL yourself. Furthermore, any object code linked with perl does not automatically fall under the terms of the GPL, provided such object code only adds definitions of subroutines and variables, and does not otherwise impair the resulting interpreter from executing any standard Perl script. I consider linking in C subroutines in this manner to be the moral equivalent of defining subroutines in the Perl language itself. You may sell such an object file as proprietary provided that you provide or offer to provide the Perl source, as specified by the GNU General Public License. (This is merely an alternate way of specifying input to the program.) You may also sell a binary produced by the dumping of a running Perl script that belongs to you, provided that you provide or offer to provide the Perl source as specified by the GPL. (The fact that a Perl interpreter and your code are in the same binary file is, in this case, a form of mere aggregation.) This is my interpretation of the GPL. If you still have concerns or difficulties understanding my intent, feel free to contact me. Of course, the Artistic License spells all this out for your protection, so you may prefer to use that.

mod_perl 1.29 License

```
/* ====================================================================
 * The Apache Software License, Version 1.1
 *
 * Copyright (c) 1996-2000 The Apache Software Foundation.  All rights
 * reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. The end-user documentation included with the redistribution,
 *    if any, must include the following acknowledgment:
 *       "This product includes software developed by the
 *        Apache Software Foundation (http://www.apache.org/)."
 *    Alternately, this acknowledgment may appear in the software itself,
 *    if and wherever such third-party acknowledgments normally appear.
 *
 * 4. The names "Apache" and "Apache Software Foundation" must
 *    not be used to endorse or promote products derived from this
 *    software without prior written permission. For written
 *    permission, please contact apache@apache.org.
 *
 * 5. Products derived from this software may not be called "Apache",
 *    nor may "Apache" appear in their name, without prior written
 *    permission of the Apache Software Foundation.
 *
 * THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
```

```
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 * ====================================================================
 */
```

mod_perl 1.99_16 License

```
Copyright (c) 1999-2004, The Apache Software Foundation
Licensed under the Apache License, Version 2.0 (the "License"); you may not use
this file except in compliance with the License.  You may obtain a copy of the
License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied.  See the License for the
specific language governing permissions and limitations under the License.
Copyright (c) 1999-2004, The Apache Software Foundation
                                Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).
```

```
   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and
```

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a

```
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.
```

Perl Artistic License

The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

Definitions

"Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.

"Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or copyrights for the package.

"You" is you, if you're thinking about copying or distributing this Package.

"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.
2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.
3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a. place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b. use the modified Package only within your corporation or organization.

c. rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d. make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

   a. distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

   b. accompany the distribution with the machine-readable source of the Package with your modifications.

   c. give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

   d. make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package's interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whoever generated them, and may be sold commercially, and may be aggregated with this Package. If such scripts or library files are aggregated with this Package through the so-called "undump" or "unexec" methods of producing a binary executable image, then distribution of such an image shall neither be construed as a distribution of this Package nor shall it fall under the restrictions of Paragraphs 3 and 4, provided that you do not represent such an executable image as a Standard Version of this Package.

7. C subroutines (or comparably compiled subroutines in other languages) supplied by you and linked into this Package in order to emulate subroutines and variables of the language defined by this Package shall not be considered part of this Package, but are the equivalent of input as in Paragraph 6, provided these subroutines do not change the language in any way that would cause it to fail the regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always permitted provided that the use of this Package is embedded; that is, when no overt attempt

is made to make this Package's interfaces visible to the end user of the commercial distribution. Such use shall not be construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.
10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

SAXPath

This program contains third-party code from SAXPath. Under the terms of the SAXPath license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the SAXPath software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the SAXPath software is provided by Oracle "AS IS" and without warranty or support of any kind from Oracle or SAXPath.

The SAXPath License

```
Copyright (C) 2000-2002 werken digital. All rights reserved. Redistribution and
use in source and binary forms, with or without modification, are permitted
provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list
of conditions, and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this
list of conditions, and the disclaimer that follows these conditions in the
documentation and/or other materials provided with the distribution.

The name "SAXPath" must not be used to endorse or promote products derived from
this software without prior written permission. For written permission, please
contact license@saxpath.org.

Products derived from this software may not be called "SAXPath", nor may "SAXPath"
appear in their name, without prior written permission from the SAXPath Project
Management (pm@saxpath.org).

In addition, we request (but do not require) that you include in the end-user
documentation provided with the redistribution and/or in the software itself an
acknowledgment equivalent to the following: "This product includes software
developed by the SAXPath Project (http://www.saxpath.org/)." Alternatively, the
acknowledgment may be graphical using the logos available at
http://www.saxpath.org/.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE SAXPath
AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
```

```
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made
by many individuals on behalf of the SAXPath Project and was originally created by
bob mcwhirter and James Strachan . For more information on the SAXPath Project,
please see http://www.saxpath.org/.
```

W3C DOM

This program contains third-party code from the World Wide Web Consortium ("W3C"). Under the terms of the W3C license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the W3C software, and the terms contained in the following notices do not change those rights. Notwithstanding anything to the contrary in the Oracle program license, the W3C software is provided by Oracle AS IS and without warranty or support of any kind from Oracle or W3C.

The W3C License

```
W3C® SOFTWARE NOTICE AND LICENSE
http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
This work (and included software, documentation such as READMEs, or other related
items) is being provided by the copyright holders under the following license. By
obtaining, using and/or copying this work, you (the licensee) agree that you have
read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation,
with or without modification, for any purpose and without fee or royalty is hereby
granted, provided that you include the following on ALL copies of the software and
documentation or portions thereof, including modifications:

The full text of this NOTICE in a location viewable to users of the redistributed
or derivative work.
Any pre-existing intellectual property disclaimers, notices, or terms and
conditions. If none exist, the W3C Software Short Notice should be included
(hypertext is preferred, text is permitted) within the body of any redistributed
or derivative code.
Notice of any changes or modifications to the files, including the date changes
were made. (We recommend you provide URIs to the location from which the code is
derived.)
THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO
REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO,
WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE
USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS,
COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or
publicity pertaining to the software without specific, written prior permission.
Title to copyright in this software and any associated documentation will at all
times remain with copyright holders.
```

Index

Symbols

A

B

C

D

E