Oracle9*i* Application Server

Migrating from Oracle9*i*AS Release 1 (1.0.2.2.x) to Release 2 (9.0.2)

Release 2 (9.0.2) for Windows NT/2000

May 2002

Part No. A96157-02

ORACLE

Oracle9*i* Application Server Migrating from Oracle9*i*AS Release 1 (1.0.2.2.x) to Release 2 (9.0.2), Release 2 (9.0.2) for Windows NT/2000

Part No. A96157-02

Contributors: Haranadh Abburu, Gina Abeleles, Kamalendu Biswas, Chris Broadbent, Chung-Ho Chen, Will Chin, David Clay, Michele Cyran, Saheli Dey, Greg Cook, Joe Garcia, Mark Gizejewski, Cathy Godwin, Binod Gupta, Robert Hipps, Marilyn Hollinger, Pavana Jain, Clara Jaeckel, Pushkar Kapasi, Ashish Kolli, Eric Lee, Jeremy Lizt, Mark Loper, Stephen Lee, Xiaohua Lu, Leslie Marder, Duncan Mills, Oscar Naim, Probal Nandy, Raymond Ng, Frank Nimphius, Andy Page, Saurabh Pandey, Julia Pond, Harish Rawat, Frank Rovitto, Mike Rubino, Charlie Shapiro, Jimmy Shi, Preeti Somal, Baogang Song, Margaret Taft, Todd Vender, Brian Wright, Paul Wright, Liujin Yu, Naveen Zalpuri.

Contents

3 Migrating Internet Applications Components

4 Migrating Portals Components

5 Migrating Wireless Components

6 Migrating Business Intelligence Components

7 Migrating Management Components

Send Us Your Comments

Oracle9*i* Application Server Migrating from Oracle9*i*AS Release 1 (1.0.2.2.x) to Release 2 (9.0.2), Release 2 (9.0.2) for Windows NT/2000

Part No. A96157-02

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

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

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

- Electronic mail: iasdocs_us@oracle.com
- FAX: 650-506-7407 Attn: Oracle9*i* Application Server Documentation Manager
- Postal service:
  Oracle Corporation
  Oracle9*i* Application Server Documentation
  500 Oracle Parkway, M/S 2op3
  Redwood Shores, CA 94065
  USA

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

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

Preface

This guide describes how to migrate from Oracle9*i*AS Release 1 (v1.0.2.2) to Oracle9*i*AS Release 2 (v9.0.2).

This preface contains these topics:

Audience

Migrating from Oracle9*i*AS Release 1 (1.0.2.2.x) to Release 2 (9.0.2) is intended for application server administrators and managers of databases used by application servers.

Organization

This book is organized by Oracle9*i*AS solution areas, and contains the following chapters:

Chapter 1, "Overview of Oracle9iAS Migration"

Describes the migration scope, components in the previous and current releases, and migration approaches and terminology.

Chapter 2, "Using the Oracle9iAS Migration Assistant"

Explains how to use the Oracle9*i*AS Migration Tool to migrate the Oracle HTTP Server, Oracle9*i*AS Containers for J2EE, and Oracle9*i*AS WebCache.

Chapter 3, "Migrating Internet Applications Components"

Explains how to migrate JSP and JServ applications to Oracle9*i*AS Containers for J2EE, Oracle9*i*AS SOAP, Oracle8*i* PLSQL.

Chapter 4, "Migrating Portals Components"

Explains how to migrate Oracle9*i*AS Portal and UltraSearch.

Chapter 5, "Migrating Wireless Components"

Explains how to migrate Oracle9*i*AS Wireless.

Chapter 6, "Migrating Business Intelligence Components"

Explains how to migrate Oracle9*i*AS Forms Services, Oracle9*i*AS Reports Services, and Oracle9*i*AS Discoverer.

Chapter 7, "Migrating Management Components"

Explains how to migrate Oracle Enterprise Manager, Oracle9*i*AS Single Sign-On, and Oracle Internet Directory.

Chapter 8, "Migrating e-Business Integration Components"

Explains how to migrate Oracle9*i*AS InterConnect.

Related Documentation

In North America, printed documentation is available for sale in the Oracle Store at

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

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

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

`http://technet.oracle.com/membership/index.htm`

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

`http://technet.oracle.com/docs/index.htm`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width font)`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.

Convention	Meaning	Example
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
lowercase monospace (fixed-width font) italic	Lowercase monospace italic font represents placeholders or variables.	You can specify the *parallel_clause*. Run `U`*old_release*`.SQL` where *old_release* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*digits* `[ ,` *precision* `])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`

Convention	Meaning	Example
. . .	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS subquery;` `SELECT col1, col2, ... , coln FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/system_password` `DB_NAME = database_name`

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other

market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

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

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

1

Overview of Oracle9*i*AS Migration

This chapter describes the Oracle9*i*AS migration scope, methods and process. It contains these major sections:

Migration Scope and Methods

The information in this migration guide applies only to migrating Oracle9*i*AS Release 1 (v1.0.2.2) to Release 2 (v9.0.2).

If you have a Release 1 (1.0.2.1.x) installation, you must first migrate to Oracle9*i*AS Release 1 (1.0.2.2.x).

> **See Also:** *Oracle9i Application Server Migration Guide* in the Oracle9*i* Application Server documentation library

Identifying Source and Target Oracle9*i*AS Installations

This guide supports migration from one Oracle home to another, where Oracle9*i*AS Release 1 (1.0.2.2.x) is installed in a different Oracle home than Oracle9*i*AS Release 2 (9.0.2). The Oracle homes are designated as follows:

The location of Release 1 (v1.0.2.2), is called *ORACLE_HOME_1*, or the source Oracle home.

The location of Release 2 (v9.0.2) is called *ORACLE_HOME_2*, or the target Oracle home.

Migration Paths

This section compares Release 1 and Release 2 architectures, components, and installation types, and provides recommendations on selecting the Release 2 installation type to which you should migrate. Also included is information about database and inter-component dependencies, and how to proceed with migration in light of these. This section contains the following topics:

Components and Installation Types

Table 1–1 shows Oracle9*i*AS components and the installation types to which they belong for the previous and current releases. The components you plan to migrate must belong to the install type that you chose when installing Release 2.

Table 1–1 Oracle9iAS Components and Installation Types

Component	Oracle9*i*AS Release 1 (1.0.2.2.x) Install Types	Oracle9*i*AS Release 2 (9.0.2) Install Types
Oracle HTTP Server	Core, Minimal, Standard, Enterprise	■ J2EE and Web Cache ■ Portal and Wireless ■ Business Intelligence and Forms ■ Unified Messaging
Oracle9*i*AS Containers for J2EE	Core	■ J2EE and Web Cache ■ Portal and Wireless ■ Business Intelligence and Forms ■ Unified Messaging
Oracle Business Components for Java	Core, Minimal, Standard, Enterprise	■ J2EE and Web Cache
Oracle9*i*AS Forms Services	Enterprise	■ Business Intelligence and Forms ■ Unified Messaging

Table 1–1 Oracle9iAS Components and Installation Types

Component	Oracle9*i*AS Release 1 (1.0.2.2.x) Install Types	Oracle9*i*AS Release 2 (9.0.2) Install Types
Oracle9*i*AS Portal	Minimal, Standard, Enterprise	■ Portal and Wireless ■ Business Intelligence and Forms ■ Unified Messaging
Oracle9*i*AS Wireless	Minimal, Standard, Enterprise	■ Portal and Wireless ■ Business Intelligence and Forms ■ Unified Messaging
Oracle9*i*AS Web Cache	Core, Enterprise	■ J2EE and Web Cache ■ Portal and Wireless ■ Business Intelligence and Forms ■ Unified Messaging
Oracle9*i*AS Reports Services	Enterprise	■ Business Intelligence and Forms ■ Unified Messaging
Oracle9*i*AS Discoverer	Enterprise	■ Business Intelligence and Forms ■ Unified Messaging
Oracle9*i*AS Personalization	N / A	■ Business Intelligence and Forms ■ Unified Messaging
Oracle Enterprise Manager Web Site	N / A	■ J2EE and WebCache ■ Portal and Wireless ■ Business Intelligence and Forms ■ Unified Messaging

Table 1–1 Oracle9iAS Components and Installation Types

Component	Oracle9*i*AS Release 1 (1.0.2.2.x) Install Types	Oracle9*i*AS Release 2 (9.0.2) Install Types
Oracle Enterprise Manager (Console and Management Server)	Enterprise	Infrastructure
Oracle Internet File System	Standard, Enterprise	See the *Oracle9iFS Installation and Configuration Guide* for migration instructions. Oracle Internet File System is provided on a separate CD from Oracle9*i*AS.
Oracle9*i*AS Unified Messaging	N / A	■ Unified Messaging

Oracle9*i*AS Release 1 and Release 2 Installation Types

To migrate from Release 1 to Release 2, it is helpful to understand the relationship between the architectures and installation types. This section depicts and describes the relationship in detail.

Figure 1–1 Oracle9iAS Release 1 (1.0.2.2.x) Architecture

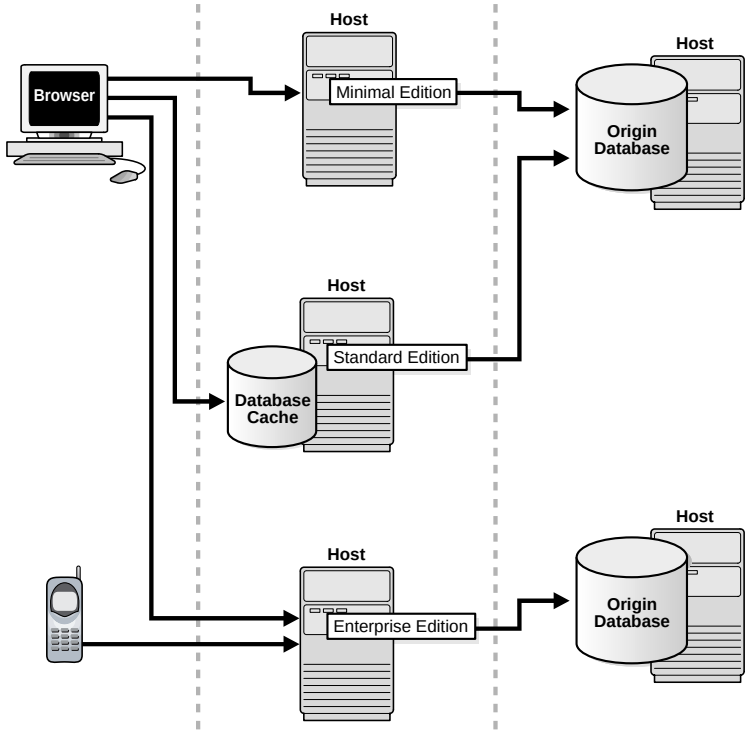


Figure 1–2 Oracle9iAS Release 2 (9.0.2) Architecture

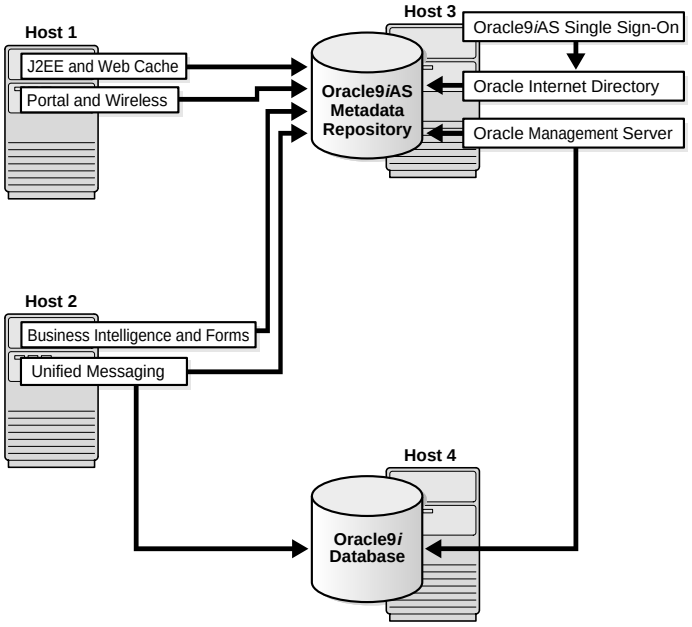


Table 1–2 Oracle9iAS Release 1 and Release 2 Installation Types

Release 1 Installation Type	Release 1 Component Configured	Recommended Release 2 Install Type	Recommended Release 2 Component Configuration
Core Minimal Edition Standard Edition Enterprise Edition	Oracle HTTP Server	J2EE and Web Cache	Oracle HTTP Server Oracle9*i*AS Containers for J2EE
Core Enterprise Edition	Oracle HTTP Server	J2EE and Web Cache	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache
Core	Oracle9*i*AS Containers for J2EE	J2EE and Web Cache	Oracle HTTP Server Oracle9*i*AS Containers for J2EE

Table 1–2 Oracle9iAS Release 1 and Release 2 Installation Types

Release 1 Installation Type	Release 1 Component Configured	Recommended Release 2 Install Type	Recommended Release 2 Component Configuration
Core Enterprise Edition	Oracle9*i*AS Web Cache	J2EE and Web Cache	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache
Minimal Edition Standard Edition Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Portal	Portal and Wireless	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Portal
Minimal Edition Standard Edition Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Wireless	Portal and Wireless	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Portal
Minimal Edition Standard Edition Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Portal Oracle9*i*AS Wireless	Portal and Wireless	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Portal Oracle9*i*AS Wireless
Standard Edition Enterprise Edition	Oracle HTTP Server Oracle Enterprise Java Engine	J2EE and Web Cache	Oracle HTTP Server Oracle9*i*AS Containers for J2EE See the statement of direction for the Java platform at http://otn.oracle.com/tech/java/oc4j/htdocs/oc4j_sod.html
Standard Edition Enterprise Edition	Oracle HTTP Server Oracle Internet File System	Portal and Wireless, and install and configure Oracle Internet File System (available on Supplemental CD)	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Portal Oracle Internet File System

Table 1–2 Oracle9iAS Release 1 and Release 2 Installation Types

Release 1 Installation Type	Release 1 Component Configured	Recommended Release 2 Install Type	Recommended Release 2 Component Configuration
Standard Edition Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Portal Oracle9*i*AS Wireless Oracle Enterprise Java Engine Oracle Internet File System	Portal and Wireless, with Oracle Internet File System installed and configured (available on the Supplemental CD) See the statement of direction for the Java platform at http://otn.oracle.com/tech/java/oc4j/htdocs/oc4j_sod.html	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Portal Oracle9*i*AS Wireless Oracle Internet File System
Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Database Cache	Not applicable.	See the statement of direction for the Oracle9*i*AS Database Cache at http://otn.oracle.com/products/ias/web_cache/htdocs/db_sod.html
Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Web Cache	J2EE and Web Cache	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache
Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Discoverer	Business Intelligence and Forms	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Portal Oracle9*i*AS Discoverer
Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Reports Services	Business Intelligence and Forms	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Portal Oracle9*i*AS Reports Services

Table 1–2 Oracle9iAS Release 1 and Release 2 Installation Types

Release 1 Installation Type	Release 1 Component Configured	Recommended Release 2 Install Type	Recommended Release 2 Component Configuration
Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Forms Services	Business Intelligence and Forms	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Portal Oracle9*i*AS Forms Services
Enterprise Edition	Oracle HTTP Server Oracle Management Server	Oracle9*i*AS Infrastructure	Oracle9*i*AS Metadata Repository Oracle Management Server

Table 1–2 Oracle9iAS Release 1 and Release 2 Installation Types

Release 1 Installation Type	Release 1 Component Configured	Recommended Release 2 Install Type	Recommended Release 2 Component Configuration
Enterprise Edition	Oracle HTTP Server Oracle9*i*AS Portal Oracle9*i*AS Wireless	Business Intelligence and Forms, with the configuration shown in the next column, and with Oracle Internet File System installed and configured (available on the Supplemental CD)	Oracle HTTP Server Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Portal Oracle9*i*AS Forms Services Oracle9*i*AS Wireless Oracle9*i*AS Discoverer Oracle9*i*AS Forms Services Oracle9*i*AS Reports Services
	Oracle Internet File System Oracle9*i*AS Database Cache Oracle9*i*AS Discoverer Oracle9*i*AS Forms Services Oracle9*i*AS Reports Services Oracle9*i*AS Web Cache Oracle Management Server Oracle Enterprise Java Engine	Oracle9*i*AS Infrastructure installation, with Oracle9*i*AS Metadata Repository and Oracle Management Server configuration See the statement of direction for the Oracle9*i*AS Database Cache at http://otn.oracle.com/products/ias/web_cache/htdocs/db_sod.html See the statement of direction for the Java platform at http://otn.oracle.com/tech/java/oc4j/htdocs/oc4j_sod.html	

Components Requiring Database Migration

Table 1–3 lists components that require database migration and their associated migration tasks.

Table 1–3 Components Requiring Database Migration

Component	MIgration Tasks
Oracle9*i*AS Portal	1. Migrate the middle tier (migrate Database Access Descriptors and Providers). 2. Point to the existing Portal metadata in the Portal database.
Oracle9*i*AS Wireless	1. Install Oracle9*i*AS Release 2 (9.0.2) middle tier and Infrastructure. 2. Migrate Panama users to OID. 3. Migrate User Agent Property files to database. 4. Migrate Transformers. 5. Migrate site and node properties to database.
Oracle9*i*AS Discoverer	1. Install Oracle9*i*AS Release 2 (9.0.2) middle tier and Infrastructure. 2. Migrate middle tier (migrate preferences, update URL references, configure session timeout, and migrate viewer customization). 3. Migrate EUL tables in customer database.
Oracle9*i*AS Reports Services	1. Migrate middle tier.
Oracle9*i*AS Personalization	1. Perform database upgrade.
Oracle Internet Directory	1. Install Oracle9*i*AS Release 2 (9.0.2) Infrastructure. 2. Run OIDCA to upgrade the OID instance.
Oracle Management Server	1. Install Oracle9*i*AS Release 2 (9.0.2) Infrastructure. 2. Migrate database.

Component Interdependencies

This section identifies the dependencies between Oracle9*i*AS components, and explains how to manage these for migration.

Table 1–4 Oracle9iAS Component Interdependencies

Component Name	Oracle9*i*AS Release 1 (1.0.2.2.x) Component Interdependency	Oracle9*i*AS Release 2 (9.0.2) Component Interdependency
Oracle9*i*AS Containers for J2EE	None.	Oracle HTTP Server, mod_oc4j
Oracle9*i*AS Portal	Oracle HTTP Server, mod_plsql Oracle9*i*AS Single Sign-On	Oracle HTTP Server, mod_plsql, mod_oc4j Oracle9*i*AS Containers for J2EE Oracle9*i*AS Web Cache Oracle9*i*AS Single Sign-On Oracle Internet Directory Metadata Repository
Oracle9*i*AS Wireless	Oracle HTTP Server, mod_jserv	Oracle HTTP Server, mod_oc4j Oracle9*i*AS Containers for J2EE Oracle9*i*AS Portal Oracle9*i*AS Web Cache Oracle9*i*AS Single Sign-On Oracle Internet Directory Metadata Repository
Oracle9*i*AS Reports Services	Oracle HTTP Server, mod_jserv	Oracle HTTP Server, mod_oc4j Oracle9*i*AS Containers for J2EE Oracle9*i*AS Portal Oracle9*i*AS Web Cache Oracle9*i*AS Single Sign-On Oracle Internet Directory Metadata Repository

Table 1–4 Oracle9iAS Component Interdependencies

Component Name	Oracle9*i*AS Release 1 (1.0.2.2.x) Component Interdependency	Oracle9*i*AS Release 2 (9.0.2) Component Interdependency
Oracle9*i*AS Discoverer	Oracle HTTP Server, mod_jserv	Oracle HTTP Server, mod_oc4j
		Oracle9*i*AS Containers for J2EE
		Oracle9*i*AS Portal
		Oracle9*i*AS Web Cache
		Oracle9*i*AS Single Sign-On
		Oracle Internet Directory
		Metadata Repository
Oracle9*i*AS Forms Services	Oracle HTTP Server, mod_jserv	Oracle HTTP Server, mod_oc4j
		Oracle9*i*AS Containers for J2EE
		Oracle9*i*AS Single Sign-On
		Oracle Internet Directory
		Metadata Repository
Oracle Management Server	Database	Metadata Repository
Oracle9*i*AS Unified Messaging	Database	Oracle HTTP Server, mod_oc4j
		Oracle9*i*AS Containers for J2EE
		Oracle9*i*AS Single Sign-On
		Oracle Internet Directory
		Metadata Repository
Oracle9*i*AS Personalization	Database	Oracle HTTP Server, mod_oc4j
		Oracle9*i*AS Containers for J2EE
Oracle Internet File System	Oracle HTTP Server, mod_jserv	Oracle HTTP Server, mod_oc4j
		Oracle9*i*AS Containers for J2EE

To observe the dependencies during migration, migrate in the following order:

1. Migrate the infrastructure first: Oracle Internet Directory, Oracle9*i*AS Single Sign-On, Oracle Management Server.
2. Migrate the middle tier instances (J2EE and caching components, using the Migration Assistant).

3. Migrate remaining components in the order shown in the table (i.e. Oracle9*i*AS Portal, Oracle9*i*AS Wireless, Oracle9*i*AS Reports Services, etc.), which is the order in which the components are installed.

Preparing to Migrate

This section describes what you need to do before migrating from Release 1 to Release 2. If you have a Release 1 (1.0.2.1.x) installation, you must first migrate to Oracle9*i*AS Release 1 (1.0.2.2.x).

> **See Also:** *Oracle9i Application Server Migration Guide* in the Oracle9*i* Application Server documentation library

Before you begin the migration process:

1. Stop the Oracle9*i*AS Release 2 (9.0.2) instance, if necessary.
2. Back up all configuration and data files in the Oracle9*i*AS Release 2 (9.0.2) instance that are affected by the migration process. If you are unsure which files are affected, see the instructions for migrating the component in question.
3. Stop the Oracle9*i*AS Release 1 (1.0.2.2.x) Oracle9*i*AS instance.
4. Ensure that you have access rights to all target migration directories.
5. Ensure that the Oracle9*i*AS infrastructure patch(es) have been installed. They are available for download on http://metalink.oracle.com on the Patches page (for queries, use the product family '9*i* Application Server').

How To Use This Guide

The guide includes instructions for all components; skip the instructions for components not installed.

1. Use Chapter 7, "Migrating Management Components" to migrate Oracle Enterprise Manager, Oracle9*i*AS Single Sign-On, and Oracle Internet Directory.
2. Follow the instructions in Chapter 2, "Using the Oracle9iAS Migration Assistant" to migrate the Oracle HTTP Server, Oracle9*i*AS Containers for J2EE, and Oracle9*i*AS Web Cache.
3. Inventory the components in your installation, and use the applicable chapters to migrate them. Determine the type and order of the components you will migrate (see "Component Interdependencies" on page 1-11).

4. Use Chapter 3, "Migrating Internet Applications Components" to migrate JServ, OracleJSP pages, Oracle9*i*AS SOAP, Oracle8*i* PLSQL, and Oracle9*i*AS Forms Services.
5. Use Chapter 4, "Migrating Portals Components" to migrate Oracle9*i*AS Portal and UltraSearch.
6. Use Chapter 5, "Migrating Wireless Components" to migrate Oracle9*i*AS Wireless.
7. Use Chapter 6, "Migrating Business Intelligence Components" to migrate Oracle9*i*AS Reports Services, Oracle9*i*AS Discoverer, and Oracle9*i*AS Personalization.
8. Use Chapter 8, "Migrating e-Business Integration Components" to migrate Oracle9*i*AS InterConnect and Oracle Workflow.

Note: Instructions for migrating Oracle9*i*AS Email and Unified Messaging Release 5.2 to Oracle9*i*AS Release 2 (9.0.2) are not included in this guide. They are available at:

http://otn.oracle.com

2

Using the Oracle9*i*AS Migration Assistant

This chapter describes the Oracle9*i*AS Migration Assistant, a tool that migrates Oracle HTTP Server, Oracle9*i*AS Containers for J2EE, and Web Cache from Oracle9*i*AS Release 1 (1.0.2.2.x) to Oracle9*i*AS Release 2 (9.0.2). The Assistant is available in GUI and command-line versions.

> **Note:** Oracle only supports the use of the Assistant for migrations from Release 1 (1.0.2.2.x).

By automating much of the migration process, the Oracle9*i*AS Migration Assistant eliminates errors associated with migrating files manually and expedites what can otherwise be a very lengthy process. It also prepares applications for use immediately after migration, enabling you to use the new Oracle9*i*AS release soon after installation.

This chapter contains these sections:

Understanding the Migration Assistant

This section details the overall functionality of the Assistant, and the specialized functionality for each of the migration options. The Migration Assistant is designed to:

- Migrate the specified components from the Oracle9*i*AS Release 1 (1.0.2.2.x) Oracle home to the Oracle9*i*AS Release 2 (9.0.2) Oracle home.
- Create a log file of all migration activity.
- Run against an installation of Oracle9*i*AS Release 2 (9.0.2) that has not been customized.
- Make the migration process auditable. A flag is set in the target Oracle home, Oracle9*i*AS Release 2 (9.0.2), to indicate that the Assistant has performed a migration there. All migration activities are recorded in a log file. See "Restarting the Oracle9iAS Migration Assistant" on page 2-38.

Before starting the Assistant, read the section for each option you plan to use.

- The Oracle HTTP Server Migration Process on page 2-2
- The Oracle9iAS Containers for J2EE (OC4J) Migration Process on page 2-10
- The Oracle9iAS Web Cache Migration Process on page 2-10
- Using the Oracle9iAS Migration Assistant (GUI Version) on page 2-24
- Using the Oracle9iAS Migration Assistant (Command Line Version) on page 2-34

The Oracle HTTP Server Migration Process

This section describes the functionality of the Oracle HTTP Server migration option, and lists the elements migrated for each component. It contains the following topics:

- Oracle HTTP Server Migration Candidates
- The HTTP Server Directive Migration Process
- Backup and Auditing Measures for Oracle HTTP Server Migration

Oracle HTTP Server Migration Candidates

The Oracle HTTP Server migration option recognizes the following configuration files, programs, static documents, and modules as candidates for migration:

- The `httpd.conf` file. (This file must be selected, or no other files will be migrated.)

  Note: Any configuration file named in an Include directive except for `oracle_apache.conf` or `jserv.conf` can also be migrated. For example, if you had a user-defined configuration file myFile.conf and in `httpd.conf` the directive

  ```
  include "home\myDirectory\myFile.conf"
  ```

  then during migration processing, you can choose to migrate the directive and the file.

- cgi and fastcgi programs not found in the 9.0.2 instance. The default Oracle9*i*AS 1.0.2.2 instance contains

  ```
  ORACLE_HOME_1\Apache\Apache\cgi-bin\printenv
  ORACLE_HOME_1\Apache\Apache\fcgi-bin\echo.exe
  ORACLE_HOME_1\Apache\Apache\fcgi-bin\echo2.exe
  ```

  Any other cgi or fastcgi applications defined by `ScriptAlias` or `Options ExecCGI` directives in `httpd.conf` are migration candidates.

- New static documents or directories found in:

  `ORACLE_HOME_1\Apache\Apache\htdocs`

  with the exception of the following subdirectories:

  `webapp`

  `onlineorders_html`

  `manual`

  `doc`

  `_pages`

  `disco4iv`

  `discwb4`

- Any new files or directories under the following directories:

  `ORACLE_HOME_1\Apache\Apache\htdocs\WEB-INF` (including subdirectories)

`ORACLE_HOME_1\Apache\Apache\htdocs\demo` (including subdirectories)

> **Note:** `index.html` will be migrated only if its file size differs from that of the default. If the content differs, but the file size remains the same, you must migrate `index.html` manually.

- The .dll (dynamic link library) files for any modules not in the Oracle9*i*AS Release 1 (1.0.2.2.x) default set but specified in a LoadModule directive.

Default modules

Table 2–1 lists the default set of modules shipped in Oracle9*i*AS Release 1 (1.0.2.2.x).

Table 2–1 Default Modules

Module Name		
access_module	dir_module	oprocmgr_module
action_module	dms_module	perl_module
alias_module	env_module	proxy_module
anon_auth_module	expires_module	rewrite_module
asis_module	fastcgi_module	setenvif_module
auth_module	headers_module	speling_module
autoindex_module	imap_module	so_module
cern_meta_module	includes_module	ssl_module
cgi_module	info_module	status_module
config_log_module	isapi_module	userdir_module
dbm_auth_module	mime_module	usertrack_module
digest_module	mime_magic_module	
digest_auth_module	negotiation_module	

Default Directives in httpd.conf

Table 2–2 lists the directives found in both versions of the `httpd.conf` file.

Table 2–2 httpd.conf Default Directives

Directive Name		
AccessFileName	HeaderName	
AddCharset	HostnameLookups	ScriptAlias
AddEncoding	IfModule	ServerAdmin
AddHandler	IndexIgnore	ServerName
AddIcon	IndexOptions	ServerRoot
AddIconByEncoding	KeepAlive	ServerSignature
AddIconByType	KeepAliveTimeout	ServerType
AddLanguage	LanguagePriority	SetEnvIf
AddModule	Listen	SetHandler
AddType	LoadModule	SSLEngine
Alias	Location	SSLLog
Allow	LogFormat	SSLLogLevel
AllowOverride	LogLevel	SSLMutex
BrowserMatch	MaxKeepAliveRequests	SSLOptions
ClearModuleList	MaxRequestsPerChild	SSLPassPhraseDialog
CustomLog	MIMEMagicFile	SSLSessionCache
DefaultIcon	Options	SSLSessionCacheTimeout
DefaultType	Order	ThreadsPerChild
Deny	PerlHandler	Timeout
Directory	PerlModule	TransferLog
DirectoryIndex	PerlSendHeader	TypesConfig
DocumentRoot	PidFile	UseCanonicalName
ErrorLog	Port	UserDir
ExtendedStatus	ReadmeName	VirtualHost
Files	ScoreBoardFile	

This is the default set of directives for Oracle HTTP Server migration. These directives occur in the default versions of the `httpd.conf` files in the 1.0.2.2 and 9.0.2 instances. The Assistant highlights the differences so that you can select them for migration. If the setting for a directive is the same in both files, no action is taken.

In the discussion of the migration process below, directives are described as primitive directives or container directives. Primitive directives occupy a single line; for example:

```
Timeout 300

KeepAlive on
```

Container directives occupy multiple lines, have a start directive and an end directive, and contain arguments (which are primitive directives). For example:

```
<Directory "myDirectory">
    Options FollowSymLinks MultiViews
    AllowOverride None
</Directory>
```

The container directive above has start and end directives `<Directory "myDirectory">` and `</Directory>`. The arguments are the primitive directives `Options FollowSymLinks MultiViews` and `AllowOverride None`.

HTTP Server Elements Not Migrated

The Oracle HTTP Server migration option does not migrate:

JServ - JServ is included in Oracle9*i*AS Release 2 (9.0.2) only to support legacy use; the preferred servlet environment is Oracle9*i*AS Containers for J2EE (OC4J). If you used JServ in Release 1 and want to use OC4J in Release 2, see "Migrating JServ to OC4J" on page 3-20. The manual process for migrating JServ is described there.

Configuration files related to the use of mod_plsql - Files such as `oracle_apache.conf`, `plsql.conf`, `dads.conf` and `cache.conf` and the Include directive in `httpd.conf` (for `oracle_apache.conf`) are excluded from the migration.

> **See Also:** Chapter 4, "Migrating Portals Components", "Migrating Database Access Descriptors" for instructions on migrating the mod_plsql configuration and "mod_plsql Parameter Changes in Release 2" for a complete list of parameter changes.

The HTTP Server Directive Migration Process

To migrate directives, the Assistant:

1. Presents directives in the 1.0.2.2 `httpd.conf` file that are different from the default (uncustomized) file, `httpd.conf.default`, or that are new (not part of the default set of directives). The default file, `httpd.conf.default`, must be present or the program will not function.

   By default, all such directives are selected for migration via a checkbox and presented in a scrolling list. You can exclude a directive from the migration by clearing its checkbox.

   Note: An exception to this default selection of directives is the mod_proxy directive. All mod_proxy directives are unchecked by default. They will not be migrated unless they are explicitly selected in the httpd.conf: Directives screen (shown on page 2-31).

   Notes: Container directives are migrated as a whole; when you select a container directive for migration, you select all of the arguments (primitive directives) in it. For this reason, only the top level (that is, the start and end directives) of the container directive is presented as a migration selection.

   Path-related directives are presented with the destination path instead of the source path. For example, a directive from the Release 1 configuration such as

   ORACLE_HOME_1\Apache\Apache\myAlias

   will appear on the screen as

   ORACLE_HOME_2\Apache\Apache\myAlias

2. Writes selected directives to a difference file.

3. Merges the difference file with the 9.0.2 `httpd.conf` file as follows:
   - Default directives with changed settings replace the corresponding directive in the 9.0.2 `httpd.conf` file.
   - Non-default directives (that is, those not listed in Table 2–2) are written to the end of the 9.0.2 `httpd.conf` file.
4. Discards JServ directives.

Migration of SSL Settings

To accommodate the replacement of standard SSL with mod_ossl in Oracle9*i*AS Release 2 (9.0.2), the Assistant automatically creates a directive for mod_ossl, `SSLWallet`, based on the Release 1 configuration. It then starts a program that generates an Oracle wallet. You can choose not to generate the wallet during migration by commenting out the SSL configuration in the Release 1 file before you start the Migration Assistant.

> **See Also:** *Oracle9i Application Server Security Guide*

To ensure that a valid wallet gets generated in the migration, you must specify the trust points (the signers of the certificates) in the Release 1 configuration. There are two ways to do this:

- Concatenate the signer certificates (the certificate chain) into the Release 1 server certificate file.
- Concatenate all of the signers into one file, and use the `SSLCertificateChainFile` directive in the Release 1 `httpd.conf` file.

You can also import other certificate authority certificates into the wallet, by specifying them with the `SSLCACertificateFile` and `SSLCACertificatePath` in the Release 1 `httpd.conf` file.

Note: The Release 1 default SSL certificate is signed by the certificate authority 'oracle demoCA', whose certificate is at

ORACLE_HOME_1\Apache\Apache\conf\ssl.crt\demoCAcert.crt

Before migration, you must set the SSLCertificateChainFile directive to point to the default SSL certificate:

```
SSLCertificateChainFile ORACLE_HOME_
1\Apache\Apache\conf\ssl.crt\demoCAcert.crt
```

The Migration Assistant manages SSL certificate key file and wallet passwords as follows:

Table 2–3 SSL Password Requirements

If Release 1 SSL Certificate Key File has...	Then during migration...
the default 'welcome' password	you are not prompted for a password.
a password other than 'welcome'	you are prompted to enter the correct password.
no password assigned	you are not prompted for a password, and the generated wallet password is set to 'welcome'.

Note the following changes:

- `SSLVerifyClient` is set to `optional` if it was set to `optional_no_ca`.
- `SSLProtocol` is set to `all` if it was set to `TLSv1`.

The following directives are invalid in mod_ossl, and replaced by `SSLWallet`:

- `SSLCertificateFile`
- `SSLCertificateKeyFile`
- `SSLCertificateChainFile`
- `SSLCACertificatePath`
- `SSLCACertificateFile`
- `SSLRandomSeed`
- `SSLVerifyDepth`

During migration, the Assistant extracts certificate-related directives and starts a program that generates a wallet. The wallet-related directives are written to the difference file. The value of SSLWallet is the value of `SSLCertificateFile`, or, if path-related:

`ORACLE_HOME_2\Apache\Apache\conf\ssl.wlt\`*`certificate name`*

Wallet Generation for Default Virtual Host

The Assistant automatically generates the wallet for any virtual host that is SSL-enabled. The default `httpd.conf` file only defines one virtual host. If you override the certificate that is associated with the virtual host named in the Release 1 httpd.conf file, you can manually modify the Release 2 `httpd.conf` file after migration so that it points to the newly generated wallet.

Backup and Auditing Measures for Oracle HTTP Server Migration

The Assistant performs the following functions to provide a way to audit the migration process:

- Creates a backup of the default 9.0.2 `httpd.conf` file named `httpd.conf.migbak`. Because it was written by a parser, this file is not identical in format to `httpd.conf`, but the content is exactly the same.
- Logs all migration activity and errors in

  `ORACLE_HOME_2\migration\log\iASMigration.log`

The Oracle9*i*AS Containers for J2EE (OC4J) Migration Process

This section explains the functionality of the OC4J migration option. It contains the following topics:

- OC4J Migration Candidates
- The OC4J Configuration File Migration Process
- Backup and Auditing Measures for OC4J Migration

OC4J Migration Candidates

The OC4J migration option recognizes these configuration files and applications as candidates for migration:

- The `principals.xml` and `data-sources.xml` files.
- Applications defined in the `server.xml` file in the 1.0.2.2 instance, in .ear file format.

Note: Applications must be in .ear file format and defined in the source `server.xml` file in order to be migrated. Applications are deployed to the 9.0.2 Oracle9*i*AS instance using DCM (Distributed Configuration Management). The assumption is that the application was not installed there previously (the 9.0.2 instance is supposed to be a new, unchanged Oracle9*i*AS installation), but if an application exists, it will be overwritten.

The Assistant will not migrate applications in any format other than .ear (such as .war, exploded, etc.).

Standalone OC4J Instances and Migration

If you installed OC4J in a standalone configuration prior to installing Oracle9*i*AS Release 1 (1.0.2.2.x), be aware that the Migration Assistant only migrates the OC4J instance bundled with Oracle9*i*AS Release 1 (1.0.2.2.x).

For example, suppose that:

1. An OC4J instance was installed and configured, with applications deployed.
2. Subsequently, Oracle9*i*AS Release 1 (1.0.2.2.x) was installed, with its bundled OC4J instance. The applications are still deployed on the original instance.
3. The Migration Assistant is run.

   No applications are migrated, since they were not found on the Oracle9*i*AS Release 1 (1.0.2.2.x) OC4J instance.

The OC4J Configuration File Migration Process

The OC4J migration option does the following:

1. Copies selected `principals.xml` and `data-sources.xml` from *ORACLE_HOME_1*\J2EE_containers\j2ee\home\config to *ORACLE_HOME_2*\j2ee\home\config.
2. Reads application information from the `server.xml` file in *ORACLE_HOME_1* and prompts you to select the applications to migrate.
3. Starts a default OC4J instance in *ORACLE_HOME_2*.
4. Re-deploys the migrated applications in *ORACLE_HOME_2*.
5. Stops the default OC4J instance (and all of Oracle9*i*AS, if it was running).

J2EE Compliance Requirements for OC4J Migration

In Oracle9*i*AS Release 2 (9.0.2), OC4J deployment enforces J2EE compliance rules. For this reason, the Oracle9*i*AS Migration Assistant may not migrate applications that are not 100% J2EE compliant. The Assistant simply reads the files and attempts to deploy them to Oracle9*i*AS Release 2 (9.0.2); if deployment fails, it could be because an application is not J2EE compliant.

If the Assistant cannot deploy an application for any reason, it logs the exception, however, the exception may not be explicitly described as a compliance issue.

While the development of J2EE applications is standardized and portable, the XML configuration files are not. You may have to configure multiple XML files before deploying an application to OC4J. The configuration needed depends on the services the application uses. For example, if the application uses a database, you must configure the DataSource object in the `data-sources.xml` file.

Validating EAR Files for J2EE Compliance

The dcmctl utility (*ORACLE_HOME_2*\dcm\bin\dcmctl) provides a J2EE compliance validation command. It takes one input, the name of an EAR file, and lists non-compliant characteristics of that file. The syntax is:

```
dcmctl validateEarFile -v -f name.ear
```

where name is the name of the EAR file. -v specifies the verbose option of dcmctl; this provides the most detailed output of commands.

You must configure proxy settings so that the validation routine can access DTDs on the Web, if necessary (for example, on the Sun Microsystems site). To do this, you define an environment variable called ORACLE_DCM_JVM_ARGS, which specifies a hostname and port for the proxy. The command to do this is:

```
set ORACLE_DCM_JVM_ARGS=-Dhttp.proxyHost=host -Dhhtp.proxy.port=port
```

Example 2–1 validateEarFile Command and Output for J2EE-Compliant Application

```
ORACLE_HOME_2\dcm\bin> dcmctl validateEarFile -v -f ORACLE_HOME_
2\j2ee\home\applications\simple.ear

ORACLE_HOME_2\dcm\bin> echo off

    No J2EE XML/DTD validation errors were found
```

Example 2–2 validateEarFile Command and Output for non- J2EE-Compliant Application

```
ORACLE_HOME_2\dcm\bin> dcmctl validateEarFile -v -f
ORACLE_HOME_2\j2ee\home\applications\petstore.ear
ORACLE_HOME_2\dcm\bin> echo off

    Warning: J2EE/DTD validation errors were found
    Cannot get xml document by parsing WEB-INF\web.xml in petstore.war:
    Invalid element 'servlet' in content of 'web-app', expected elements
    '[servlet-mapping, session-config, mime-mapping, welcome-file-list,
    error-page, taglib,resource-ref, security-constraint, login-config,
    security-role, env-entry, ejb-ref]'.
```

It is a good idea to review all applications for overall J2EE compliance before migrating them, since there are cases in which an application is deployable, but delivers unpredictable or undesirable server behavior. For example, ensure that content is served correctly by defining a unique context root for each application in `application.xml`.

Backup and Auditing Measures for OC4J Migration

The Assistant performs the following functions to provide a way to audit the migration process:

- Creates a backup of each configuration file. The copy has the same filename, and the extension `.SAVED_COPY`.
- Logs all migration activity and errors in

  ORACLE_HOME_2\migration\log\iASMigration.log

The Oracle9*i*AS Web Cache Migration Process

This section explains the functionality of the Web Cache migration option. It contains the following topics:

- Web Cache Migration Candidates
- The Web Cache Migration Process
- Backup and Auditing Measures for Web Cache Migration

Web Cache Migration Candidates

The Web Cache migration option recognizes most of the elements in the `webcache.xml` file in *`ORACLE_HOME_1`*. They are listed in "The Web Cache Migration Process" below.

The Assistant does not migrate:

- The `internal.xml` file. You must manually migrate network timeouts, keepalive values, and `osrecv` values from the CALYPSONETINFO element of the `internal.xml` file to the `webcache.xml` file in *ORACLE_HOME_2*.
- The MULTIVERSIONHEADERRULE, if the header used for disambiguation is the Host header. Web Cache in Oracle9*i*AS Release 2 (9.0.2) supports multiple sites. See the Web Cache documentation for site-to-server mapping.

Migration of Session Definitions

A session defnition consists of a session name, a cookie, a URL parameter, and a default value. The Migration Assistant migrates session definitions as follows:

- If the session name, cookie, URL parameter, and default value, are the same in Release 2 as in Release 1, then the session definition is not migrated.
- If the session name is the same, but the cookie, URL parameter, or default value is different, then the Migration Assistant migrates the session as it is, changes the name of the Release 2 session, and updates its references.

Warning: Any application that was using WEBCACHETAG with reference to the original Release 2 Web Cache session definition must be modified to use the re-named session definitions.

The Web Cache Migration Process

The Web Cache migration option does the following:

1. Copies the following elements of the `webcache.xml` file from *ORACLE_HOME_1* to *ORACLE_HOME_2*:

   - SECURITY

     SECURESUBNET (Sub-element of SECURITY; trusted subnets).

     Note: The Migration Assistant does not migrate any passwords. The administrator and invalidation passwords have default values when Release 2 is installed; see the Web Cache documentation for the default passwords.

   - WATCHDOG
   - REQUESTBACKLOGTIMELIMIT (an attribute of the SITE element)

     Copied to the first SITE element of the `webcache.xml` file in *ORACLE_HOME_2*.

   - ERRORPAGES

     Copied from the *ORACLE_HOME_1* `webcache.xml` file to the *ORACLE_HOME_2* `webcache.xml` file under the first SITE element.

   - MULTIVERSIONCOOKIESRULE

     Copied from the `webcache.xml` file in *ORACLE_HOME_1* and merged with the data in the same sections of the GLOBALCACHINGRULES element in the `webcache.xml` file in *ORACLE_HOME_2*. This can result in duplicate or redundant multi-version cookies rules. See "Completing the Web Cache Migration" on page 2-37 for instructions on resolving this.

   - SESSIONCACHINGRULE

     Copied from the *ORACLE_HOME_1* `webcache.xml` file to the *ORACLE_HOME_2* `webcache.xml` file, in the GLOBALCACHINGRULES section. This can result in duplicate or redundant session caching rules. See "Completing the Web Cache Migration" on page 2-37 for instructions on resolving this.

- EXPIRATIONRULE

  Copied from the `webcache.xml` file in *`ORACLE_HOME_1`* and merged with the data in the same sections of the GLOBALCACHINGRULES element in the `webcache.xml` file in *`ORACLE_HOME_2`*. This can result in duplicate or redundant expiration rules. See "Completing the Web Cache Migration" on page 2-37 for instructions on resolving this.

- CACHEABILITY

  Copied from the *`ORACLE_HOME_1`* `webcache.xml` file to the *ORACLE_HOME_2* `webcache.xml` file, in the GLOBALCACHINGRULES section. This can result in duplicate or redundant cacheability rules. See "Completing the Web Cache Migration" on page 2-37 for instructions on resolving this.

  Note: Since the migrated CACHEABILITY rules are defined in the GLOBALCACHINGRULES section, they apply to all of the SITES defined in the *`ORACLE_HOME_2`* `webcache.xml` file.

  If you define another SITE element later in *`ORACLE_HOME_2`*, you must also define cacheability rules for it. The rules defined in the GLOBALCACHINGRULES section will apply to the new SITE also.

- HOST

  All of the application web servers from the *`ORACLE_HOME_1`* `webcache.xml` file are migrated to the *`ORACLE_HOME_2`* `webcache.xml` file. Host IDs are generated for each of these hosts.

- EVENTLOG
- ACCESSLOG (except for the LOGDIR attribute)
- RESOURCELIMITS

Backup and Auditing Measures for Web Cache Migration

The Assistant performs the following functions to provide a way to audit the migration process:

1. Creates a backup copy of the `webcache.xml` file from *`ORACLE_HOME_2`*. The backup file is named `webcache.xml.backup`.
2. Logs all migration activity and errors in

   `ORACLE_HOME_2``\migration\log\iASMigration.log`

> **Note:** The logging mechanism writes all pathnames with forward slashes instead of backward slashes.

Installing the Migration Assistant

This section provides information about hardware and software requirements for installation of Oracle9*i*AS Migration Assistant. The topics include:

- Hardware Requirements
- Software Requirements
- Starting Oracle Universal Installer

Hardware Requirements

The following table, Table 2–4, "Oracle9iAS Migration Assistant Hardware Requirements" contains the minimum hardware requirements for the Oracle9*i*AS Migration Assistant.

Table 2–4 Oracle9iAS Migration Assistant Hardware Requirements

Hardware Items	Minimum Requirements
CPU	An Intel-compatible 486 or higher processor
Memory	128 MB
Monitor	256 color viewing capability

Software Requirements

The Oracle9*i*AS Migration Assistant requires the following software:

- Oracle9*i*AS Release 2 (9.0.2) Middle Tier
- Oracle Universal Installer version 2.1.0.9 or higher
- JDK 1.3.1

Operating System Requirements

Table 2–5 lists the operating system requirements for the Oracle9*i*AS Migration Assistant.

Table 2–5 Oracle9iAS Migration Assistant Operating System Requirements

Software Items	Version
Operating System	■ Microsoft Windows NT 4.0 with Service Pack 6a ■ Microsoft Windows 2000 with Service Pack 1 or higher
Virtual Memory	At least 360 MB of free virtual memory. To change the virtual memory setting, go to Control Panel and select System, then change the amount of memory under the Performance tab.

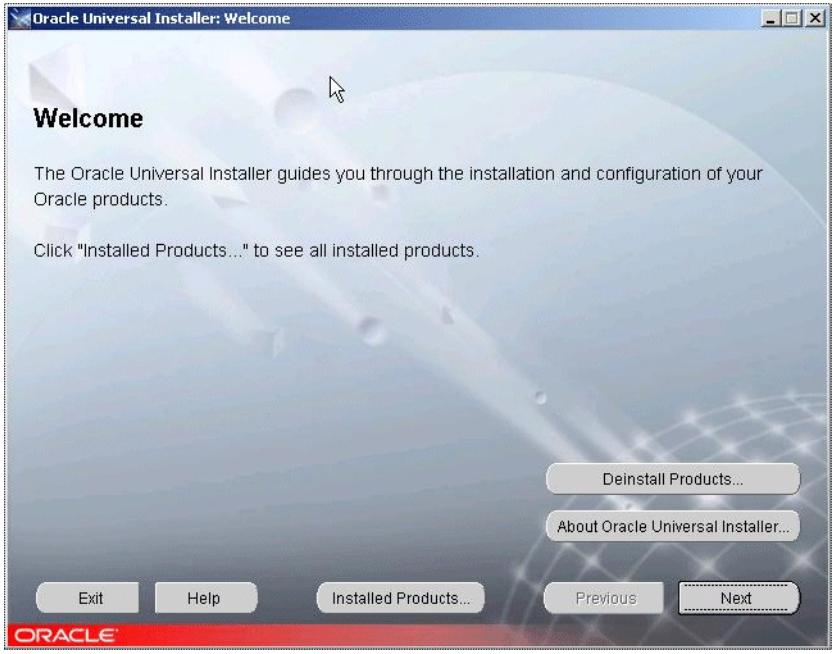
Starting Oracle Universal Installer

Follow these steps to launch Oracle Universal Installer, which installs the Oracle9*i*AS Migration Assistant:

1. Stop all Oracle processes and services (for example, the Oracle database) running on the computer.
2. Ensure that you are logged in to the computer as a member of the Windows Adminstrators group.
3. Insert the Supplemental CD into the CD-ROM drive.
4. Open the **Start** menu, select **Run**, and type `G:\Setup.exe`, where G is the CD-ROM drive letter.

   The Welcome screen appears (Figure 2–1).

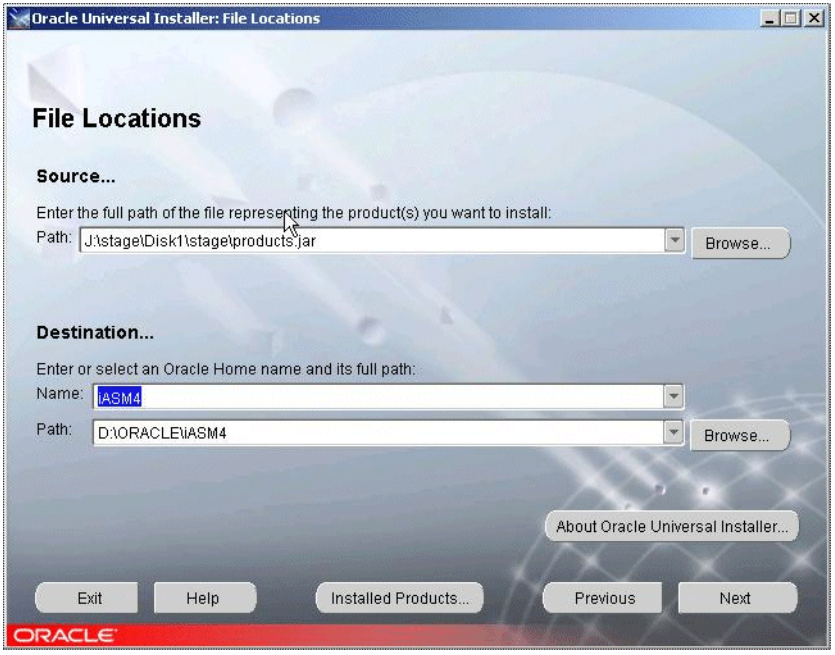
Figure 2–1 Oracle Universal Installer Welcome screen



5. Click Next.

The File Locations screen appears (Figure 2–2).

Figure 2–2 Oracle Universal Installer File Locations screen



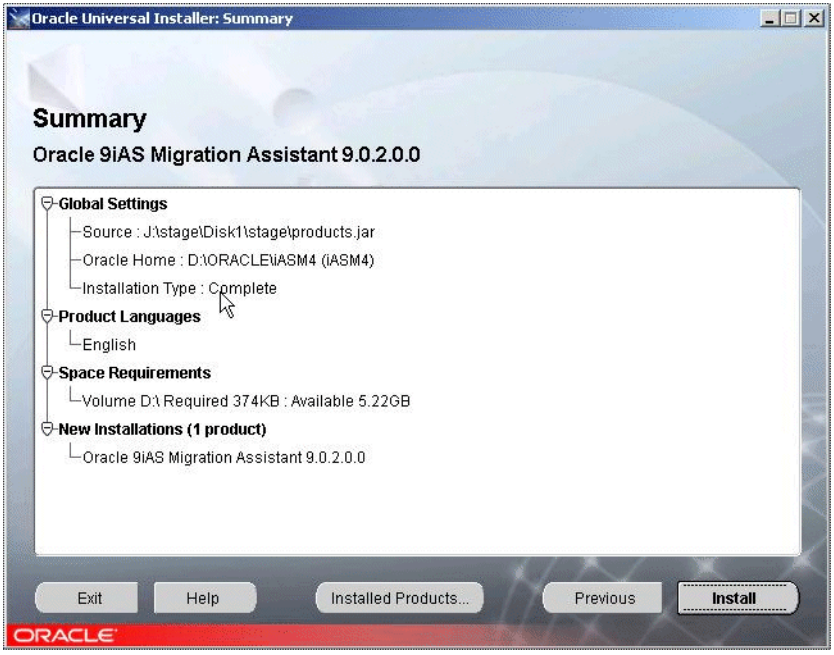
6. Complete the Source... field with the location of the products.jar file on the Supplemental CD. You can type the path, or click Browse... to navigate to it.
7. Complete the Destination... field with the Oracle home in which you want to install the Oracle9*i*AS Migration Assistant. The Name drop-down box contains a list of all middle-tier Oracle homes on the computer.

> **Note:** You can only install the Assistant into an existing Oracle9*i*AS Release 2 (9.0.2) Oracle home in a middle tier type of installation. You cannot install it into an infrastructure installation.

8. Click Next.

   The Summary screen appears (Figure 2–3).

Figure 2–3 Oracle Universal Installer Summary screen



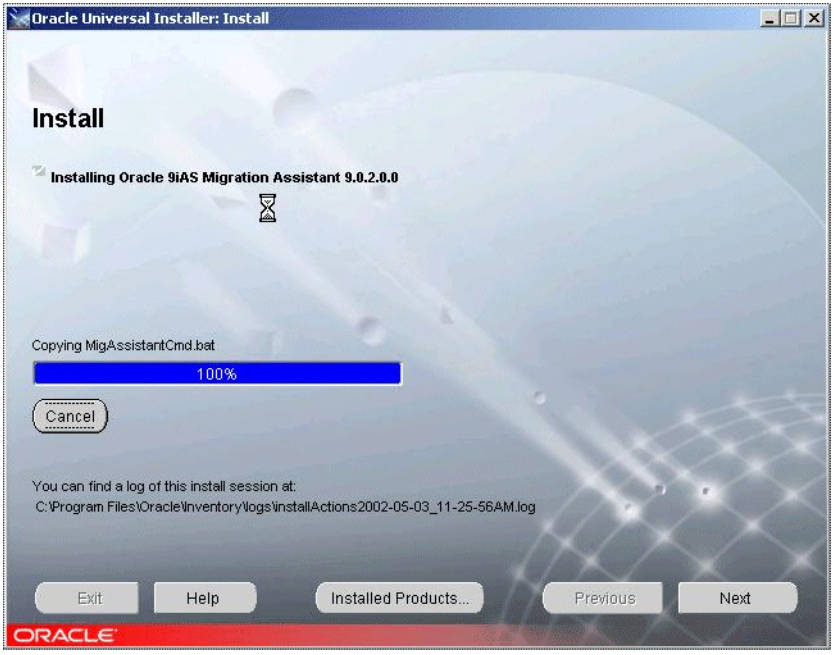
This screen summarizes the choices on the File Locations screen: the path to `products.jar` and the destination Oracle home, as well as the installation type, language, and space requirements.

9. If you need to change the source or destination path, click Previous and enter or select the path you want. Otherwise, continue with Step 6.

10. Click Next.

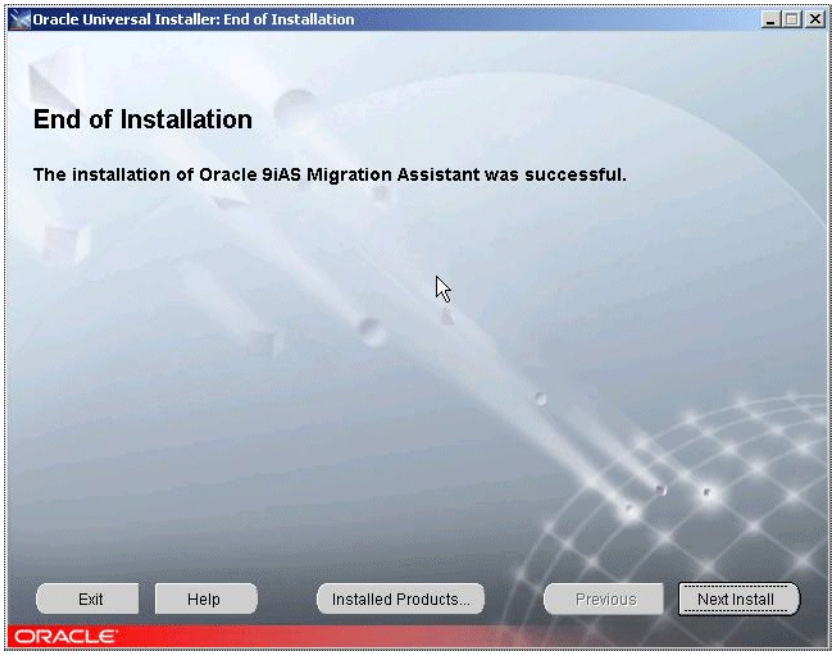
    The Install screen appears (Figure 2–4).

Figure 2–4 Oracle Universal Installer Install screen



This screen shows the progress of the installation of the Assistant to the selected Oracle home. The text above the progress bar indicates the installation actions as they occur. When the process completes, the End of Installation screen appears (Figure 2–5).

Figure 2–5 Oracle Universal Installer End of Installation screen



This screen indicates the results of the installation process.

11. Click Exit.

Migrating Applications

This section provides guidelines for preparing for a migration, and step-by-step instructions for starting and operating the Assistant.

Preparing to Migrate

This section outlines prerequisite steps for migrating.

Note: You do not need to start Oracle9*i*AS before using the Migration Assistant. The Assistant will start an OC4J instance to deploy the OC4J applications, and then stop it when deployment is complete.

Information Requirements

Before you start the Assistant, be prepared with the following (as required for the components you plan to migrate):

- Password for the SSL certificate key file for the Oracle HTTP Server, if a password other than the default 'welcome' password was assigned. (See Table 2–3 for password requirements). This password is used to generate the wallet during SSL conversion. If you enter the password incorrectly 3 times, components containing the SSL-related information are set to non-migratable status (excluded from the migration).
- Familiarity with the Oracle HTTP Server directives and their purpose, as related to your configuration.

SSL Configuration Requirements

If you want to use SSL with the Oracle HTTP Server in the Oracle9*i*AS Release 2 (9.0.2) environment, ensure that the following directives are configured (uncommented) in the `httpd.conf` file before you start the Assistant:

- `SSLCertificateFile`
- `SSLCertificateKeyFile`

`SSLCertificateFile` and `SSLCertificateKeyFile` are necessary for any SSL-enabled web site, and if the configuration being migrated is an SSL configuration, these will be configured in `httpd.conf` in the Release 1 installation.

You must also ensure that the trust points are specified by some directive in the Release 1 installation. See "Migration of SSL Settings" on page 2-8 for instructions on how to do this.

Using the Oracle9*i*AS Migration Assistant (GUI Version)

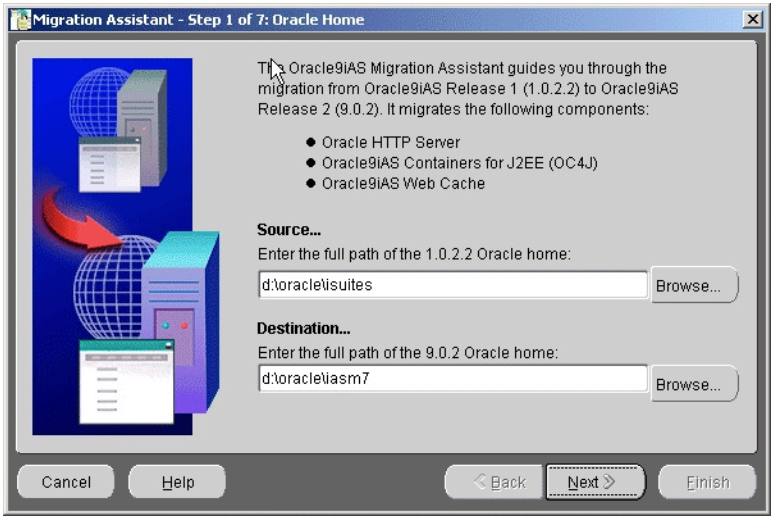
1. If necessary, change to the directory in which the Assistant is installed (*ORACLE_HOME_2*\migration).

2. Start the Assistant with the command:

   ```
   MigAssistant.bat
   ```

   The Oracle Home screen appears (Figure 2–6).

Figure 2–6 Oracle Home screen



3. Complete the Source... field with the full path to *ORACLE_HOME_1*. You can:
   - Type the full path into the field.
   - Click Browse... to specify the path by navigating.
4. Complete the Destination... field with the full path to *ORACLE_HOME_2*. You can:
   - Type the full path into the field.
   - Click Browse... to specify the path by navigating.

   If OC4J was not found in the Source... path you specified, the J2EE Home screen appears (Figure 2–7).

Note: The Migration Assistant performs an existence check for OC4J in:

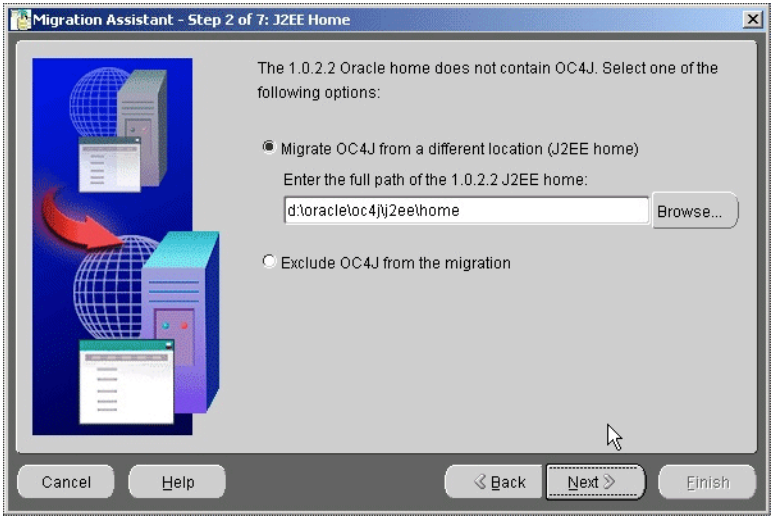
ORACLE_HOME_1\J2EE_containers\j2ee\home,

and presents the J2EE Home screen if it does not find it there.

However, the Oracle9*i*AS Release 1 (1.0.2.2.x) installation may have placed it in:

ORACLE_HOME_1\J2EE_containers**oc4j**\j2ee\home.

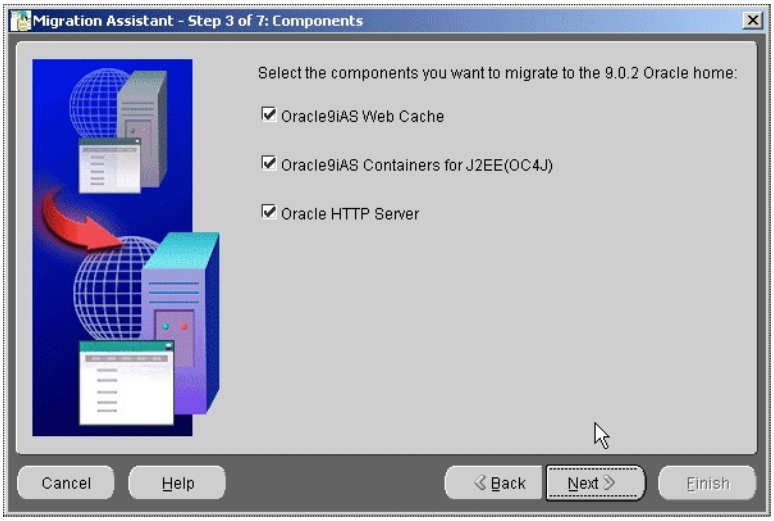
Figure 2–7 J2EE Home screen



5. If OC4J was installed in a location other than the Source... path, click the Migrate OC4J radio button and complete the 1.0.2.2 J2EE home path (type it or navigate to it), then click Next.
6. If OC4J is not installed, or you do not intend to migrate it, click the Exclude OC4J radio button, then click Next.

   The Components screen appears (Figure 2–8). By default, all of the components are selected for migration.

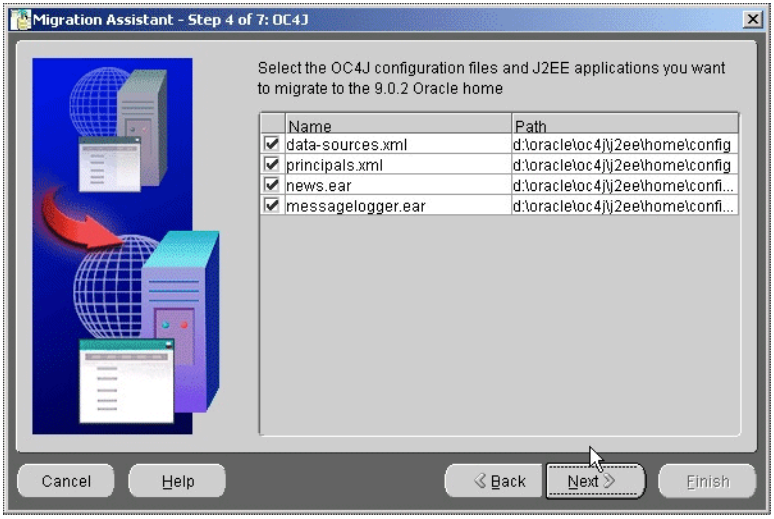
Figure 2–8 Components screen



7. To deselect a component for migration, click the checkbox to clear it.
8. Click Next.

   If OC4J was selected, the OC4J screen appears (Figure 2–9). By default, all applications are selected for migration. See "OC4J Migration Candidates" on page 2-10 for information on how the configuration files and applications are identified for migration.

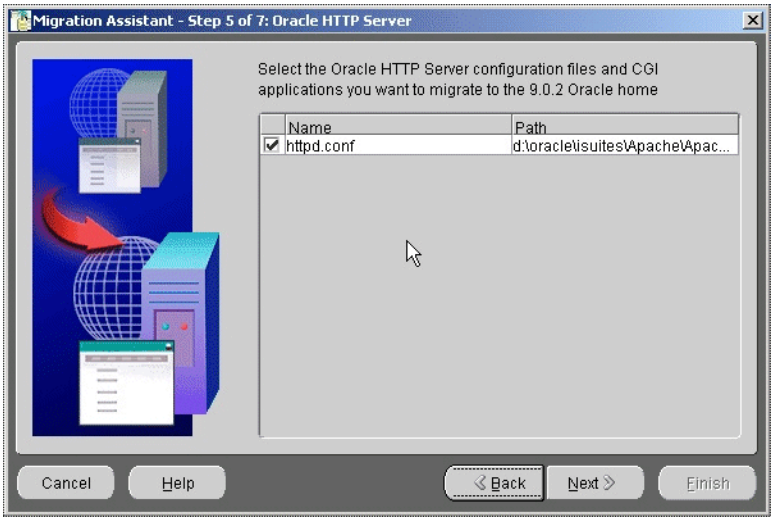
Figure 2–9 OC4J screen



9. To deselect a file or application for migration, click the checkbox to clear it.
10. Click Next.

    If Oracle HTTP Server was selected, the Oracle HTTP Server screen appears (Figure 2–10). By default, all of the configuration files, CGI applications, and static documents found are selected for migration. See "Oracle HTTP Server Migration Candidates" on page 2-2 for information on how the configuration files and applications are identified for migration.

Figure 2–10 Oracle HTTP Server screen

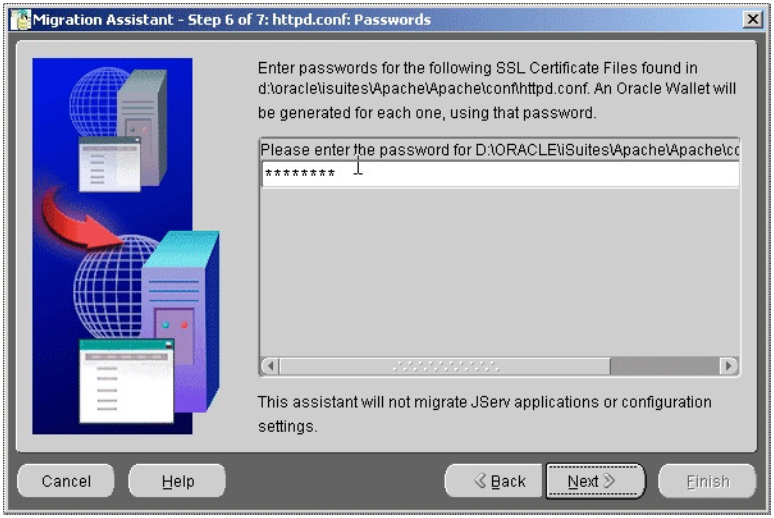


11. To deselect a file or application for migration, click the checkbox to clear it.

12. Click Next.

If an SSL certificate file was found with a password other than the default 'welcome', the httpd.conf: Passwords screen appears (Figure 2–11).

Figure 2–11 httpd.conf: Passwords screen



13. Complete the password field with the password for the certificate key file. The SSL wallet will be generated with this password.

> **Note:** The Assistant allows you three attempts to enter the correct password before setting the SSL-enabled component to non-migratable status. If this happens, you must migrate the component manually.

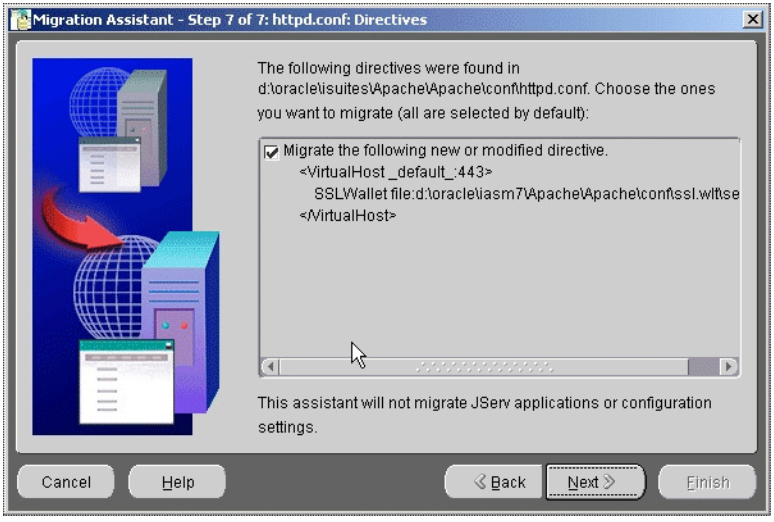
14. Click Next.

    The httpd.conf: Directives screen appears (Figure 2–12), which is populated with the directives you can choose to migrate. By default, all directives except for mod_proxy are selected for migration. See "The HTTP Server Directive Migration Process" on page 2-7 for information on how the Assistant compiled this list of directives.

Note: A new directive will appear for the default virtual host to indicate the location of the wallet. Deselect this directive if you do not want it to be appended to the Release 2 `httpd.conf` file. See "Wallet Generation for Default Virtual Host" on page 2-10.

15. To deselect a directive, click the checkbox to clear it.

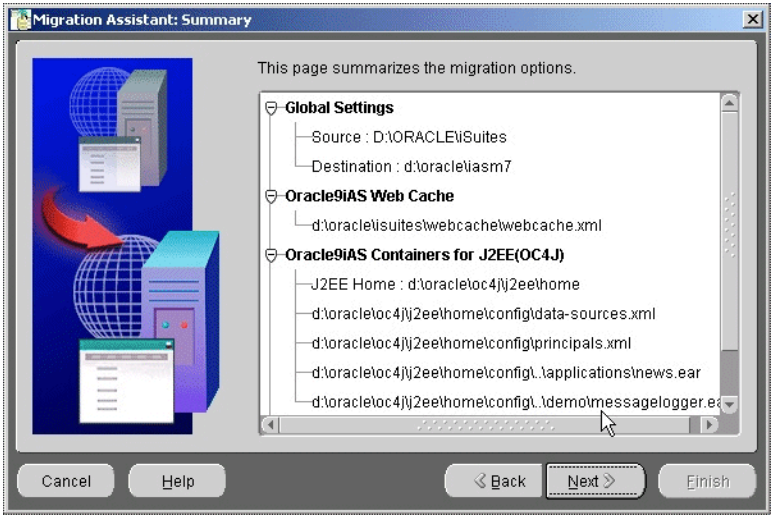
Figure 2–12 httpd.conf: Directives screen



16. Click Next.

The Summary screen appears (Figure 2–13), showing your choices of Oracle homes, configuration files, and applications.

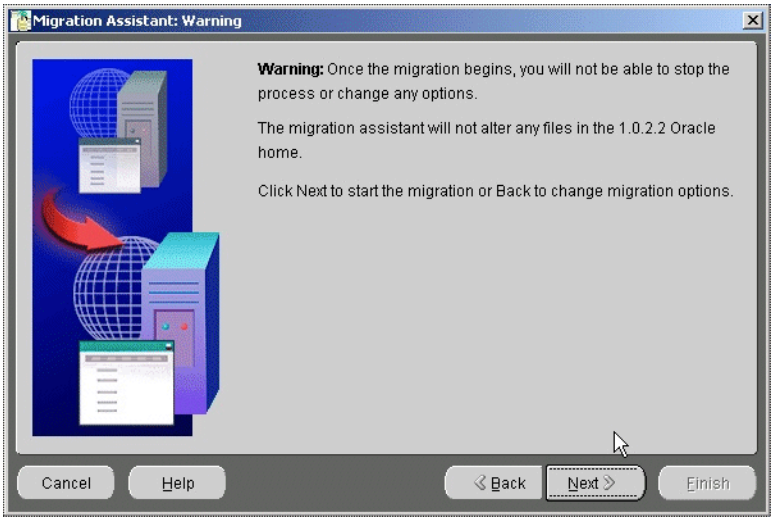
Figure 2–13 Summary screen



17. Review the choices.
18. If necessary, click Back to navigate to previous screens to make changes.
19. Click Next.

    The Warning screen appears (Figure 2–14).

Figure 2–14 Warning screen



Warning: If you click Next now, the Assistant will begin to apply the current migration selections. Once the migration begins, you can click Cancel to stop the Assistant. It will finish the migration in progress (Oracle HTTP Server, OC4J or Web Cache), and then stop. No other selected migrations will start.

To undo a migration, you must manually restore the configuration files in the 9.0.2 instance from a backup.

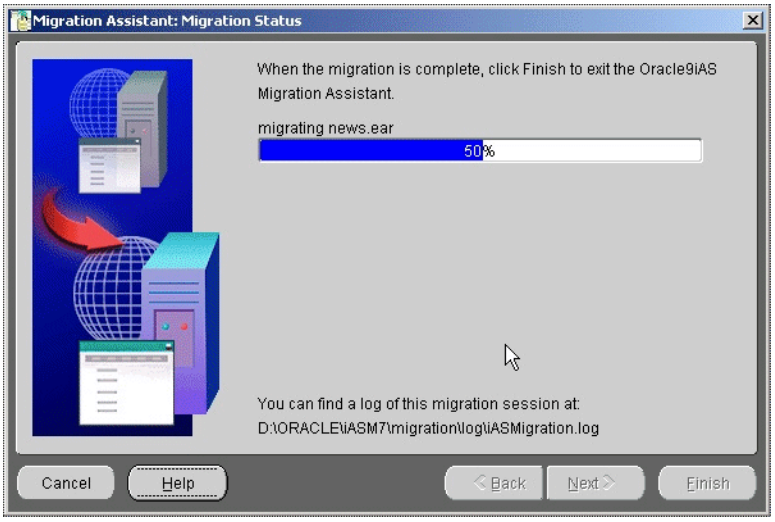
For a description of backups and file names, see:

- "Backup and Auditing Measures for Oracle HTTP Server Migration" on page 2-10
- "Backup and Auditing Measures for OC4J Migration" on page 2-13
- "Backup and Auditing Measures for Web Cache Migration" on page 2-17

20. Click Next to apply the migration choices you have made, or click Back to navigate to previous screens to make changes.

The Migration Status screen appears with a progress bar showing the percentage of the migration completed (Figure 2–15).

Figure 2–15 Migration Status screen



21. Click Finish to close the Migration Assistant.
22. Review the log files.
23. Perform tests for each application you migrated.

Using the Oracle9*i*AS Migration Assistant (Command Line Version)

1. If necessary, change to the directory in which the Assistant is installed (*ORACLE_HOME_2*\migration).
2. Start the Assistant with the command:

   ```
   MigAssistantCmd.bat
   ```

   The following prompt appears:

   ```
   Source Oracle home?
   ```

3. Enter the path to *ORACLE_HOME_1*.

The following prompt appears:

```
Target Oracle home?
```

4. Enter the path to *ORACLE_HOME_2*.

   A prompt resembling the following appears.

   ```
   Select components to migrate Migrate all components?[YES]n
   ```

5. Press Enter to accept the default in brackets, or type n and press Enter to answer No.

   The next prompt appears.

6. Repeat Step 4 for each prompt. The remaining prompts resemble the following:

   ```
   Migrate all subComponents of PlugIn  Oracle9iAS Web Cache?[YES]n

   Migrate webcache.xml[YES]

   Migrate all subComponents of PlugIn Oracle9iAS Containers for
   J2EE(OC4J)?[YES]n

   Migrate data-sources.xml[YES]

   Migrate principals.xml[YES]

   Migrate all subComponents of PlugIn  Oracle HTTP Server?[YES]n

   Migrate httpd.conf[YES]

   Migrate Globals.java[YES]

   Migrate Globals.class[YES]

   Migrate Globals$__jsp_StaticText.class[YES]

   Migrate globals.ser[YES]

   Migrate _index.java[YES]

   Migrate _index.class[YES]

   Migrate _index$__jsp_StaticText.class[YES]

   Questionaire PlugIn Oracle HTTP Server httpd.conf
   ```

```
Please enter the password for ORACLE_HOME_
1\conf\ssl.crt\server.crt:[welcome]
```

7. Press Enter to accept the default password `welcome`, or type the password and press Enter.

   A summary of selections resembling the following appears:

```
Summary page PlugIn Oracle9iAS Web Cache
webcache.xml PlugIn
Oracle9iAS Containers for J2EE(OC4J)
data-sources.xml
principals.xml
news.ear
petstore.ear
atm.ear PlugIn
Oracle HTTP Server
Globals.java
Globals.class
Globals$__jsp_StaticText.class
globals.ser  _index.java  _index.class  _index$__jsp_StaticText.class
Start migration...
```

8. Press Enter to start the migration.

   Migration processing begins. Status messages resembling the following appear:

```
Migrating plugin   Oracle9iAS Web Cache
Outcome Status code       0
Status description            SUCCESS
Migrating plugin   Oracle9iAS Containers for J2EE(OC4J)
Outcome Status code       0
Status description            SUCCESS
Migrating plugin   Oracle9iAS HTTP Server
Outcome Status code       0
Status description            SUCCESS
```

9. Review the log files.

10. Perform tests for each application you migrated to ensure it is working as it did in the previous release.

Completing the Web Cache Migration

To complete the Web Cache migration, you may need to perform the following tasks. Use the Administration user interface to review and, if necessary, change the configuration as follows:

> **Note:** If, because of a port conflict, the Web Cache administration process does not start, you must specify the correct administration port in the MULTIPORT element of the `webcache.xml` file, and restart Web Cache.

- Create the site-to-server mappings.
- To use the same Operations Port or Listening Port wallets as in Release 1, obtain the wallet information from Release 1 and modify the wallet information in Release 2, using the Administration user interface.
- Review post-installation user changes.

  The Migration Assistant can be invoked any time after the Oracle9*i*AS Release 2 (9.0.2) installation. If changes were made to the `webcache.xml` file, they are preserved. There may be redundant cacheability rules. You must review the file to resolve these. The rules themselves, and the order in which they appear, determine the caching behaviors executed by Web Cache.
- Resolve any port conflicts introduced by the migration.

  Port numbers are not migrated from the `webcache.xml` file. Compare the Release 1 and Release 2 `webcache.xml` files to ensure that there are no port conflicts after the Migration Assistant has executed.

  Web Cache does not migrate administration, listen, statistics, and invalidation port numbers. To use the Release 1 port numbers in Release 2, perform the following steps:

  1. Determine the Web Cache port numbers used in Release 1.
  2. Use the Administration user interface to change the port numbers in Release 2.
- Review session caching rules and resolve any duplications.
- Review expiration rules and resolve any duplications.
- Review multi-version cookies rules and resolve any duplications.

Restarting the Oracle9*i*AS Migration Assistant

You must restore the Oracle9*i*AS Release 2 (9.0.2) instance to its pre-migration condition before you restart the Migration Assistant. Follow these steps:

1. Delete the flag file `firstRun` from the Release 2 Oracle home directory.
2. Restore all configuration files and directories to their pre-migration state. (Use the log file to determine which files were altered or copied.)
3. Follow the instructions in:

   "Using the Oracle9iAS Migration Assistant (GUI Version)" on page 2-24

   or

   "Using the Oracle9iAS Migration Assistant (Command Line Version)" on page 2-34.

3

Migrating Internet Applications Components

This chapter explains how to change the necessary configuration files, application deployment files, and metadata schema in order to migrate Internet Applications components. It contains these major sections:

Migrating Oracle9*i*AS Containers for J2EE

This section explains how to perform the following migrations from Oracle9*i*AS Release 1 (1.0.2.2.x) to Oracle9*i*AS Release 2 (9.0.2):

- JSP to OC4J
- JServ to OC4J

Migrating JSP to OC4J

This section covers the major considerations for Oracle9*i* Application Server customers in migrating JSP applications from Oracle9*i*AS Release 1 (1.0.2.2.x) to Oracle9*i*AS Release 2 (9.0.2). This involves adapting from an Apache JServ servlet 2.0 environment to the Oracle9*i*AS Containers for J2EE (OC4J) servlet 2.3 environment supplied in the new release. There are also differences in the JSP container.

The following topics are covered:

Note: See the servlet migration section for relevant information about application environment, servlet context and servlet path mapping, application root configuration, OC4J configuration, and deployment.

JSP Pages in Servlet 2.3 Environments Compared to Servlet 2.0

Prior to Oracle9*i*AS Release 2 (9.0.2), JServ was the primary servlet environment. There are significant differences between the OC4J servlet environment, a servlet 2.3 implementation, and the JServ servlet environment, which is a servlet 2.0 implementation. The following list offers a summary of highlights:

- **Standard application environment versus globals.jsa**—A well-defined concept of a Web application exists in the servlet 2.2 and 2.3 definitions, but did not exist in the servlet 2.0 definition. The servlet standard and OC4J implementation now define the concept of the document root of a Web application, and how to package an application. See "Application Environment and Related Considerations" on page 3-6 for more information. For JServ, Oracle's JSP implementation has emulated the application framework through the `globals.jsa` mechanism. Use of a `globals.jsa` file is no longer necessary in Oracle9*i*AS Release 2 (9.0.2), and is not supported in OC4J.
- **Request dispatcher**—The concept of the request dispatcher was introduced in the servlet 2.1 specification. This mechanism allows a JSP page or servlet to include content from another page or servlet, or to forward execution to another page or servlet. For JServ, Oracle's JSP implementation emulated request dispatcher functionality. For OC4J, this emulation is no longer necessary.
- **Attribute storage**—Beginning with the servlet 2.1 specification, request-level and application-level attribute storage is possible. Developers can use HTTP

request objects and servlet context (application-level) objects to store and retrieve state information. For JServ, Oracle's JSP implementation has emulated this functionality. For OC4J in Oracle9*i*AS Release 2 (9.0.2), this emulation is no longer necessary.

- **Servlet filtering**—The concept of servlet filtering was introduced in the servlet 2.3 specification. This mechanism allows verification and modification of HTTP request and response objects by developers. This might be used, for example, for common headers and footers or customized authentication or authorization. This functionality was not available in previous Oracle9*i*AS or Oracle JSP releases, but is available in OC4J in Oracle9*i*AS Release 2 (9.0.2).
- **Globalization**—The servlet 2.3 specification provides globalization support for HTTP parameters through the standard `setCharacterEncoding()` method of the HTTP request object. Previous Oracle JSP implementations supported this through the `translate_params` configuration parameter, and later through the `setReqCharacterEncoding()` method of a public utility class. You should now migrate your applications to `setCharacterEncoding()`. See "Globalization Considerations for HTTP Parameters" on page 3-18.

Migration Considerations for the JSP Container

In Oracle9*i*AS Release 1 (1.0.2.2.x), the first release to include OC4J, there were two JSP containers:

- a container developed by Oracle and formerly known as "OracleJSP"
- a container licensed from Ironflare AB and formerly known as the "Orion JSP container"

The OracleJSP container offered a number of advantages, including useful value-added features and enhancements such as for globalization and SQLJ support. The Orion container also offered advantages, including superior speed, but had disadvantages as well. It did not always exhibit standard behavior when compared to the JSP 1.1 reference implementation (Tomcat), and its support for internationalization and globalization was not as complete.

With Oracle9*i*AS Release 2 (9.0.2), these two containers have been integrated into a single JSP container, referred to as the "OC4J JSP container". This container offers the best features of both previous versions, runs efficiently as a servlet in the OC4J servlet container, and is integrated with other OC4J containers as well. The integrated container primarily consists of the OracleJSP translator and the Orion JSP container runtime, running with a newly simplified dispatcher and the OC4J 1.0.2.2 core runtime classes.

The Orion container was the default JSP container in Oracle9*i*AS Release 1 (1.0.2.2.x). If that is the container you used, there are a number of considerations when migrating to the OC4J JSP container in Oracle9*i*AS Release 2 (9.0.2). Note that it is possible, but not advisable, to configure OC4J to continue to use the original Orion JSP container. If you must do this, it should be a temporary solution only.

Following is a summary of the migration considerations.

taglib-location Setting Consider the following `taglib` definition in `web.xml`:

```
<taglib>
   <taglib-uri>\hello</taglib-uri>
   <taglib-location>WEB-INF\lib\taglib.tld</taglib-location>
</taglib>
```

Note there is no opening "`\`" in the `taglib-location` setting. The OC4J JSP container in release 9.0.2 resolves this to the following:

```
\WEB-INF\WEB-INF\lib\taglib.tld
```

This is compliant with the JSP specification. The Orion JSP container resolves it to the following (which was presumably the intent of the developer):

```
\WEB-INF\lib\taglib.tld
```

It is advisable to change the `taglib-location` setting to the following (adding the opening "`\`"):

```
<taglib-location>\WEB-INF\lib\taglib.tld</taglib-location>
```

HTML Comments The Orion JSP container sometimes ignored the content of HTML comments, denoted by `<!-- ... -->` (as opposed to `<%-- ... --%>` for JSP comments). As a result, the content within the HTML comment was not output.

The OC4J JSP container does not interpret HTML comments, but also does not ignore them. The container passes them through to the browser, which is behavior that complies with the JSP specification. This makes it feasible for developers to add JavaScript to an HTML comment, for example.

Use of an Include Directive to Include a Page with an Unclosed Tag The JSP specification does not specify whether an `include` directive should accept a page header without a proper closing tag. The Orion JSP container would accept such a situation, but the OC4J JSP engine and the Tomcat reference implementation do not. Consider the following example:

```
-------------------------
a.jsp
<jsp:useBean id="b" class="pkgA.BeanB" >
<% // init the bean %>
-------------------------

-------------------------
b.jsp
<%@ include file="a.jsp" %>
</jsp:useBean>
<%= new java.util.Date() %>
-------------------------
```

This would be accepted by the Orion JSP container. To migrate this to the OC4J JSP container, modify the syntax as follows:

```
-------------------------
a.jsp
<jsp:useBean id="b" class="pkgA.BeanB" >
<% // init the bean %>
</jsp:useBean>
-------------------------

-------------------------
b.jsp
<%@ include file="a.jsp" %>
<%= new java.util.Date() %>
-------------------------
```

Include Directive Syntax The Orion JSP container accepted the following incorrect syntax:

```
<%@ include file="value" />
```

This does not follow the specification and is not accepted by the OC4J JSP container. The correct Include directive syntax is:

```
<%@ include file="value" %>
```

Quotes in Tag Attribute Settings According to the JSP 1.2 specification, tag attribute settings must always be quoted. Quotes within a setting must use an escape character. This was not clarified in previous JSP specifications, and the Orion JSP container accepted settings that were not properly quoted.

The following is incorrect, but was accepted by the Orion container:

```
<jsp:tag prop=<%=bean.getProperty("name")%> />
```

The following is correct and is now required by the OC4J JSP container; note the additional quotes and escapes:

```
<jsp:tag prop="<%=bean.getProperty(\"name\")%>"/>
```

Application Environment and Related Considerations

The servlet 2.0 specification did not have a clearly defined concept of a Web application and there was no defined relationship between servlet contexts and applications, as there is in later servlet specifications. In a servlet 2.0 environment such as JServ, there is only one servlet context object per JVM. A servlet 2.0 environment also has only one session object.

Oracle JSP implementations, however, have offered the use of `globals.jsa` files (a non-standard Oracle extension) to provide support for multiple applications and multiple sessions in a Web server, particularly for use in a servlet 2.0 environment. Where a distinct servlet context object would not otherwise be available for each application, the presence of a `globals.jsa` file for an application allowed the Oracle JSP container to provide the application with a distinct `ServletContext` object.

Because OC4J in Oracle9*i*AS Release 2 (9.0.2) offers a servlet 2.3 environment with standard application support, use of `globals.jsa` is no longer supported. (See "Migration from globals.jsa" on page 3-6.)

The rest of this section discusses `globals.jsa` considerations and other issues related to the application environment:

- migration away from `globals.jsa`
- classpath considerations
- changes in the Oracle JSP scope listener

Migration from globals.jsa

The OC4J JSP container in release 9.0.2 no longer supports `globals.jsa`. If an existing application uses `globals.jsa`, you should migrate away from this usage. The following substitutions for `globals.jsa` functionality are recommended:

- Instead of using `globals.jsa` as an application marker, use standard WAR packaging to denote the application structure.

- Instead of using `globals.jsa` start-session, end-session, start-application, and end-application events, use standard servlet 2.3 listener functionality. For example, equivalent capabilities are offered through the standard `javax.servlet.ServletContextListener` and `javax.servlet.http.HttpSessionListener` interfaces.
- Instead of using `globals.jsa` for global variable declarations, make the declarations in a single source file and use "global include" functionality of the OC4J JSP engine, introduced in release 9.0.2. See "JSP Global Includes" on page 3-9.

If you cannot migrate your code immediately, an application that uses `globals.jsa` can still run in OC4J if you use the previous `oracle.jsp.JspServlet` front-end servlet instead of the new `oracle.jsp.runtimev2.JspServlet` front-end. You can specify this in the `<servlet>` element in the application `web.xml` file, which overrides definitions in the OC4J `global-web-application.xml` file. This should be for short-term use only, however, given that the new front-end servlet supports many new features and configuration parameters and offers improved performance. (See "JSP Configuration" on page 3-13.)

Classpath Functionality

The OC4J JSP container in Oracle9*i*AS Release 2 (9.0.2) uses standard locations on the Web server in searching for translated JSP pages and any `.class` files and `.jar` files that they require. The container will find files in these locations without any Web server classpath configuration, and has the ability to automatically reload classes in these locations (depending on configuration settings).

The locations for dependency classes are as follows and are relative to the application root:

```
\WEB-INF\classes\...
/WEB-INF/lib
```

The location for JSP page implementation classes (translated pages) is as follows:

```
...\_pages\...
```

The `\WEB-INF\classes` directory is for individual Java `.class` files. These classes should be stored in subdirectories under the `classes` directory according to Java package naming conventions. For example, consider a JavaBean called `LottoBean` whose code defines it to be in the `oracle.jsp.sample.lottery`

package. The JSP container will look for `LottoBean.class` in the following location relative to the application root:

```
\WEB-INF\classes\oracle\jsp\sample\lottery\LottoBean.class
```

The `lib` directory is for `.jar` files. Because the Java package structure is specified in the `.jar` file structure, the `.jar` files are all directly in the `lib` directory, not in subdirectories. As an example, `LottoBean.class` might be stored in `lottery.jar`, located relative to the application root:

```
\WEB-INF\lib\lottery.jar
```

The `_pages` directory is under the following directory in OC4J:

```
\j2ee\home\application-deployments\app-name\web-app-name\temp
```

The `app-name` is determined through the `application` tag in the OC4J `server.xml` file; the `web-app-name`, which corresponds to the WAR file name, is mapped to the `app-name` through the `web-app` tag in the OC4J `default-web-site.xml` file. See the *Oracle9iAS Containers for J2EE User's Guide* and the *Oracle9iAS Containers for J2EE Servlet Developer's Guide* for information.

Generated page implementation classes for translated JSP pages are placed in subdirectories under the `_pages` directory according to the locations of the original `.jsp` files.

Important: Implementation details, such as the location of the `_pages` directory, are subject to change in future releases.

Migration of JspScopeListener Functionality

The Oracle `JspScopeListener` interface has been ported to OC4J to track page-scope, request-scope, session-scope, and application-scope events. To conform with servlet 2.3 standards, however, there are changes from how this mechanism was used in previous releases.

For page-scope objects, no special steps or configuration are necessary if you use the OC4J JSP container. There is an Oracle-specific runtime implementation to support `page` scope. If you want to migrate to another JSP environment, however, you can use a custom tag, `checkPageScope`, to support `page` scope.

If you use `JspScopeListener` for session-scope events, you should now implement the standard `HttpSessionBindingListener` interface as well as the `JspScopeListener` interface. This is necessary because the servlet 2.3 standard

uses the servlet container instead of the JSP container to provide notification for session-based events. Delegate the `valueUnbound()` method of `HttpSessionBindingListener` to a common method shared by the `outOfScope()` method of the `JspScopeListener` interface.

`JspScopeListener` now supports request-scope objects through a servlet filter. The filtering applies to any servlets matching a specified URL pattern. For event-handling for request-scope objects, add an entry such as the following to the `web.xml` file for your application. To ensure proper operation of the `JspScopeListener` functionality, this setting must be *after* any other `filter` settings.

```
<filter>
   <filter-name>Request Filter</filter-name>
   <filter-class>oracle.jsp.event.impl.RequestScopeFilter</filter-class>
</filter>
<!-- Define filter mappings for the defined filters -->
<filter-mapping>
   <filter-name>Request Filter</filter-name>
   <url-pattern>/jsp/*</url-pattern>
</filter-mapping>
```

`JspScopeListener` now supports application-scope objects through a servlet context listener implementation class, in accordance with the servlet 2.3 specification. For event-handling for application-scope objects, add an entry such as the following to the `web.xml` file for your application. To ensure proper operation of the `JspScopeListener` functionality, this setting must be *after* any other `listener` settings.

```
<listener>
   <listener-class>oracle.jsp.event.impl.AppScopeListener</listener-class>
</listener>
```

For additional information and examples, see the *Oracle9iAS Containers for J2EE JSP Tag Libraries and Utilities Reference*.

JSP Global Includes

In Oracle9*i*AS Release 2 (9.0.2), the OC4J JSP container introduces a feature called *global includes*. You can use this feature to specify one or more files to statically include into JSP pages in (or under) a specified directory, through virtual JSP `include` directives. During translation, the JSP container looks for a configuration file, `/WEB-INF/ojsp-global-include.xml`, that specifies the included files and the directories for the pages.

This enhancement is particularly useful in migrating applications that had used `globals.jsa` or `translate_params` functionality in previous Oracle JSP releases.

Globally included files can be used for the following, for example:

- global bean declarations (formerly supported through `globals.jsa`)
- common page headers or footers
- `translate_params` equivalent code

The ojsp-global-include.xml File

The `ojsp-global-include.xml` file specifies the names of files to include, whether they should be included at the tops or bottoms of JSP pages, and the locations of JSP pages to which the global includes should apply. This section describes the elements of `ojsp-global-include.xml`.

<ojsp-global-include>

This is the root element of the `ojsp-global-include.xml` file. It has no attributes.

Subelements:

<include>

<include ... >

Use this subelement of <ojsp-global-include> to specify a file to be included, and whether it should be included at the top or bottom of JSP pages.

Subelements:

<into>

Attributes:

- `file`: Specify the file to be included, such as `"\header.html"` or `"\WEB-INF\globalbeandeclarations.jsph"`. The file name must start with a backslash ("\"). In other words, it must be context-relative, not page-relative.
- `position`: Specify whether the file is to be included at the top or bottom of JSP pages. Supported values are `"top"` (default) and `"bottom"`.

<into ... >

Use this subelement of `<include>` to specify a location (a directory, and possibly subdirectories) of JSP pages into which the specified file is to be included. This element has no subelements.

The attributes of the info element are described below:

- `directory` - Specify a directory. Any JSP pages in this directory, and optionally its subdirectories, will statically include the file specified in the `file` attribute of the `<include>` element. The `directory` setting must start with a backslash (\), such as `"\dir1"`. The setting can also include a backslash after the directory name, such as `"\dir1\"`, or a backslash will be appended internally during translation.
- `subdir` - Use this to specify whether JSP pages in all subdirectories of the `directory` should also have the file statically include. Supported values are "`true`" (the default) and "`false`".

Global Include Examples

This section provides examples of global includes.

Example: Header/Footer

Assume the following `ojsp-global-include.xml` file:

```
<?xml version="1.0" standalone='yes'?>
<!DOCTYPE ojsp-global-include SYSTEM 'ojsp-global-include.dtd'>

<ojsp-global-include>
  <include file="\header.html">
     <into directory="\dir1" />
  </include>
  <include file="\footer1.html" position="bottom">
     <into directory="\dir1" subdir="false" />
     <into directory="\dir1\part1\" subdir="false" />
  </include>
  <include file="\footer2.html" position="bottom">
     <into directory="\dir1\part2\" subdir="false" />
  </include>
</ojsp-global-include>
```

This example accomplishes three objectives:

- The `header.html` file is included at the top of any JSP page in or under the `dir1` directory. The result would be the same as if each `.jsp` file in or under this directory had the following `include` directive at the top of the page:

  ```
  <%@ include file="\header.html" %>
  ```

- The `footer1.html` file is included at the bottom of any JSP page in the `dir1` directory or its `part1` subdirectory. The result would be the same as if each `.jsp` file in those directories had the following `include` directive at the bottom of the page:

  ```
  <%@ include file="\footer1.html" %>
  ```

- The `footer2.html` file is included at the bottom of any JSP page in the `part2` subdirectory of `dir1`. The result would be the same as if each `.jsp` file in that directory had the following `include` directive at the bottom of the page:

  ```
  <%@ include file="\footer2.html" %>
  ```

Note: If multiple header or multiple footer files are included into a single JSP page, the order of inclusion is according to the order of `<include>` elements in the `ojsp-global-include.xml` file.

Example: translate_params Equivalent Code

Assume the following `ojsp-global-include.xml` file:

```
<?xml version="1.0" standalone='yes'?>
<!DOCTYPE ojsp-global-include SYSTEM 'ojsp-global-include.dtd'>

<ojsp-global-include>
  <include file="\WEB-INF\nls\params.jsf">
     <into directory="/" />
  </include>
</ojsp-global-include>
```

And assume `params.jsf` contains the following:

```
<% request.setCharacterEncoding(response.getCharacterEncoding(); %>
```

The `params.jsf` file is included at the top of any JSP page in or under the application root directory. In other words, it is included in any JSP page in the

application. The result would be the same as if each `.jsp` file in or under this directory had the following `include` directive at the top of the page:

```
<%@ include file="\WEB-INF\nls\parms.jsf" %>
```

Also see "Global Includes for translate_params Migration" on page 3-15.

JSP Configuration

In OC4J in Oracle9*i*AS Release 2 (9.0.2), the `oracle.jsp.runtimev2.JspServlet` front-end servlet replaces the `oracle.jsp.JspServlet` front-end used for the JServ component of Oracle9*i*AS.

Mapping of this class as the JSP servlet is handled automatically in the OC4J `global-web-application.xml` file, as in the following entry:

```
<servlet>
   <servlet-name>jsp</servlet-name>
   <servlet-class>oracle.jsp.runtimev2.JspServlet</servlet-class>
   ...
      init params
   ...
</servlet>
```

This file also includes `<servlet-mapping>` elements where file name extensions (`.jsp`, `.JSP`, `.sqljsp`, `.SQLJSP`) are mapped to this front-end servlet.

Configuration parameters ("init params", see above) can also be set in `global-web-application.xml`, as in the following example:

```
   <init-param>
      <param-name>precompile_check</param-name>
      <param-value>true</param-value>
   </init-param>
```

Support for Previous Oracle JSP Configuration Parameters

Table 3–1 summarizes JSP configuration parameters supported in the previous front-end servlet for the JServ environment, and notes which are still relevant and supported for OC4J in Oracle9*i*AS Release 2 (9.0.2).

Table 3–1 Configuration Parameter Support: JServ to OC4J

Supported Config Params in JServ	Supported Config Params in OC4J	Comments
alias_translation		Unnecessary in OC4J.
bypass_source		Migrated to new OC4J param (`main_mode`).
classpath		Unnecessary in OC4J.
debug_mode	debug_mode	No change.
developer_mode		Migrated to new OC4J param (`main_mode`).
emit_debuginfo	emit_debuginfo	No change.
external_resource	external_resource	No change.
javaccmd	javaccmd	No change.
send_error		Unnecessary in OC4J.
session_sharing		Unnecessary in OC4J.
sqljcmd	sqljcmd	No change.
translate_params		Unnecessary in OC4J; use standard servlet. request `setCharacterEncoding()` method. Note: See "Global Includes for translate_params Migration" on page 3-15.
unsafe_reload		Unnecessary in OC4J.

New Oracle JSP Configuration Parameters

The following JSP configuration parameters are added for Oracle9*i*AS Release 2 (9.0.2):

- `main_mode`—This determines whether classes are automatically reloaded or JSP pages are automatically recompiled, in case of changes. Possible settings are `justrun`, `reload`, and `recompile`.
- `old_include_from_top`—Set this boolean to `true` for page locations in nested `include` directives to be relative to the top-level page, for backwards compatibility with Oracle JSP behavior prior to Oracle9*i*AS Release 2 (9.0.2).
- `precompile_check`—Set this boolean to `true` to check the HTTP request for a standard `jsp_precompile` setting.

- `reduce_tag_code`—Set this boolean to `true` for further reduction in the size of generated code for custom tag usage.
- `req_time_introspection`—Set this boolean to `true` to enable request-time JavaBean introspection if compile-time introspection is not possible. If compile-time introspection is possible and succeeds, this parameter is ignored and there is no request-time introspection.
- `static_text_in_chars`—Set this boolean to `true` to instruct the JSP translator to generate static text in JSP pages as characters instead of bytes. Also see "Static Text as Characters" on page 3-19.
- `tags_reuse_default`—This specifies a default setting for JSP tag handler pooling (true to enable by default; false to disable by default). This default setting can be overridden for any particular JSP page. Also see "Tag Handler Reuse" on page 3-19.
- `xml_validate`—Set this boolean to specify whether XML validation is to be performed on the `web.xml` file and TLD files.

> **See Also:** *Oracle9iAS Containers for J2EE Support for JavaServer Pages Reference* for additional information.

Global Includes for translate_params Migration

The new global includes functionality in the OC4J JSP container in Oracle9*i*AS Release 2 (9.0.2), described in "JSP Global Includes" on page 3-9, is useful in migrating applications that have previously used `translate_params` for globalization.

In this case, the globally included file can consist of a scriptlet similar to one of the following to achieve functionality that is equivalent to that of `translate_params`:

- Hardcode the request character set:

```
<% request.setCharacterEncoding("desired_charset"); %>
```

or:

- Use the character set of the response as the character set of the request, where the character set of the response is specified in the `contentType` attribute of a JSP `page` directive:

```
<% request.setCharacterEncoding(response.getCharacterEncoding()); %>
```

or:

- Use the character set of the response as the character set of the request, where the character set of the response is determined dynamically by Java logic:

```
<% String yourCharSet = yourLogicToDetermineCharset();
   response.setContentType("text/html; charset="+yourCharSet);
   request.setCharacterEncoding(response.getCharcterEncoding());
   // NOTE: The relative ordering of response.setContentType()
   // and request.setCharacterEncoding() is important.
%>
```

Possible Issues with the ojspc Utility

There are a few relatively minor migration considerations regarding the `ojspc` pre-translation utility in Oracle9*i*AS Release 2 (9.0.2).

Running ojspc for the OC4J Environment

The `ojspc` front-end script that sets up the classpath for pre-translation has been modified for the OC4J environment. Most users running `ojspc` for OC4J should not encounter problems using the new `ojspc` defaults; however, there are two potential issues to consider:

- If your application relies on the pre-JSP 1.2 behavior of the `include` directive, you can set `-oldIncludeFromTop=true` for compatibility with the previous behavior. This `ojspc` option has the same functionality as the new `old_include_from_top` JSP configuration parameter.
- In OC4J, static text is now generated in bytes by default, whereas it was generated in characters in previous releases (in the JServ environment). You can set `-staticTextInChars=true` if you want the old behavior. This `ojspc` option has the same functionality as the new `static_text_in_chars` JSP configuration parameter.

See "New Oracle JSP Configuration Parameters" on page 3-14 for information about the `old_include_from_top` and `static_text_in_chars` parameters.

Running ojspc_jserv for the JServ Environment

If you want to use `ojspc` for pre-translation for the JServ environment in Oracle9*i*AS Release 2 (9.0.2), use the new `ojspc_jserv.bat` script instead of the `ojspc.bat` script.

Packaging and Deployment

In previous Oracle9*i*AS releases, one deployment mechanism was through the standard servlet 2.2 WAR (Web archive) file. In Oracle9*i*AS Release 2 (9.0.2), deployment is through the standard J2EE 1.2 EAR (Enterprise archive) file, with the WAR file included inside the EAR file.

For OC4J, deploy each application through a standard EAR file. Specify the name of the application and the name and location of the EAR file through an `application` tag in the OC4J `\j2ee\home\config\server.xml` file.

You can accomplish this by using Oracle9*i*AS Enterprise Manager for deployment.

The EAR file includes the following:

- a standard `application.xml` configuration file, in `\META-INF`
- (optionally) an `orion-application.xml` configuration file, in `\META-INF`
- a standard WAR file

The WAR file includes the following:

- a standard `web.xml` configuration file, in `\WEB-INF`
- optionally an `orion-web.xml` configuration file, in `\WEB-INF`
- all JSP pages and Java classes (servlets, JavaBeans, and other classes) necessary to run the application, under `\WEB-INF\classes` and in JAR files in `\WEB-INF\lib`

Place the EAR file in the OC4J applications directory, which is specified in the `application-directory` setting in the `application-server` tag of the `server.xml` file, and is typically the `\j2ee\home\applications` directory. This would be the same directory as is specified for the EAR file location in the `application` tag in `server.xml`.

Through the OC4J auto-deployment feature, a new EAR file in the applications directory (as specified in `server.xml`) is detected automatically and is hierarchically extracted into the OC4J application deployment directory, according to the `deployment-directory` setting in the `application-server` tag of the `server.xml` file. This is typically the `\j2ee\home\application-deployments` directory.

Other Considerations

This section discusses the following additional considerations:

- Globalization Considerations for HTTP Parameters
- Tag Handler Reuse
- Session Key Seed Generation

Globalization Considerations for HTTP Parameters

Oracle9*i*AS release 9.0.2 includes new globalization functionality.

The setCharacterEncoding() Method

Effective with the servlet 2.3 specification, the `setCharacterEncoding()` method is available in the `javax.servlet.ServletRequest` class as the standard mechanism for specifying a non-default character encoding for reading HTTP requests. The signature of this method is as follows:

```
void setCharacterEncoding(java.lang.String enc)
                          throws java.io.UnsupportedEncodingException
```

The `enc` parameter is a string specifying the name of the desired character encoding, and overrides the default character encoding. Call this method before reading request parameters or reading input through the `getReader()` method (also of the `ServletRequest` class).

There is also a corresponding getter method:

```
String getCharacterEncoding()
```

In previous Oracle9*i*AS releases, which used pre-2.3 servlet environments, the `setCharacterEncoding()` method was not available. For such environments, particularly the JServ servlet 2.0 environment, Oracle's JSP implementation provided two alternative mechanisms as non-standard extensions:

- `oracle.jsp.util.PublicUtil.setReqCharacterEncoding()` method (preferred)
- `translate_params` configuration parameter (or equivalent code)

You should now migrate to the standard `setCharacterEncoding()` mechanism when you use OC4J.

Static Text as Characters

In JServ, which was the primary servlet environment under previous Oracle9*i*AS releases, static text is output in character format. In OC4J, static text is output as bytes by default, for faster throughput.

Some globalization functionality and flexibility is unavailable if static text is generated as bytes. For this reason, the OC4J JSP container supports the configuration parameter `static_text_in_chars` if you want to revert to character format for any reason.

Enable this flag, for example, if your application requires the ability to change the character encoding dynamically during runtime, such as in the following example:

```
<% response.setContentType("text/html; charset=UTF-8"); %>
```

Proper Handling of jsp:param Settings

Consider the following `jsp:include` tag and nested `jsp:param` tag:

```
<jsp:include page="..." >
   <jsp:param name="..." value="..." />
</jsp:include>
```

With the OC4J JSP container in Oracle9*i*AS Release 2 (9.0.2), there is no need to manually encode the `name` and `value` settings for the `jsp:param` tag. Just use the appropriate Java strings; encoding is handled automatically. This was *not* the case with the Orion JSP container in Oracle9*i*AS Release 1 (1.0.2.2.x) (manual encoding was required).

Tag Handler Reuse

In OC4J with Oracle9*i*AS Release 2 (9.0.2), you can specify whether JSP tag handler instances are pooled in a particular JSP page (always in the `application` scope) by setting the `oracle.jsp.tags.reuse` attribute in the JSP page context. Set it to `true` to enable pooling, or to `false` to disable pooling. For example:

```
pageContext.setAttribute("oracle.jsp.tags.reuse", new Boolean(true));
```

You can use separate settings in different pages, or even in different sections of the same page.

The default is according to the setting of the `tags_reuse_default` JSP configuration parameter. This default is `true` in OC4J, but `false` in JServ.

Session Key Seed Generation

OC4J in Oracle9*i*AS Release 2 (9.0.2) has a more secure session key seed generation process. When the first `HttpSession` object is created in an OC4J instance, there are a number of threads created to generate the session key seed. Therefore, users experience a longer delay during the first compilation or serving of a JSP page that uses session objects, compared to when using the OC4J in Oracle9*i*AS Release 1 (1.0.2.2.x). After the seed is generated, the process of compiling and serving JSP pages is as fast as before.

Migrating JServ to OC4J

This section covers the major considerations for migrating servlet applications from Oracle9*i*AS Release 1 (1.0.2.2.x) to Oracle9*i*AS Release 2 (9.0.2). This involves adapting from the Apache JServ servlet 2.0 environment to the Oracle9*i*AS Containers for J2EE (OC4J) servlet 2.3 environment.

Included is information about application environment and servlet zones, class and file locations, mount points, servlet aliases, initialization parameters, environment and JVM settings, prestarting, class loading, logging, servlet sessions, load balancing, and fault tolerance. The chapter is organized as follows:

- Web Application Environment
- Application Structure, File Location, and Deployment
- Servlet and Environment Setup
- Servlet Runtime Considerations
- Load Balancing and Fault Tolerance

Most sections present information for JServ as a reference, followed by the corresponding information for OC4J.

For more information about topics in this chapter, refer to the *Oracle9iAS Containers for J2EE Servlet Developer's Guide* and the *Oracle9iAS Containers for J2EE User's Guide*.

Note: It is assumed that you have some prior knowledge of the Sun Microsystems *Java Servlet Specification*, versions 2.2 and 2.3, including WAR (Web archive) files, EAR (enterprise archive) files, and `web.xml`. It is also helpful to have some prior knowledge of OC4J configuration files.

Web Application Environment

In Oracle9*i*AS Release 1 (1.0.2.2.x), Apache JServ, a servlet 2.0 environment, was the primary servlet environment. The servlet 2.0 specification did not have a clearly defined concept of a Web application and there was no defined relationship between servlet contexts and Web applications, as there is in later servlet specifications. In a servlet 2.0 environment such as JServ, there is only one servlet context object per JVM. A servlet 2.0 environment also has only one session object.

In Oracle9*i*AS Release 2 (9.0.2), OC4J is the primary application environment. OC4J includes a servlet 2.3 container with standard Web application support.

The rest of this section covers the following topics:

- Servlet Context and Servlet Path Mapping
- Change in Root Context of Default Web Application
- Servlet Zones Versus Web Applications

Servlet Context and Servlet Path Mapping

The servlet 2.2 and 2.3 specifications provide for each Web application to have its own servlet context, unlike in the servlet 2.0 JServ environment. This section reviews the servlet 2.2 and 2.3 functionality.

Each servlet context is associated with a directory path in the server file system that is the base path for modules of the Web application. This is the *application root*. Each Web application has its own application root. For a Web application in a servlet 2.2 or 2.3 environment, servlets, JSP pages, and static files such as HTML files are all based out of this application root. By contrast, in servlet 2.0 environments the application root for servlets and JSP pages is distinct from the doc root for static files.

Note that a URL for a servlet has the following general form:

`http://host[:port]/contextpath/servletpath`

When a servlet context is created, a mapping is specified between the application root and the *context path* portion of a URL. The *servlet path* is defined in the application `web.xml` file—the `<servlet>` tag within `web.xml` associates a servlet class with a servlet name, and the `<servlet-mapping>` tag within `web.xml` associates a URL pattern with a servlet name. When a request reaches a Web application, the servlet container will compare the path in the request with known URL patterns defined in `web.xml`, and invoke the servlet according to a matched

URL pattern. See the *Oracle9iAS Containers for J2EE Servlet Developer's Guide* for more information.

For example, consider an application with the application root `\home\dir\mybankapp\mybankwebapp`, which is mapped to the context path `\mybank`. Further assume the application includes a servlet whose servlet path is `loginservlet`. This servlet can be invoked as follows:

```
http://host[:port]/mybank/loginservlet
```

The application root directory name itself is not visible to the end-user.

To continue this example for an HTML page in this application, the following URL points to the file `\home\dir\mybankapp\mybankwebapp\dir1\abc.html`:

```
http://host[:port]/mybank/dir1/abc.html
```

For each servlet environment, there is also a *default* servlet context. For this context, the context path is simply "\", which is mapped to the default servlet context application root. For example, assume the application root for the default context is `\home\dir\defaultapp\defaultwebapp`, and a servlet with the servlet path `myservlet` uses the default context. Its URL would be as follows:

```
http://host[:port]/myservlet
```

The default context is also used if there is no match for the context path specified in a URL.

Continuing this example for an HTML file, the following URL points to the file `\home\dir\defaultapp\defaultwebapp\dir2\def.html`:

```
http://host[:port]/dir2/def.html
```

Change in Root Context of Default Web Application

In Oracle9*i*AS release 9.0.2, the root context of the default Web application is "\j2ee". This is a change from release 1.0.2.2, where the root context was "\", and affects anything deployed in the default Web application. This includes the samples and demos provided with OC4J.

Consider a servlet in the default Web application that was invoked as follows in release 1.0.2.2:

```
http://host[:port]/servlet/myservlet
```

In release 9.0.2, invoke it as follows:

```
http://host[:port]/j2ee/servlet/myservlet
```

Servlet Zones Versus Web Applications

JServ has the concept of servlet zones, somewhat comparable to the Web application concept in servlet 2.2 and higher specifications. This section compares basic zone setup in JServ to basic application setup in OC4J.

Basics of JServ Zone Specification

JServ uses *servlet zones*, where a servlet zone is somewhat similar in concept to a Web application. The use of zones helps developers separate the overall JServ environment into separate groups of servlets, according to conditions such as work load, usage, and security privileges. In JServ, servlets are grouped and managed based on servlet zones, not based on the servlet container itself. It is mandatory to have at least one servlet zone.

Servlet zones are specified in the `jserv.properties` file. In addition, each zone has its own configuration file, known as a "zone properties file", typically with a naming convention such as `zone`*`xxx`*`.properties`.

Here is an example of zone settings in `jserv.properties`:

```
zones = zone1,zone2
```

The locations of the corresponding zone properties files are also specified in `jserv.properties`, as in the following example:

```
zone1.properties =\servlet\zone1\zone1.properties
zone2.properties =\servlet2\zone2\zone2.properties
```

Basics of OC4J Application Specification

In OC4J, Web applications can be considered as equivalent to zones. It is also true that in OC4J, servlets are grouped and managed on a per-application basis. Additionally, OC4J has the concept of a default *global application* that is the parent of all applications and also defines a default Web application.

In OC4J, deploy applications using EAR files. When you deploy an EAR file, you must configure at least the following files:

- `default-web-site.xml`

- `web.xml`

You can also optionally use an `orion-web.xml` file for configuration of application-specific OC4J features. The `web.xml` and `orion-web.xml` files are typically part of the WAR file.

Following is an overview of key OC4J configuration files for Web applications.

These are global files for all OC4J applications, typically in the OC4J `\j2ee\home\config` directory:

- `default-web-site.xml`—This includes a `<web-app>` element for each Web application for the default Web site, mapping the application name to the "Web application name". The Web application name corresponds to the WAR deployment file name. Additional Web site XML files, as specified for additional Web sites in the `server.xml` file, have the same functionality.
- `global-web-application.xml`—This is a global configuration file for OC4J Web applications, establishing default configurations, and including setup and configuration of the JSP front-end servlet, `JspServlet`.
- `application.xml`—This is a global configuration file for OC4J J2EE applications.
- `data-sources.xml`—This specifies data sources for database connections.

In addition to the global `application.xml` file, there is a standard `application.xml` file, and optionally an `orion-application.xml` file, for each application. These files are in the application EAR file.

Also, in an application WAR file, which is inside the application EAR file, there is a standard `web.xml` file and optionally an `orion-web.xml` file. These are for application-specific and deployment-specific configuration settings, overriding `global-web-application.xml` settings or providing additional settings as appropriate. The `global-web-application.xml` and `orion-web.xml` files support the same elements, which is a superset of those supported by the `web.xml` file.

If the `orion-application.xml` and `orion-web.xml` files are not present in the archive files, they will be generated during initial deployment according to settings in the `global-web-application.xml` file.

> **See Also:** *Oracle9iAS Containers for J2EE User's Guide* and the *Oracle9iAS Containers for J2EE Servlet Developer's Guide*.

Application Structure, File Location, and Deployment

This section discusses the typical OC4J Web application structure, how to configure file locations in OC4J compared to JServ, and the basics of how to deploy an application. The section is organized as follows:

- JServ File Repositories
- OC4J Application Structure and File Locations
- OC4J Deployment

JServ File Repositories

The locations, or repositories, of servlets under a servlet zone are specified through `repositories` commands in the zone properties file. (See "Basics of JServ Zone Specification" on page 3-23 for information about zone properties files.) JServ loads classes from locations specified in repository entries. Here are some examples:

```
repositories=\private\mydir\net.jar
repositories=\private\mydir\tmp\net.zip
repositories=\private\mydir\applications
```

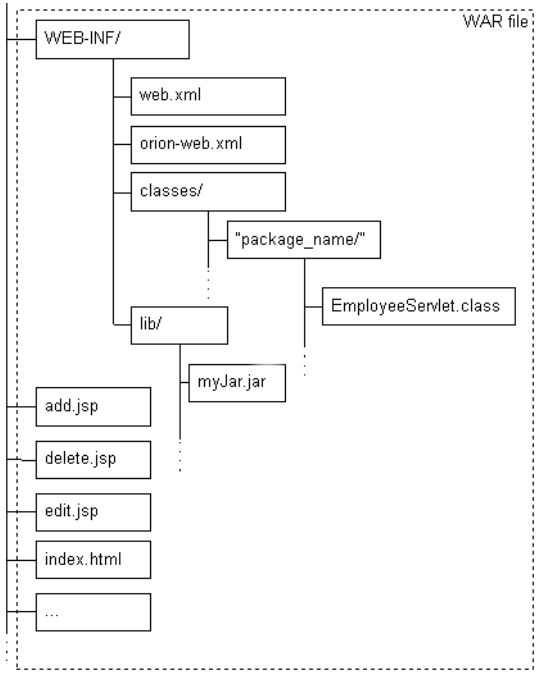
As shown, class files can be read directly from directories or from archive files (`.zip` or `.jar`). See "Class Loading in JServ" on page 3-36 for more information about repositories and class loading.

OC4J Application Structure and File Locations

Figure 3–1 shows the typical (and recommended) Web application structure under the application root directory. (Much of it applies generally to servlet 2.2 and 2.3 environments.)

In OC4J, the root directory is *`app-name/web-app-name`*. The application name is defined in the `server.xml` file, and mapped to a Web application name in the `default-web-site.xml` file or other Web site XML file.

Figure 3–1 OC4J Application Structure



According to this structure, put servlet classes under the *application_name**web-app-name*\WEB-INF\classes directory, in subdirectories according to package names as appropriate. For example, if you have `HelloWorldServlet` in the `examples` package, then the class file should be located as follows:

*application_name**web-app-name*\WEB-INF\classes\examples\HelloWorldServlet

Place HTML files, JSP pages, and other resource files in the application root directory.

Place required library files, such as JAR files, in the following directory:

*application_name**web-app-name*\WEB-INF\lib

As you can see, in OC4J as well as in JServ, class files can be read either directly from a directory or from an archive file (ZIP or JAR).

You can also specify a *codebase*, placing additional required files in a desired location, then adding that location to your classpath through the `<classpath>`

subelement of the <orion-web-app> element in the orion-web.xml file, as in the following example:

```
<classpath path="\private\test\test.jar" />
```

OC4J Deployment

With OC4J, you deploy applications using Oracle Enterprise Manager (OEM). Also note that you will typically use an EAR (Enterprise archive) file for OC4J deployment.

Consider the following example, creating an EAR file for a simple application with a servlet HelloServlet and a JSP page Hello.jsp:

1. Create and configure web.xml, and optionally orion-web.xml, to specify application-specific settings.
2. Create a WAR file, helloapp.war, from the application root using the JAR utility.

   The helloapp.war file contains the following:

   ```
   META-INF\MANIFEST.MF
   WEB-INF\
   WEB-INF\classes\
   WEB-INF\classes\HelloServlet.class
   WEB-INF\orion-web.xml
   WEB-INF\web.xml
   Hello.jsp
   ```

   (The META-INF/MANIFST.MF file is created by the JAR utility. There is no need to modify it.)
3. Create an EAR file, helloapp.ear, for deploying the application.

   The EAR file contains the following:

   ```
   META-INF\
   META-INF\MANIFEST.MF
   META-INF\application.xml
   META-INF\orion-application.xml
   helloapp.war
   ```

   The application.xml file is required when deploying an EAR file, but the orion-application.xml file is optional.

Using this helloapp.ear file, you can deploy the Web application using OEM.

Servlet and Environment Setup

This section covers a variety of servlet setup issues, reviewing configuration steps for JServ and providing the equivalent steps for OC4J. This includes the following topics:

- JVM Parameters and Environment Variables
- Mount Settings
- Servlet Aliases and URL Mapping
- Initialization Parameters

JVM Parameters and Environment Variables

This section describes how to set JVM and environment variables for OC4J, compared to how to do so for JServ.

Setting Environment Variables in JServ

In `jserv.properties`, you can set environment variables for the JVM to use, including Oracle environment settings. The following examples are for Oracle settings:

```
wrapper.env=LD_LIBRARY_PATH=...
wrapper.env=ORACLE_HOME=...
wrapper.env=ORACLE_SID=...
```

There are also settings for the Java and system environment:

- Use the following to set the classpath:

  ```
  wrapper.classpath=...
  ```

- Use the following to set the path:

  ```
  wrapper.path=...
  ```

- Use the following to set the full path of the Java interpreter (if it is not visible in the path):

  ```
  wrapper.bin=...
  ```

- Use `wrapper.bin.parameters` for JVM loading parameters, such as for heap or stack size. Following is an example:

  ```
  wrapper.bin.parameters=-Xms64m
  ```

- Use `wrapper.env.copy` or `wrapper.env.copyall` to copy environment parameters from the caller to the JVM. The following examples copy all environment parameters, and a particular environment parameter (`myparam`), respectively:

```
wrapper.env.copyall

wrapper.env.copy=myparam
```

In addition, JServ has the following security parameters: `security.maxConnections`, `security.allowedaddresses`, and `security.authentication`.

Setting Environment Variables in OC4J

The method used to set environment parameters for OC4J depends on how it is started.

Startup Through OPMN

If OC4J is started through OEM (more specifically, through the `opmn` process management module OPMN), then you can set parameters through the *`ORACLE_HOME`*`\opmn\conf\opmn.xml` configuration file. In particular, the following subelements of the `<oc4j>` element are of interest:

- Use `<java-bin>` to specify a path to the Java executable. If this element is not specified, the *`ORACLE_HOME`*`/jdk/bin/javai`
- *`ORACLE_HOME`*`\jdk\bin\java` is used by default.
- Use `<java-option>` to specify the command line parameters required by the JVM.

The `<java-bin>` subelement is equivalent to `wrapper.bin` in JServ. The `<java-option>` subelement is equivalent to `wrapper.bin.parameters` in JServ.

Here is an example (showing only the relevant portions of <oc4j> element syntax):

```
<oc4j numProcs="1"  maxRetry="4" ... >
      <java-bin path="\private\my-pc\jdk\bin\java" />
      <java-option value="-Xmls32m -Xmx64m -Xss128K
                          -Doracle.ons.oraclehome=\private\oracle" />
   ...
</oc4j>
```

You can also use the opmn.xml file to specify Oracle environment variables, through the <environment> element, such as in the following example:

```
<environment>
   <prop name="PATH" value="\private\home\ias\lib"/>
   <prop name="CLASSPATH" value="\private\home\ias\bin" />
   <prop name="LD_LIBRARY_PATH" value="\private\home\lib" />
</environment>
```

These settings specify the environment for the new process when it is spawned.

See the *Oracle9i Application Server Administrator's Guide* for information about OPMN.

Mount Settings

JServ (mod_jserv) and OC4J (mod_oc4j) each have "mount" commands to establish application root locations, or "mount points".

JServ Mount Settings

In JServ, mount points are used to define the root locations for different servlet zones. If the protocol, host, or port are not specified, then they are picked from default entries in the jserv.conf file, which is included into the httpd.conf file.

Consider the following sample mount commands:

```
ApJServMount /servlets /root
ApJServMount /servlets/admin ajpv11://myhost:9009/admin
```

Based on these commands, the following would be true, for example:

- The following URL, as a result of the first ApJServMount command, would request the servlet HelloWorldServlet in the servlet zone root:

  ```
  http://myhost.mycompany.com/servlets/HelloWorldServlet
  ```

- The following URL, as a result of the second `ApJServMount` command, would be handled through port 9009, using Apache JServ Protocol (AJP) version 1.1.

  ```
  http://myhost.mycompany.com/servlets/admin/HelloWorldServlet
  ```

OC4J Mount Settings

In the OC4J and Oracle HTTP Server environment, you can specify mount points through the *ORACLE_HOME*`\Apache\modoc4j\conf\mod_oc4j.conf` file. You can specify additional relevant Java-side settings, such as host and port, through the `default-web-site.xml` file.

Refer again to "JServ Mount Settings" above, for comparisons. Consider the following JServ example again:

```
ApJServMount /servlets/admin ajpv11://myhost:9009/admin
```

You can make equivalent protocol and host settings through the `<web-site>` element in `default-web-site.xml` (or some other Web site XML file, for a Web site other than the default), as follows:

```
<web-site port="9009" protocol="ajp13" ...>
...
</web-site>
```

Also note that within the `<web-site>` element, you can use `<web-app>` subelements to specify information about individual applications on the site, such as the corresponding Web application name and application root. Here is an example:

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

On the target host, `myhost`, OC4J would find the Web application according to application settings in `default-web-site.xml` (or some other Web site XML file).

Based on the preceding `Oc4jMount` command, the following request would be routed to an OC4J process that listens at `myhost` on port 9009 using Apache JServ Protocol (AJP) version 1.3:

```
http://myhost.mycompany.com/servlets/admin/HelloWorld
```

OC4J would find the application according to application settings in the Web site XML file on `myhost`.

Important: The host and port specified in the `Oc4jMount` command should be the same as the host and port specified for `ajp13` protocol in the `default-web-site.xml` file.

If OEM (the `opmn` process) starts OC4J, then in `mod_oc4j.conf` you can add the following, in which case `opmn` will scan all possible ports for a suitable and available AJP port to use. Requests of the form `/servlets/admin/*` will be directed to one of the OC4J JVMs in the default "home" OC4J instance.

```
Oc4jMount /servlets/admin/*
```

This is in conjunction with the following default settings in the `default-web-site.xml` file:

```
<web-site port="0" protocol="ajp13" ...>
...
</web-site>
```

You can also specify a particular OC4J instance, or load balancing between clusters, or load balancing between Oracle9*i*AS instances, as in the following examples:

```
Oc4jMount /servlets/admin/* oc4j_inst1
Oc4jMount /servlets/admin/*  cluster://ias_cluster_1:home,ias_cluster_2:home
Oc4jMount /servlets/admin/* instance://ias_inst_1:home_1,ias_inst_2:home_2
```

See Also: *Oracle HTTP Server Administration Guide* and the *Oracle9iAS Containers for J2EE Servlet Developer's Guide*

Servlet Aliases and URL Mapping

This section compares how to specify servlet aliases and URL mapping in JServ to the equivalent functionality in OC4J.

Aliases and URL Mapping in JServ

In JServ, servlet aliases are specified in the appropriate zone properties file. For example, for a servlet class `example.extensionmapping.InputServlet`, you can specify the alias `inputservlet` to avoid having to specify the full path when you invoke the servlet. This is done as follows:

```
servlet.inputservlet.code = example.extensionmapping.InputServlet
```

URL extension mappings, like mount points, are defined in the `jserv.conf` file. Following is an example:

```
ApJServAction .inp /servlets/example.extensionmapping.InputServlet
```

This results in URL patterns ending in ".inp" being mapped to `InputServlet`.

Also assume the following mount command:

```
ApJServMount /servlets /root
```

The following URL, because of this `ApJServMount` command and the `ApJServAction` command, is passed to `example.extensionmapping.InputServlet` in the servlet zone `root`.

```
http://myhost.mycompany.com/EmployeeInput.inp
```

Aliases and Extension Mapping in OC4J

In OC4J, specify servlet aliases and URL mapping through entries such as the following in the `global-web-application.xml` file:

```
<servlet>
   <servlet-name>inputservlet</servlet-name>
   <servlet-class>example.extensionmapping.InputServlet</servlet-class>
</servlet>
...
<servlet-mapping>
   <servlet-name>inputservlet</servlet-name>
   <url-pattern>/*.inp</url-pattern>
</servlet-mapping>
```

The servlet name (alias) can be anything—it simply serves as a reference name to associate the servlet class with the URL extension that is specified in the `<servlet-mapping>` element.

Initialization Parameters

This section details the differences between JServ and OC4J in how to set servlet initialization parameters.

Setting Initialization Parameters in JServ

JServ supports servlet-based initialization parameter settings ("initArgs") as well as zone-wide default parameter settings. These settings are specified in the appropriate zone properties file.

For example, for a servlet `foo1`, you would define a servlet-based setting for the `name` parameter as follows:

```
servlet.foo1.initArgs=name=scott
```

A zone-wide default setting, shared by all servlets in the zone, is specified as in the following example. This specifies a default setting for the `company` parameter:

```
servlets.default.initArgs=company=oracle
```

A servlet-based setting overrides a zone-wide (default) setting for the same named parameter.

Setting Initialization Parameters in OC4J

In OC4J, specify servlet-based initialization parameter settings through subelements of the `<servlet>` element in the standard `web.xml` file. The following example is equivalent to the example for `foo1` in the preceding section (but also specifies the servlet class):

```
<servlet>
   <servlet-name>foo</servlet-name>
   <servlet-class>FooServlet</servlet-class>
   <init-param>
        <param-name>name</param-name>
        <param-value>scott</param-value>
   </init-param>
</servlet>
```

There is no exact equivalent in OC4J for the concept of a zone-wide setting—there is no mechanism for application-wide default settings. However, `context-param` settings are conceptually similar. For each Web application, there is a servlet context. You can set attributes for the context in the application `web.xml` file, as in the following example.

```
<context-param>
    <param-name>company</param-name>
    <param-value>oracle</param-name>
</context-param>
```

You can access context parameters through standard servlet 2.3 methods of the `javax.servlet.ServletContext` class. The following example will return an enumerated list of the initialization parameter names for a `ServletContext` instance:

```
ServletContext ctx = getConfig().getServletContext();
ctx.getInitParameterNames();
```

The following call will obtain the value of the `company` parameter:

```
ctx.getInitParameter("company");
```

Servlet Runtime Considerations

This section covers migration considerations regarding servlet execution and class loading. The following topics are covered:

- Pre-Started Servlets
- Class Loaders and Automatic Class Reloading
- Session Tracking and Behavior
- Message and Error Logging

Pre-Started Servlets

In both JServ and OC4J, servlets can be pre-started—rather than having servlet instances created only after the first request arrives, they can be created in advance and pre-started by the servlet container when the container starts up. This reduces the time to service the first request.

Pre-Start and Timeout Settings in JServ

In JServ, servlets to pre-start are specified in the appropriate zone properties file, as in the following example:

```
servlets.startup=oracle.sample.test1.HelloWorld,foo1
```

This prestarts `HelloWorld` and `foo1`.

Alternatively, you can use an alias name instead of the complete name.

JServ zone properties files also support the following parameters, which have no equivalents in OC4J:

- Parameters to specify the timeout period for initialization (after which the servlet container will stop trying to initialize) and the timeout period after which a servlet is destroyed:

  ```
  init.timeout
  destroy.timeout
  ```

- Parameters for the single-threaded model—to specify the number of servlet instances to be created if the servlet implements the `javax.servlet.SingleThreadModel` interface:

  ```
  SingleThreadModelServlet.initialCapacity
  SingleThreadModelServlet.incrementCapacity
  SingleThreadModelServlet.maximumCapacity
  ```

Pre-Start Settings in OC4J

In OC4J, you can pre-start servlets by using the `<load-on-startup>` subelement of the `<servlet>` element in the application `web.xml` file, as in the following example:

```
<servlet>
   <servlet-name>HelloWorld</servlet-name>
   <servlet-class>oracle.sample.test1.HelloWorld</servlet-class>
   <load-on-startup/>
</servlet>
```

Class Loaders and Automatic Class Reloading

This section discusses servlet class loaders, and class reloading during servlet execution.

Class Loading in JServ

This section discusses class loaders and class reloading in the JServ environment.

There is a separate class loader for each of the following:

- system classes—classes found on the system classpath

  For automatic JServ startup, the system classpath is determined by the `wrapper.classpath` setting in the `jserv.properties` file. For manual JServ startup, the system classpath is determined by the `CLASSPATH` setting for the particular JServ instance.

Classes loaded from the system classpath (including servlet classes) cannot be automatically reloaded without restarting the server.

- zone classes—classes found on the zone classpath

  The zone classpath is specified through the `repositories` parameter in the zone properties file.

Be aware of the following:

- Classes loaded from the system classpath are shared across all zones in the same JVM.
- Each servlet zone has its own instance of a custom class loader, for classes in the zone classpath.
- Classes loaded from the zone classpath are not shared between zones. Furthermore, for a class that is available in different zones, its static variables cannot be shared across zones, even when the zones are in the same JVM. For static variables to be sharable, the class must be in the system classpath, and therefore loaded by the system class loader.

Classes in the zone classpath, and therefore loaded by the zone class loader, can be automatically reloaded if they are modified. This is useful if you are in the process of developing your application, and is determined by the following setting in the zone properties file:

```
autoreload.classes=true
```

Class Loading in OC4J

OC4J classpath and class loader configuration that is equivalent to that of JServ is determined as follows:

- For server-wide class loading, you can make classpath settings through the `<library>` element of the global `j2ee/home/config/application.xml` file, as in the following example:

  ```
  <library path="tmp\net.jar" />
  ```

  You can specify the relative or absolute path to directories, and these directories are scanned for JAR or ZIP files to include in the classpath at startup. By default, the `application.xml` file specifies the inclusion of files from the `j2ee/home/lib` directory. (A `<library>` element exists for this path in the default global `application.xml` file).

> **Note:** Do not confuse the global `application.xml` file with the `application.xml` file for each application.

- For application-based class loading, the default classpath consists of the directories `WEB-INF\classes` (for class files, in subdirectories according to package names) and`WEB-INF\lib` (for JAR and ZIP libraries).

  You can add to the application classpath through the `<classpath>` element of the `orion-web.xml` file, as in the following example:

  ```
  <classpath path="\private\test\test.jar" />
  ```

You can specify automatic recompilation and reloading of servlets in OC4J by setting the `development` attribute to `"true"` in the `<orion-web-app>` element of the `j2ee\home\config\global-web-application.xml` file or the application `orion-web.xml` file (which takes precedence over `global-web-application.xml` for a particular application).

In this case, classes in the target directory are automatically reloaded whenever they are modified, or whenever an application-level XML file is modified. This is useful when you are in the process of developing your application.

By default, when automatically reloading, source files (`.java`) are picked up from the target directory `WEB-INF\src` if it exists. If the `src` directory does not exist, then source files are picked up from the `WEB-INF\classes` directory instead. You can specify an alternative target directory through the `source-directory` attribute of the `<orion-web-app>` element in the application `orion-web.xml` file, as in the following abbreviated example:

```
<orion-web-app ... source-directory="\private\scott\myservletsource" ...>
   ...
</orion-web-app>
```

In this case, files are picked up from the specified directory only, not from the `src` or `classes` directory.

Session Tracking and Behavior

This section discusses session behavior and related configuration in JServ and OC4J.

Session Tracking and Behavior in JServ

JServ provides the following parameters in the zone properties file for specifying session behavior:

- `session.useCookies`—Specifies whether to use cookies for sessions (default is `true`). If `false`, then the `encodeUrl()` method of the response object is the only means of session-tracking.
- `session.timeout`—Specifies the number of milliseconds to wait before invalidating a session (default is 1800000, which is 30 minutes).
- `session.checkFrequency`—Specifies how frequently (in seconds) to check for timed-out sessions. (The default is 30.)

Session Tracking and Behavior in OC4J

For session tracking in OC4J, the servlet container will first attempt to accomplish tracking through cookies. If cookies are disabled, session tracking can be maintained only by using the `encodeURL()` method of the response object explicitly in the servlet. (The `encodeURL()` method replaces the servlet 2.0 `encodeUrl()` method, which has been deprecated.)

You can also specify the number of minutes to wait before a session is invalidated. (The default is 20.) Use the `<session-timeout>` subelement of the `<session-config>` element in the application `web.xml` file, as follows:

```
<session-config>
   <session-timeout>30</session-timeout>
</session-config>
```

Additionally, you can disable the use of session cookies with the following setting in the `global-web-application.xml` or `orion-web.xml` file:

```
<session-tracking cookies="disabled" ... >
   ...
</session-tracking>
```

(Cookies are enabled by default.) This is equivalent to the functionality of `session.useCookies` in JServ. As in JServ, with cookies disabled you must explicitly use the `encodeURL()` method of the response object for session-tracking.

Message and Error Logging

This section compares the JServ and OC4J mechanisms for setting up log files.

Message Logging for JServ

Message logging parameters such as the log file name, timestamp format, and other settings are specified through the `jserv.properties` file. This can help with debugging.

Following are the key parameters:

- `log`—Specifies whether or not to log messages (`log=true` by default).
- `log.file`—Specifies the file where log messages are written. (An absolute path name is recommended.)
- `log.timestamp`—Specifies that messages are to be preceded by a timestamp (`log.timestamp=true` by default).
- `log.dateFormat`—Specifies the date format in the timestamps. The default is "dd/MM/yyyy HH:mm:ss:SSS.zz".
- `log.queue.maxage`—Specifies the maximum time (in milliseconds) that a message can be in the queue.
- `log.queue.maxsize`—Specifies the maximum number of messages in the queue.

There are also the following parameters to enable different levels of logging information:

- `log.channel`
- `log.channel.info`
- `log.channel.ServletException`
- `log.channel.jservException`
- `log.channel.warning`
- `log.channel.servletLog`
- `log.channel.critical`
- `log.channel.debug`

Message Logging for OC4J

Setting up logging in OC4J is relatively simple, with few settings required. There are two different logs to consider.

Servlet Logging

In this document, the term "servlet logging" refers to the logging of servlet context information. This includes servlet exceptions and information logged through the `log()` method of the servlet context object.

You can specify a log file for servlet logging through the <file> subelement of the <log> element in the global j2ee\home\config\application.xml file, as in the following example:

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

For a particular application, you can override this setting in the application-specific orion-application.xml file. Refer to the *Oracle9iAS Containers for J2EE User's Guide* for information about creating this file.

Server Logging

There can also be a log file for server-wide information, such as notices of server startup, shutdown, and recovery, for example. Specify this log file name through the <file> subelement of the <log> element in the server.xml file, using the same syntax as for the application.xml file above.

Web Access Logging

In this document, the term "web access logging" refers to the tracking of HTTP-related information, such as the host, IP address, time, request URI, and HTTP response status code.

In the default-web-site.xml file or other Web site XML file, you can specify the log file through the <access-log> subelement of the <web-site> element, as in the following example:

```
<web-site ... >
...
   <access-log path="..\log\http-web-access.log" />
...
</web-site>
```

If the server is started through OEM (the opmn process), then logs are in the opmn\logs directory. Log file names are typically defined as follows:

ORACLE_HOME\opmn\logs*oc4j_instance_name*.default_island.proc

Alternatively, you can specify log files through the <log-file> element in the *ORACLE_HOME*\opmn\conf\opmn.xml file, as in the following example:

```
<log-file path="..\logdir\my-web-access.log" level="3" />
```

The level attribute specifies the severity level of the logging, from 1 to 6: 1=FATAL, 2=ERROR, 3=WARN, 4=NOTIFY, 5=DEBUG, 6=VERBOSE.

See Also: *Oracle9i Application Server Administrator's Guide* for information about `opmn`

Load Balancing and Fault Tolerance

This section describes issues and configuration for load balancing and fault tolerance in JServ and OC4J.

Request Routing and Load Balancing in JServ

JServ supports request routing by appending the appropriate JServ instance ID to the session ID when an `HttpSession` object is used.

When an HTTP request is received before a session is started, an arbitrary JServ instance is chosen from the available instances to service the request, and a cookie with the JServ instance ID is sent back to the Web browser (or other HTTP client). Later, when the next request comes from the same session, it is forwarded to the same JServ instance by matching the JServ ID. (If the original JServ instance is down, the request will be forwarded automatically to an alternative instance.)

Given this functionality, `HttpSession` objects in JServ are non-distributable—the session object cannot be distributed between different JServ instances. As a result, a long-lived HTTP session in JServ decreases the flexibility of load balancing. Also, the session data is lost if the corresponding JServ JVM crashes. Fault tolerance is low, because there is no session failover functionality.

Load Balancing and Fault Tolerance in OC4J

OC4J supports clusters of OC4J instances, and clusters can be customized to the specific needs of the users. Through OC4J *load balancing*, more user traffic can be handled by distributing the request workload to multiple servers within the cluster. The load balancer replicates the state, such as `HttpSession` data, of each individual node to the cluster. (The state information is not saved to any persistent storage, but is in memory.) Through OC4J *fault tolerance*, in case of the failure of a server, a client can automatically be redirected to an alternative server in the cluster.

In OC4J, HTTP sessions are replicated to other OC4J JVM instances within a load-balanced cluster island. This avoids loss of session state in case of JVM failure, and is a feature that is not available in JServ.

Assuming proper configuration, when the request is routed through another JVM in the cluster island, the session state is available in the other JVM as well. Furthermore, the session state is still available even in individual JVM failure scenarios. The Web application proceeds smoothly.

For this functionality, the Web application must be marked as "distributable" in the application `web.xml` file, through the `<distributable>` element. Objects in a distributable `HttpSession` instance must be serializable or remoteable for the replication to work properly.

See the *Oracle9i Application Server Performance Guide* for more information about OC4J load balancing and fault tolerance.

Example: JServ to OC4J Migration

This section provides a general example of migrating a Web application from Apache JServ to OC4J, noting both the original JServ configuration settings and the OC4J configuration settings. This example does not necessarily reflect a typical or optimal scenario; it is merely for illustrative purposes.

The example includes two servlets (source files `HelloWorldServlet.java` and `SessionServlet.java`), two JSP pages (`Hello.jsp` and `snoop.jsp`), and accompanying `.gif` and `index.html` files.

Setup in JServ

This section shows the original setup in JServ, prior to migration.

Assume the following directory structure:

```
\private\scott-pc\migration-example\
                    index.html

                    classes\
                             HelloWorldServlet.java
                             SessionServlet.java

                     jsps\
                             Hello.jsp
                             snoop.jsp

                    examples\
                             index.html

                             images\
                                    blk_line_bullet_35.gif
                                    red_arrow_bullet_35.gif
```

The following JServ configuration files would have related entries, as described in the subsections immediately below:

```
ORACLE_HOME\apache\conf\jserv\jserv.conf
                              jserv.properties
                              zone.properties
```

Setup in jserv.conf

In this example, the `jserv.conf` file includes the following entries:

```
Alias  \migdemos \private\scott-pc\migration-example
ApJServMount \servlet /root
```

Setup in jserv.properties

In this example, the `jserv.properties` file includes the following entries:

```
zones=root

# Configuration file for each servlet zone (one per servlet zone)
# Syntax: [servlet zone name as on the zones list].properties=
#                                       [full path to configFile] (String)
# Default: NONE
# Note: if the file could not be opened, try using absolute paths.
root.properties=\private\scott-pc\apache\conf\jserv\zone.properties
```

Setup in zone.properties

In this example, the `zone.properties` file includes the following entries:

```
# List of Repositories
##########################

# The list of servlet repositories controlled by this servlet zone
# Syntax: repositories=[repository],[repository]...
# Default: NONE
# Note: The classes you want to be reloaded upon modification should be put
# here.
repositories=\private\scott-pc\migration-example\classes
```

URLs for Invocation in JServ

Assuming the preceding configuration, you could use the following URLs (specifying the appropriate port) to directly invoke the various pages.

To invoke the servlets:

```
http://scott-sun:port/servlet/HelloWorldServlet
http://scott-sun:port/servlet/SessionServlet
```

To invoke the JSP pages:

```
http://scott-sun:port/migdemos/jsps/Hello.jsp
http://scott-sun:port/migdemos/jsps/snoop.jsp
```

To invoke the index HTML pages:

```
http://scott-sun:port/migdemos/index.html
http://scott-sun:port/migdemos/examples/index.html
```

Setup in OC4J

This section shows the setup in OC4J in order to migrate. Assume the following structure:

```
\private\scott-pc\migration-example\
                    migration-example.ear

                     META-INF\
                               application.xml

                      migration-example.war
                        index.html
                        WEB-INF\
                                 web.xml
                                 orion-web.xml

                                 classes/
                                         HelloWorldServlet.java
                                         SessionServlet.java

                        jsps\
                                 Hello.jsp
                                 snoop.jsp

                        examples\
                                 index.html
```

```
                                    images\
                                          blk_line_bullet_35.gif
                                          red_arrow_bullet_35.gif
```

Initially, of course, the EAR file (`migration-example.ear`) and WAR file (`migration-example.war`) would not yet exist. You would create them with the JAR utility after the rest of the directory structure has been established. They are shown within the directory structure, with contents nested beneath them, for illustrative purposes.

The WAR file would have the following structure:

```
META-INF\
META-INF\MANIFEST.MF
WEB-INF\
WEB-INF/classes\
WEB-INF\classes\HelloWorldServlet.java
WEB-INF\classes\HelloWorldServlet.java
WEB-INF\web.xml
WEB-INF\orion-web.xml
jsps\
jsps\Hello.jsp
jsps/snoop.jspjsps\snoop.jsp
examples\
examples\index.html
examples\index.html
examples\images\blk_line_bullet_35.gif
examples\images\red_arrow_bullet_35.gif
index.html
```

Then the EAR file would have the following structure:

```
META-INF\
META-INF\MANIFEST.MF
META-INF\application.xml
migration-example.war
```

Deploy the EAR file according to OC4J deployment conventions. See the *Oracle9iAS Containers for J2EE User's Guide* and the *Oracle9iAS Containers for J2EE Servlet Developer's Guide* for more information.

Be aware of the following general suggestions for OC4J migration and deployment:

- Recompiling your servlets in OC4J is not required, but might be useful to discover any deprecated servlet 2.0 methods that you are using. Even with deprecated methods, however, the servlets would still run in OC4J.
- In a servlet 2.0 environment, there are no separate servlet contexts—all servlets are grouped together. In a servlet 2.2 or 2.3 environment, you are deploying into a particular servlet context, per the WAR file. Interdependencies between servlets that work effectively together in JServ, but are in different contexts in OC4J, will cause runtime problems in OC4J.
- Place application-specific library or utility JAR files in the `WEB-INF\lib` directory.
- Place system-wide library or utility JAR files in the `j2ee\home\lib` directory, where they will be accessible to all applications.

System-wide path settings, such as for class file and JAR file locations, are specified in the global `application.xml` file:

```
j2ee\home\config\application.xml
```

This is where, for example, `j2ee\home\lib` is set as a server-wide library location.

Setup in application.xml

In this example, the application-specific `application.xml` file includes the following entries:

```
<?xml version="1.0"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application 1.
2//EN" "http://java.sun.com/j2ee/dtds/application_1_2.dtd">
<application>
  <display-name>OC4J demo application</display-name>
  <description>
     Collection of servlet samples.
  </description>
  <module>
    <web>
      <web-uri>migration-example.war</web-uri>
      <context-root>/migdemos</context-root>
    </web>
  </module>
</application>
```

Notes:

- The `<context-root>` setting is the root for the entire application.
- Each application has its own `application.xml` file containing application-specific settings. Do not confuse this with the global `application.xml` file mentioned earlier, which contains server-wide settings.

Setup in orion-web.xml

In this example, the `orion-web.xml` file includes the following entries:

```
<?xml version="1.0"?>
<!DOCTYPE orion-web-app PUBLIC "-//Evermind//DTD Orion Web Application 2.3//EN"
"http://xmlns.oracle.com/ias/dtds/orion-web.dtd">

<orion-web-app
        deployment-version="9iAS 9.0.2"
        servlet-webdir="/servlet"
>
</orion-web-app>
```

Setup in web.xml

In this example, the `web.xml` file includes the following entries:

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN
" "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
        <!-- A demo servlet, add servlets below -->
        <servlet>
                <servlet-name>HelloServlet</servlet-name>
                <servlet-class>HelloWorldServlet</servlet-class>
        </servlet>
        <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        </welcome-file-list>
</web-app>
```

URLs for Invocation in OC4J

Assuming the preceding configuration, you could use the following URLs (specifying the appropriate port) to directly invoke the various pages.

To invoke the servlets:

```
http://scott-sun:port/migdemos/servlet/HelloWorldServlet
http://scott-sun:port/migdemos/servlet/SessionServlet
```

(The `<context-root>` setting in `application.xml` and the `servlet-webdir` setting in `orion-web.xml` are relevant here.)

To invoke the JSP pages:

```
http://scott-sun:port/migdemos/jsps/Hello.jsp
http://scott-sun:port/migdemos/jsps/snoop.jsp
```

To invoke the index HTML pages:

```
http://scott-sun:port/migdemos/index.html
http://scott-sun:port/migdemos/examples/index.html
```

Migrating the principals.xml File to the Java Authentication and Authorization Service

In the Oracle9*i*AS Release 1 (1.0.2.2.x) OC4J security services, the `prinicpals.xml` file defines users and groups for mapping to roles defined in application deployment descriptors.

In Oracle9*i*AS Release 2 (9.0.2), security services are provided through the Java Authentication and Authorization Service (JAAS). For information about JAAS, see "Security Services Provided by Oracle9*i*AS JAAS" in the *Oracle9i*AS *Security Guide*.

The JAAS Admintool is provided for security administrators to manage users, realms, roles and policies. It has a command switch that migrates `principals.xml` to a JAAS realm. The syntax is:

```
java -jar jazn.jar -convert filename realm
```

where *`filename`* is the name and location of the OC4J `principals.xml` file and *`realm`* is the realm defined in the JAAS. For example:

```
java -jar jazn.jar -convert /home/config/principals.xml MyCompanyRealm
```

All permissions granted to a `principals.xml` group are granted to the JAAS role. Users that were deactivated are not migrated.

Migrating Oracle9*i*AS SOAP

This section describes how to migrate SOAP applications from Oracle9*i*AS Release 1 (1.0.2.2.x) to Oracle9*i*AS Release 2 (9.0.2).

SOAP is implemented as a servlet. A servlet delegates service invocations to user supplied implementation classes. In Oracle9*i*AS Release 1 (1.0.2.2.x), JServ was the default servlet engine. In Oracle9*i*AS Release 2 (9.0.2), OC4J is the servlet engine. To migrate SOAP applications, you must copy and re-packaging the service implementation classes and descriptors, and also consider the configuration aspects of the JServ and OC4J containers. (See "Migrating JServ to OC4J" on page 3-20 for more information on JServ and OC4J configuration.)

Oracle9*i*AS Release 2 (9.0.2) contains empty (that is, containing no services) SOAP application and web application archives ready to install. These files are named `soap.ear` and `soap.war`, and are located in *`ORACLE_HOME`*`\soap\webapps\soap.ear`. The `soap.war` file is a copy of the WAR file contained in the `soap.ear` file.

The SOAP migration process involves inserting Oracle9*i*AS Release 1 (1.0.2.2.x) files into a copy of the empty SOAP application, and then deploying the application in OC4J. Files can be “inserted” in one of two ways:

- Using `jar -x` to unpack the `soap.ear` and `soap.war` files into component directories, copying old files to the corresponding directories, then using `jar -c` to create new .ear and .war files.
- Using `jar -u` to update the contents of the .war and .ear files without unpacking them.

Below are the steps in the migration process.

1. Copy *`ORACLE_HOME_2`*`\soap\webapps\soap.ear` and *`ORACLE_HOME_2`*`\soap\webapps\soap.war` to a convenient work directory (workdir, in this example).
2. Insert all files from *`ORACLE_HOME_1`*`\soap\webapps\soap\WEB-INF\classes` into `workdir\soap.war`.
3. Insert all jar files *except* `soap.jar` and `samples.jar` from *`ORACLE_HOME_1`*`\soap\webapps\soap\WEB-INF\lib` into `workdir\soap.war`.
4. If you are sure that the old configuration file, *`ORACLE_HOME_1`*`\soap\webapps\soap\WEB-INF\config\soapConfig.xml` was never changed, go to step 6.

5. Make a copy of the old configuration file, `ORACLE_HOME_1\soap\webapps\oap\WEB-INF\config\soapConfig.xml`, renaming it to `soap.xml`.
6. Edit the file, examining the class attribute of the `providerManager` and `serviceManager` elements.

   Note: The providerManager and serviceManager interfaces have changed from Release 1, so if you supplied the class, you must change and recompile your code, then insert it in `workdir\soap.war`. The location in soap.war is directly in `WEB-INF`, not `WEB-INF\config`. The SOAP javadocs on the Oracle9*i*AS documentation CD detail the changes.

   If you did *not* supply the class, delete the class attribute from the `soap.xml` file (the line containing `class =`). Replace the `soap.xml` file in `workdir\soap.war` with the new `soap.xml`.

   All of the code to be migrated is now in `workdir\soap.jar`.
7. Insert the new `workdir\soap.jar` into `workdir\soap.ear`.
8. Deploy the .ear file in OC4J.
9. Activate the installed SOAP services as described in the *Oracle9i*AS *Web Services Developer's Guide*.

Migrating Oracle Business Components for Java

Perform the steps below to migrate Oracle Business Components for Java applications to Oracle9*i*AS Release 2 (9.0.2) using JServ. JServ is provided in Oracle9*i*AS Release 2 (9.0.2) for legacy use only, and is disabled by default. The default servlet container is OC4J.

If you want your business components applications to use OC4J, you must redeploy them from Oracle9*i* JDeveloper. However, JDeveloper cannot deploy to an Oracle9*i*AS instance. Within JDeveloper, you can deploy J2EE applications to JDeveloper's embedded OC4J server for development or testing purposes, or deploy to a standalone OC4J instance that is not embedded or within an Oracle9i Application Server (Oracle9iAS) instance. JDeveloper uses the Oracle9iAS Containers for J2EE administration console command-line tool, `admin.jar`, to deploy to the embedded OC4J or standalone OC4J.

Note: The following steps assume your business components were deployed in local mode. If your business components were deployed to Oracle8*i* JServer as an EJB or CORBA server object, you will need to redeploy them to another platform (such as the OC4J EJB module) from Oracle9*i* JDeveloper. For more information on deployment in Oracle9*i* JDeveloper, and migrating JDeveloper 3.2.3 projects to Oracle9*i* JDeveloper, see the JDeveloper documentation.

Migrating BC4J Applications

Follow this procedure to migrate BC4J applications. Instructions are included for configuring JServ in Oracle9*i*AS Release 2 (9.0.2).

1. Copy JSP files, and business components JAR files to *ORACLE_HOME_2*.
2. For each JAR (business components, web beans, and / or any other archives you created), add the line `wrapper.classpath=`*`path`*, where *path* is the path to the JAR file.
3. Uncomment the Include directive for the `jserv.conf` file in *ORACLE_HOME_2*`\Apache\Apache\conf\httpd.conf`.

   ```
   #include "/ORACLE_HOME_2/Apache/Jserv/etc/jserv.conf"
   ```

4. Edit `jserv.conf` to set directives as appropriate for how you want to use JServ. (`jserv.conf` contains Include directives for mod_jserv and mod_oprocmgr, an Oracle module that provides process management and load balancing services.)

   See Also: *Oracle HTTP Server Administration Guide* in the Oracle9*i* Application Server documentation library.

5. Edit the *ORACLE_HOME_2*`\Apache\Jserv\etc\jserv.properties` file, if needed.
6. Edit the *ORACLE_HOME_2*`\Apache\Jserv\etc\zone.properties` file, if needed.
7. (Optional) Perform the following configuration steps to enable JServ and Oracle9*i*AS Containers for J2EE (OC4J) to coexist. This is important if you have the Portal and Wireless installation type, because of the Portal dependency on OC4J.

You can specify that some applications execute on JServ and some on OC4J. Suppose you have these URLs:

`/application1/file1.jsp` to execute on JServ, and

`/application2/file2.jsp` to execute on OC4J.

You must rewrite the URL for application1.

a. Edit *`ORACLE_HOME_2`*`\Apache\Apache\conf\httpd.conf` and ensure that the following directives are active (uncommented) and present:

```
LoadModule rewrite_module libexec/mod_rewrite.so
RewriteEngine on
```

b. Edit *`ORACLE_HOME_2`*`\Apache\jsp\conf\ojsp.conf` to add these directives:

```
RewriteRule /application1/(.*)/(.*)\.jsp$ /application1/$1/$2.jsp1
ApJServAction .jsp1 /servlets/oracle.jsp.JspServlet
```

c. Remove this directive:

```
ApJServAction .jsp /servlets/oracle.jsp.JspServlet
```

d. Edit *`ORACLE_HOME_2`*`\Apache\Jserv\etc\jserv.conf` and mount `/servlets` to the JVM that will service the JSP requests. Use the ApJServMount or ApJServGroupMount directive (depending on how the JServ processes are started).

8. Restart the Oracle HTTP Server.

4

Migrating Portals Components

This chapter explains how to change the necessary configuration files and application deployment files necessary to migrate Portals components. It contains these major sections:

Migrating Oracle9iAS Portal

Migrating Oracle Ultra Search

Migrating Oracle9*i*AS Portal

This section explains how to migrate Oracle9*i*AS Portal. It is divided into the following sub-sections:

- Migrating the Mid-Tier
- Migrating Portal Development Kit (PDK) Java Web Providers
- Troubleshooting Tips for Portal Migration

Migrating the Mid-Tier

This section describes the process of migrating the Portal mid-tier from Portal 3.0.9 to 9.0.2. It contains the following topics:

Architectural Changes in Portal Release 9.0.2

Oracle Portal 9.0.2 is based on Oracle9*i*AS Release 2 (9.0.2), which contains a new J2EE-compliant servlet container (OC4J), replacing the old servlet container (JServ). Manual steps are necessary to migrate the mid-tier from the old architecture to the new architecture.

Migrating from JServ Zones to OC4J Applications

1. Configure `mod_oc4j` to forward requests of type `/servlet/page/*` to the OC4J Portal instance:

   a. Edit *`ORACLE_HOME_2`*`\Apache\Apache\conf\mod_oc4j.conf` and add the following line:

   ```
   OC4JMount /servlet/page OC4J_Portal
   ```

   Note: Place this directive next to the other OC4JMount directives in this file. The added entry must be inside the <IfModule mod_oc4j.c> block.

2. Configue OC4J to use a Portal application to run all `/servlet/page` requests:

   a. Edit *`ORACLE_HOME_2`*`\j2ee\OC4J_Portal\config\default-web-site.xml` and add the following line:

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

   Note: Place this directive next to the other <web-app application> directives in this file. The added entry must be inside the <web-site> block.

Migrating the Parallel Page Engine

The Parallel Page Engine (PPE), which ships with the Oracle Portal 9.0.2, is backward-compatible and works with Oracle Portal 3.0.9 (Oracle9*i*AS Release 1 (1.0.2.2.x). All the new settings that are required for Oracle Portal 9.0.2 are configured automatically. Manual steps are required to migrate custom settings to the new environment. Table 4–1 lists parameters in Oracle Portal 3.0.9 that can be modified in the PPE:

Table 4–1 Parallel Page Engine Parameters

Parameter	Description
cacheBuffer	Total size of the memory buffer used to retrieve a cached page. This number should be set to a value matching approximately the total size of the system's completed pages. The buffer is used to read the file from the disk and write it to the browser. If the value is too small, then multiple disk reads occur, which hampers performance. The larger the value, the fewer reads, however, this requires more resources. The default is 32768.
httpsports	A list of ports that are running SSL / HTTPS. The port numbers are delimited by a : (colon) and resembles the following: `9999:8888:7777` If Portal is running in SSL / HTTPS mode, then the port on which it is running must be specified here. The default is null.
logmode	This setting can be set to debug. If this value is set, then the PPE runs in a debug mode. If this value is not set, then the PPE runs normally with minimal writing of warnings and errors when needed.
logpath	The absolute path to which PPE logs are written. This setting may be set to any existing path. The default is the default JServ log path.
minTimeout	Number of seconds the portlet waits for an outbound HTTP request to execute. Any timeout value given by the portlet that is less than this value is increased to this value. If this parameter is not specified in initArgs, it defaults to 5 seconds.
offlinePath	If for some reason Portal needs to be off line, then this setting allows the servlet to return a pre-determined file to let the client know that Portal is off line. When this value is set, the PPE can operate in off line mode. The default is null.
poolSize	Total number of threads to use for parallel processing. The higher the number, the more requests that can be processed, however this also means that more resources will be used. The lower the number, the fewer requests which can be processed, and the fewer resources are needed. The default is 25.
prefix	The prefix used to point to modplsql. The default is `/pls`.
proxyHost	Host to use for proxy requests when needed. The default is null.

Table 4–1 Parallel Page Engine Parameters

Parameter	Description
proxyIgnore	A list of domains to ignore when using a proxy server. The domain must follow the HTTP 1.1 standard, which means that it must start with a ".", and end in a character. The default is null.
proxyPort	Port to use for proxy requests when needed. The default is null.
queueTimeout	Number of seconds a request waits in the queue before timing out and being removed from the queue. The default is 10 seconds if it is not specified in initArgs.
requesttime	Number of seconds a request is allowed to execute before it times out, if no timeout parameter is specified by the metadata. This parameter is bounded by minTimeout and maxTimeout. The default is 15 seconds if it is not specified in initArgs.
showError	Specifies whether to display PPE errors like time-out, and portlet flaws to the user. If set to true then users will see errors, if set to false then users will not see error messages. The default is true.
stall	Number of seconds a connection remains open for an outbound HTTP request. Any timeout value given by the portlet that is greater than this will be decreased to this value. The default is 65 seconds if it is not specified in initArgs.

To migrate the parameters above, copy the parameter names and values to the *ORACLE_HOME_2*`\j2ee\OC4J_Portal\applications\portal\portal\WEB-INF\web.xml` file. The examples below show the location and format of the names and values in version 3.0.9 and version 9.0.2:

Example 4–1 initArgs Parameter in 3.0.9 Format

The 3.0.9 parameter syntax is:

```
servlet.name.initArgs=NAME=VALUE
```

To configure the `httpsports` parameter with a value of 443:

```
servlet.page.initArgs=httpsports=443
```

Example 4–2 initArgs Parameter in 9.0.2 Format

The 9.0.2 parameter syntax is:

```
<servlet>
  <servlet-name>xxxx</servlet-name>
  <servlet-class>aaa.bbb</servlet-class>
  <init-param>
    <param-name>NAME</param-name>
    <param-value>VALUE</param-value>
  </init-param>
</servlet>
```

To configure the `httpsports` parameter with a value of 443:

```
<servlet>
  <servlet-name>page</servlet-name>
  <servlet-class>oracle.webdb.page.ParallelServlet</servlet-class>
  <init-param>
    <param-name>httpsports</param-name>
    <param-value>443</param-value>
  </init-param>
</servlet>
```

mod_plsql Parameter Changes in Release 2

mod_plsql DADs now use the Oracle HTTP Server Location directive for configuration. In this syntax, parameter names and values are separated by white space instead of the equal sign. Table 4–2 lists the Release 1 parameters and their Release 2 equivalents. Values are also compared for selected parameters.

> **See Also:** *Oracle HTTP Server Administration Guide*. for complete documentation of all DAD parameters.

Table 4–2 DAD Parameters

Release 1 Parameter	Release 2 Parameter	Release 1 Value	Release 2 Value
debugModules	PlsqlLogEnable	all, debug ,info notice, warn, alert crit	Off, On (All other levels besides debug / all are controlled by LogLevel in `httpd.conf`.)
username	PlsqlDatabaseUserName		
password	PlsqlDatabasePassword		

Table 4–2 DAD Parameters

Release 1 Parameter	Release 2 Parameter	Release 1 Value	Release 2 Value
connect_string	PlsqlDatabaseConnectString		
default_page	PlsqlDefaultPage		
document_table	PlsqlDocumentTablename		
document_path	PlsqlDocumentPath		
document_proc	PlsqlDocumentProcedure		
upload_as_long_raw	PlsqlUploadAsLongRaw		
always_describe	PlsqlAlwaysDescribeProcedure		
before_proc	PlsqlBeforeProcedure		
after_proc	PlsqlAfterProcedure		
reuse	PlsqlMaxRequestsPerSession	Yes, No	1000 (default), 1
pathalias	PlsqlPathAlias		
pathaliasproc	PlsqlPathAliasProcedure		
enablesso	PlsqlAuthenticationMode	Yes	SingleSignOn
custom_auth	PlsqlAuthenticationMode	Not Set	Basic
		Custom	GlobalOwa
		Global	CustomOwa
		PerPackage	PerPackageOwa
sncookiename	PlsqlSessionCookieName		
stateful	PlsqlSessionStateManagement	STATELESS_RESET (default)	StatelessWithResetPackageState
		STATELESS_FAST_RESET	StatelessWithFastResetPackageState
		STATELESS_PRESERVE	StatelessWithPreservePackageState
response_array_size	PlsqlFetchBufferSize		
exclusion_list	PlsqlExclusionList		One per line
cgi_env_list	PlsqlCGIEnvironmentList		One per line
nls_lang	PlsqlNLSLanguage		

Table 4–2 DAD Parameters

Release 1 Parameter	Release 2 Parameter	Release 1 Value	Release 2 Value
error_style	PlsqlErrorStyle	WebServer	ApacheStyle
		Gateway	ModplsqlStyle
		GatewayDebug	DebugStyle
bind_bucket_widths	PlsqlBindBucketWidths		One per line
bind_bucket_lengths	PlsqlBindBucketLengths		One per line

DAD Parameter Usage Notes

This section provides details about using the Release 2 parameters.

- Do not specify a parameter in 9.0.2 if it was not configured in the previous release. If a parameter is not specified at all, mod_plsql automatically uses the internal defaults for that parameter.

  The following parameters are deprecated because there are no longer any administration pages for mod_plsql.

  - administrators
  - adminPath
  - admindad

- The following parameters are deprecated at the global level, and must be explicitly specified in each DAD:
  - upload_as_long_raw
  - upload_as_blob
  - enablesso
  - stateful
  - custom_auth
  - error_style
  - name_prefix
  - upload_as_blob
- The `defaultDad` parameter is removed, since you can use Oracle HTTP Server rewrite rules to get the same results.

- If you use Portal to download files that had embedded space characters in their filename, you will need to set the DAD configuration parameter `PlsqlCompatibilityMode` to 1. Setting this flag prevents you from downloading documents with a plus (+) sign in them.

Migrated DAD Example

An example of the DAD in the Release 1 configuration file *`ORACLE_HOME_1`*`\Apache\modplsql\cfg\wdbsvr.app` is shown below:

```
;
[DAD_portal30]
username = portal30
password = portal30
connect_string = pk.us.oracle.com
default_page = portal30.home
document_table = portal30.wwdoc_document
document_path = docs
document_proc = portal30.wwdoc_process.process_download
upload_as_long_raw = txt, gif
upload_as_blob = *
;name_prefix =
always_describe = No
;before_proc =
;after_proc =
reuse = Yes
pathalias = url
pathaliasproc = portal30.wwpth_api_alias.process_download
enablesso = Yes
;sncookiename =
;stateful =
;custom_auth =
;response_array_size =
;exclusion_list =
;cgi_env_list =
;error_style =
;nls_lang =
;
```

The migration process converts it to the following arguments in a Location directive:

```
<Location /pls/portal30>
   SetHandler pls_handler
   Order allow, deny
```

```
   Allow from All
   AllowOverride None
   PlsqlDatabaseUsername portal30
   PlsqlDatabasePassword portal30
   PlsqlDatabaseConnectString pk.us.oracle.com
   PlsqlDefaultPage portal30.home
   PlsqlDocumentTable portal30.wwdoc_document
   PlsqlDocumentPath docs
   PlsqlDocumentProc portal30.wwdoc_process.process_download
   PlsqlUploadAsLongRaw txt
   PlsqlUploadAsLongRaw gif
   PlsqlMaxRequestsPerSession 1000
   PlsqlAuthenticationMode SingleSignOn
   PlsqlPathAlias url
   PlsqlPathAliasProcedure portal30.wwpth_alias.process_download
</Location>
```

Migrating Database Access Descriptors

The Portal and Login Server DADs have changed significantly between releases, and there is a migration tool for migrating the DADs. This section explains how to use the migration tool to migrate DADs, and describes the differences between the mod_plsql parameters.

Running the dadMigration Script

This section explains how to migrate Database Access Descriptors (DADs) from Oracle9*i*AS Release 1 (1.0.2.2.x) to Oracle9*i*AS Release 2 (9.0.2). The information in the DAD configuration file in Oracle9*i*AS Release 1 (1.0.2.2.x), `wdbsvr.app`, must be migrated to the Oracle9*i*AS Release 2 (9.0.2) configuration file, `dads.conf`. As a result of this operation, Web DAV entries are created in `oradav.conf`.

The migration is performed using the Portal Configuration Interface (PCI) via the dadMigration script. The full path and file name of the script is shown below.

`ORACLE_HOME_2``\bin\dadMigration.cmd`

The migration script reads in the old format DADs, and creates new DADs and corresponding Web DAV entries. It takes two parameters, targetOracleHome (-t) and migrationSource (-s). Follow the steps below to run the script.

1. Issue the command:

```
dadMigration.cmd -t ORACLE_HOME_2 -s ORACLE_HOME_1/Apache/modplsql/cfg/wdbsvr.app
```

where *ORACLE_HOME_2* specifies the target Oracle home (the default is the value of the ORACLE_HOME environment variable)

and *ORACLE_HOME_1* specifies the location of the v.3.0.9 DAD format file to be migrated (the default is `wdbsvr.app`).

2. Restart the Oracle HTTP Server to pick up the changes:
   a. Select the application server in ORACLE_HOME_2.
   b. Select the HTTP Server from the System Components page.
   c. Select Restart.
3. Verify that the DAD was migrated on the mod_plsql services page. To navigate to the mod_plsql services page:
   a. Select the application server in ORACLE_HOME_2.
   b. Select the HTTP Server from the System Components page.
   c. Select PL/SQL Properties from the Administration section.
   d. Locate the migrated DADs in the DADs section, using the Next and Previous links.

Note: Migration changes may be lost if another migration action or a Portal configuration action is performed before the EMD is restarted.

Warning: If the middle tier hostname and port are not the same in Release 2 as in Release 1, you must use the `ssodatan` script to re-register Portal. See the *Oracle9iAS Portal Configuration Guide* for more information.

Migrating the Mid-Tier Cache Configuration

The cache configuration migration process is manual, and requires that you change the value of parameters based on their configured values in the Oracle9*i*AS Release 1 (1.0.2.2.x) Oracle home.

Table 4–3 Mid-Tier Cache Parameters

Release 1 Parameter	Release 2 Parameter	Release 1 Value	Release 2 Value
enabled	PlsqlCacheEnable	Yes, No	On, Off
cache_dir	PlsqlCacheDirectory	Directory of cache	Directory of cache
total_size	PlsqlCacheTotalSize		
cleanup_size	PlsqlCacheCleanupSize		
cleanup_interval	PlsqlCacheCleanupInterval	User-specified value in seconds	User-specified value in minutes

Note: The Release 1 session cache settings and PL/SQL cache settings have been collapsed into one single setting for cache. See the examples below.

Example 4–3 Oracle9iAS Release 1 Cache Configuration File

The *ORACLE_HOME_1*\Apache\modplsql\cfg\cache.cfg file is shown below:

```
[PLSQL CACHE]
enabled = yes
cache_dir = \u01\app\oracleproduct\IAS1022\Apache\modplsql\cache\plsql
total_size = 25600000
cleanup_size = 10240000
cleanup_interval = 43200
max_size = 1024000
;
[Cookie Cache]
enabled = yes
cache_dir = \u01\app\oracleproduct\IAS1022\Apache\modplsql\cache\session
total_size = 25600000
cleanup_size = 10240000
cleanup_interval = 43200
max_size = 1024000
```

Example 4–4 Oracle9iAS Release 2 Cache Configuration File

The Release 1 file will get converted to *ORACLE_HOME_2*\Apache\modplsql\conf\cache.conf as follows:

```
PlsqlCacheEnable On
```

```
PlsqlCacheDirectory \u01\app\oracle\product\IAS1022\Apache\modplsql\cache
PlsqlCacheTotalSize 25600000
PlsqlCacheCleanupSize 10240000
PlsqlCacheCleanupInterval 720
PlsqlCacheMaxSize 1024000
```

Migrating Portal Development Kit (PDK) Java Web Providers

This section describes how to upgrade your PDK-Java v3.0.9 provider to PDK-Java (9.0.x). The increase in version numbers reflects the synchronization of version numbers withOracle9*i*AS Release 2 (9.0.2), of which the PDK-Java is a component. This section contains the following topics:

Installing the PDK-Java Framework and Samples

The PDK-Java Framework (included with Oracle9*i*AS in `\ORACLE_HOME\portal\pdkjava\v2\`) includes the tools and documentation necessary to build Web portlets. The PDK-Java Framework and Web Provider Samples will help you get started. Included in the PDK-Java Framework is the provider framework; portlets will use this framework and build upon it.

This section provides installation and configuration information for the PDK-Java Framework and samples. It explains how to configure the Oracle HTTP Server to run the PDK-Java Framework, the samples, and your own portlets.

The PDK-Java Framework includes these application files:

- `jpdk.ear` - Contains the sample provider applications including Java libraries, JSP files, HTML files and other resources necessary to run the sample applications.
- `template.ear` - A template EAR file to help you deploy providers.

Configuration Requirements for the PDK-Java Framework

You must have Oracle9*i*AS with Oracle9*i*AS Containers for J2EE (OC4J) installed and configured in order to run the framework and samples.

Deploying The Sample Providers

Follow these steps to deploy the sample providers:

1. Copy `jpdk.ear` into your OC4J applications subdirectory (typically, *`ORACLE_HOME_2`*`\j2ee\OC4J_Portal\applications`).
2. Add the following line to the *`ORACLE_HOME_2`*`\j2ee\OC4J_Portal\config\server.xml` file:

   ```
   <application name="jpdk" path="..\applications\jpdk.ear" />
   ```

3. Bind the application to the default site by adding the following line to the *`ORACLE_HOME_2`*`/j2ee/OC4J_Portal/config/default-web-site.xml` file:

   ```
   <web-app application="jpdk" name="jpdk" path="\jpdk\" />
   ```

4. Start OC4J, or restart the listener.

   The application will be automatically deployed based on the information specified in Step 3.

5. Access the provider using the following URL:

   `http://`*`server`*`:`*`port`*`/jpdk/providers/sample/`*`provider name`*

There are several sample providers included with the PDK-Java. They are described in Table 4–4.

Table 4–4 PDK-Java Sample Providers

Provider Name	Description
sample	Includes a variety of portlets demonstrating different rendering techniques, customization and caching
dbPersonalization	Portlets that demonstrate the use of the dbPersonalizationManager
externalApp	Demonstrates external application integration
feedback	A sample portlet that allows you to capture feedback from users of your portal

Table 4–4 PDK-Java Sample Providers

Provider Name	Description
invalidation	Sample portlets that demonstrate invalidation-based caching using Oracle9iAS Web Cache
partnerApp	Demonstrates partner application integration
subscriber	Demonstrates the use of subscription keys to identify the source of requests

Registering the Sample Providers

After setting up the Web Provider Sample with the Oracle HTTP Server, you must register the provider with Oracle9iAS Portal before adding the sample portlet(s) to a page.

1. Under the Build tab (on Oracle Portal Home Page), click on Add a Portlet Provider within the portlet called Provider.
2. Complete the provider information fields for the sample as follows:

   Name - SampleWebProvider

   Display Name - Sample Web Provider

   Timeout - 100

   Timeout Message - Application Timed Out

   Implementation Style - Web

   Register on Remote Nodes - No

   Provider Login Frequency - Once per User Session

   URL:

   *http://myserver.mydomain.com:port/jpdk/*providers/sample

   (Replace this with your provider URL.)

   Click the radio button for this selection:

   The user has the same identity in the Web providers application as in the Single Sign-On identity.

   Require Proxy - No (If no proxy is required to contact the Provider Adapter).
3. Click OK.

Once registered, all the sample portlets are displayed in the Portlet Repository.

> **Note:** When registering a new provider with Oracle Portal, only the user who registered the provider has privileges to see the provider and portlets. If necessary, go to the folder within the Portlet Repository content area with the name of the provider and update the provider privileges as required.

Adding Sample Portlets to a Page

Follow these steps to add sample portlets to a page:

1. Create a page.
2. Add the portlets from SampleWebProvider to the page.
3. View the page you just added.

   You can preview the portlets in the Portlet Repository.

Securing Your Provider

Oracle recommends that you secure access to providers when using the PDK-Java framework in a production environment. Oracle9*i*AS (via the Oracle HTTP Server configuration file `httpd.conf`) has the capability to deny access by IP address or hostname, as shown in the examples below:

Accepting Requests only from Portal

To make your provider accept requests only from Oracle Portal, you can allow access only from the Portal IP addresses in a Location directive for the provider path. Include the following in your `httpd.conf` file:

```
<Location provider path>
  order deny, allow
  deny from all
  allow from ip address 1
  allow from ip address 2
</Location>
```

where IP address 1 and IP address 2 are the IP addresses of the computer on which Portal resides.

Securing a Path

You may also want to secure the /servlet or other paths. To do this, add similar directives for those paths.

```
<Location /servlet >
  order deny, allow
  deny from all
  allow from ip address 1
  allow from ip address 2
</Location>
```

Securing a Servlet on a Path

To restrict access for a single servlet called "provider", so that it can only be accessed from my.oracle.com or portal.oracle.com, use a directive similar to the following:

```
<Location /servlet/provider >
  order deny, allow
  deny from all
  allow from my.oracle.com
  allow from portal.oracle.com
</Location>
```

See Also: *Oracle HTTP Server Administration Guide.*

Overview of PDK-Java 9.0.2

PDK-Java 9.0.x includes two versions of the PDK-Java framework:

Version 1 extends the framework that was included in PDK-Java 3.0.9, adding more complete support for mobile or wireless content providers and portlets. This version of the framework runs in the JServ servlet container.

Version 2 enables you to create web providers that run in the J2EE-compliant, OC4J servlet container. In addition to J2EE support, version 2 has been modified to make the framework more object-oriented and easier to extend without breaking existing web providers.These changes required the addition of some new classes, and some API changes, so any existing web providers will have to be modified before they can use the new framework.

Summary of Changes Between PDK-Java Versions

PDK-Java v9.0.2 includes several changes that are designed to increase the overall stability of the framework and make it easier to introduce new features with minimal impact on existing code. These changes include:

- Changing from Java interfaces to abstract classes for defining APIs
- Re-organization of the package structure
- Make framework more object-oriented

Abstract Classes Replace Interfaces

During the development of earlier releases of the PDK-Java, it became evident that the use of Java interfaces to define top level APIs made adding new features difficult. This was because new features often required API changes and those API changes had to be reflected in changes to the Java interfaces.

For developers using the default implementations, these changes were not invasive. However, for developers writing more complex providers, any interface change could cause their code to break, because their implementations no longer matched the interface definitions.

To resolve this issue, most of the Java interfaces have been replaced with abstract classes. The switch to abstract classes allows new features to be added without breaking any existing code. While this change does reduce flexibility (due to the lack of support for multiple inheritance in Java) and requires code changes to adopt the new release of PDK-Java, the long term gains in stability are worth the cost.

Package Reorganization

In addition to replacing Java interfaces with abstract classes, the package structure of the JPDK has been modified to include a new version number and to organize the classes based on functionality.

All packages now include "v2" in the package name to indicate that the package belongs to version 2 of the PDK-Java. The inclusion of a version number in the package name allows different versions of the PDK-Java to be included in the same classpath without causing collisions.

In addition to the version number change, the classes have been re-organized into a more logical structure. The new structure groups classes based on functionality. This makes it easier to find existing classes, and organize new classes as new functionality is added.

Object-Oriented Framework

To make the code more understandable for developers and create a clean separation between the provider APIs that developers implement and the underlying code that communicates with Oracle Portal, the provider and portlet interfaces have been modified.

Previously, the provider and portlet APIs were not very object-oriented, and API calls from the communication layer of the framework sometimes bypassed these objects and accessed the "controller" objects directly. In the new API, several changes have been made: the Provider interface has been split into 2 abstract classes: ProviderInstance and ProviderDefinition.

The old Provider interface represented:

- A registered instance of a provider (a registered instance representing a registered provider on a specific portal instance)
- The metadata that defines the provider (currently its portlets)

Splitting the information in this way makes the usage of the classes clearer for developers and allows a single set of metadata to be shared by many registered instances. It creates a platform for supporting provider instance-specific behavior (for example, provider configuration settings that are specific to a registered instance of that provider).

The Portlet interface has been split into 2 abstract classes: PortletInstance and PortletDefinition. The old Portlet interface really represented only the portlet metadata. All API calls that affected a portlet were actually routed directly to the appropriate controller. This architecture did not create a clean interface with the communication layer of the framework and made it difficult for developers working with more complex portlets to understand the flow of control within the framework.

To solve this problem, the PortletInstance and PortletDefinition were created. The PortletDefinition represents the sharable definition of the portlet and the PortletInstance represents a specific instance of that portlet being accessed by, or on behalf of, a specific user. The result of this split is that all APIs that affect an instance of a portlet (rendering, copying, security etc.) have been brought together in the PortletInstance class.

This change in architecture also allows developers more freedom because they are not forced to use the rendering, personalization and security frameworks provided by the PDK-Java. The default implementation of PortletInstance continues to use the familiar rendering, personalization and security controllers introduced in earlier versions of the PDK-Java. All calls from the communication layer are now made via the ProviderInstance and (if necessary) the PortletInstance.

With cleanly defined classes that represent instances of a provider and portlet, all the API calls from the communication layer of the PDK-Java can be neatly funneled through the ProviderInstance and PortletInstance interfaces, making the framework easier to understand for new developers.

Migration Options

Since there are two versions of the framework, there are three migration options:

- **Option 1:** Migrate from PDK-Java 3.0.9.x to PDK-Java 9.0.x (v1)
- **Option 2:** Migrate from PDK-Java 3.0.9.x to PDK-Java 9.0.x (v2)
- **Option 3:** Migrate from PDK-Java 3.0.9.x to PDK-Java 9.0.x (v1) and then migrate to PDK-Java 9.0.x (v2)

Option 1 is the simplest, because it does not require any changes to existing web providers. However, you must configure the JServ servlet container in Oracle9*i*AS. This approach allows you to migrate to Oracle9*i*AS Release 2 (9.0.2) and implement existing web providers quickly. However, you will not be able to use any of the features of OC4J.

Option 2 is slightly more involved, since you must modify web providers to use the new framework. For most web providers, the necessary changes are straightforward (i.e., modifying the import statements in Java classes, and changing the classes used for some method calls). In general, the methods that existed in version 1 of the framework are still there, and their function has not changed—but the class in which they are contained may have changed. Option 2 enables you to take advantage of the features of the OC4J servlet container and the new functionality in version 2 of the framework. Most new features will only be added to version 2 of the framework.

Option 3 combines Options 1 and 2. Using this migration approach, you can quickly get web providers running on Oracle9*i*AS Release 2 (9.0.2), and then migrate them to version 2 of the framework at your convenience.

Migrating from PDK-Java 3.0.9.x to PDK-Java 9.0.x (v1)

This section explains how to migrate a PDK-Java 3.0.9 provider to PDK-Java 9.0 version 1. This migration path does not require any changes to the provider code or provider definition file. The migration primarily involves configuring JServ and declaring the servlet that represents the web provider.

1. Uncomment the Include directive for the `jserv.conf` file in *`ORACLE_HOME_2`*`\Apache\Apache\conf\httpd.conf`.

   ```
   #include "/ORACLE_HOME_2/Apache/Jserv/etc/jserv.conf"
   ```

2. Edit `jserv.conf` to set directives as appropriate for how you want to use JServ. (`jserv.conf` contains Include directives for mod_jserv and mod_

oprocmgr, an Oracle module that provides process management and load balancing services.)

> **See Also:** *Oracle HTTP Server Administration Guide* in the Oracle9i Application Server documentation library.

3. Edit the *ORACLE_HOME_2*\Apache\Jserv\conf\jserv.properties file, if needed.
4. Edit the *ORACLE_HOME_2*\Apache\Jserv\conf\zone.properties file, if needed.
5. Perform the following configuration steps to enable JServ and Oracle9*i*AS Containers for J2EE (OC4J) to coexist.

   You can specify that some applications execute on JServ and some on OC4J. Suppose you have these URLs:

   `/application1/file1.jsp` to execute on JServ, and

   `/application2/file2.jsp` to execute on OC4J.

   You must rewrite the URL for application1.

   a. Edit *ORACLE_HOME_2*\Apache\Apache\conf\httpd.conf and ensure that the following directives are active (uncommented) and present:

      ```
      LoadModule rewrite_module libexec/mod_rewrite.so
      RewriteEngine on
      ```

   b. Edit *ORACLE_HOME_2*\Apache\jsp\conf\ojsp.conf to add these directives:

      ```
      RewriteRule /application1/(.*)/(.*)\.jsp$ /application1/$1/$2.jsp1
      ApJServAction .jsp1 /servlets/oracle.jsp.JspServlet
      ```

   c. Remove this directive:

      ```
      ApJServAction .jsp /servlets/oracle.jsp.JspServlet
      ```

   d. Edit *ORACLE_HOME_2*\Apache\Jserv\conf\jserv.conf and mount `/servlets` to the JVM that will service the JSP requests. Use the ApJServMount or ApJServGroupMount directive (depending on how the JServ processes are started).
6. Install the products the provider needs.
   - Oracle XML Parser v2 (required)

- Oracle JSP (required if your provider uses JSPs)
- Oracle JDBC (required if you use either of the database personalization managers or if your provider needs to access a database)

7. Install PDK-Java 9.0.x version 1. This is shipped in a zip file located in *ORACLE_HOME_2*\portal\pdkjava\v1\jpdkv1.zip. You can unzip this file in any location, but this document will assume *ORACLE_HOME_2*\portal\pdkjava/v1.
8. Add *ORACLE_HOME_2*\portal\pdkjava\v1\jpdk\v1\lib\provider.jar to the wrapper.classpath in the *ORACLE_HOME_2*\Apache\Jserv\conf\jserv.properties file.
9. Declare the servlet entries for your providers in your *ORACLE_HOME_2*\Apache\Jserv\conf\zone.properties file. (Copy and paste the servlet entries from *ORACLE_HOME_1*\Apache\Jserv\conf\zone.properties file.)

   The file now resembles that shown in Example 4–5. The following typographical conventions are used in the example:

   - Bold text indicates that a value has changed. Most of the changed values are class names.
   - Bold italics indicate that the structure of the file has changed.The highlighted section may represent removal, addition or modification.

Example 4–5 Sample Provider Definition File After Upgrade to v1

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?providerDefinition version="2.0"?>
<!DOCTYPE provider [
<!ENTITY virtualRoot "/jpdk/">
<!ENTITY physicalRoot "E:\9iAS\Apache\Apache\htdocs\jpdk\">
]>
<provider class="oracle.portal.provider.v1.http.DefaultProvider">
<session>true</session>
<containerRenderer class="oracle.portal.provider.v1.DefaultContainerRenderer" />

<portlet class="oracle.portal.provider.v1.http.DefaultPortlet">
<id>1</id>
<name>samplePortlet</name>
<title>Sample Portlet</title>
<shortTitle>Sample</shortTitle>
```

```
<description>PDK-Java version 1 portlet definition</description>
<timeout>10</timeout>
<timeoutMessage>Sample Portlet timed out</timeoutMessage>
<hasHelp>true</hasHelp>
<hasAbout>true</hasAbout>
<showDetails>true</showDetails>
<showEdit>true</showEdit>
<showEditDefault>true</showEditDefault>
<acceptContentType>text/html</acceptContentType>
<renderer class="oracle.portal.provider.v1.RenderManager">
<resourcePath>&virtualRoot;samplePortlet</resourcePath>
<appRoot>&physicalRoot;samplePortlet</appRoot>
<contentType>text/html</contentType>
<showPage class="oracle.portal.provider.v1.http.JspRenderer">
<name>showPage.jsp</name>
</showPage>
<helpPage class="oracle.portal.provider.v1.http.FileRenderer">
<name>help.html</name>
</helpPage>
<editPage class="oracle.portal.provider.v1.http.Servlet20Renderer">
<servletClass>your.package.EditServlet</servletClass>
</editPage>
<editDefaultsPage class="oracle.portal.provider.v1.http.Servlet20Renderer">
<servletClass>your.package.EditServlet</servletClass>
</editDefaultsPage>
<aboutPage class="oracle.portal.provider.v1.http.FileRenderer">
<name>about.html</name>
</aboutPage>
<showDetailsPage class="oracle.portal.provider.v1.http.JspRenderer">
<name>details.jsp</name>
</showDetailsPage>
</renderer>
<personalizationManager
class="oracle.portal.provider.v1.FilePersonalizationManager">
<dataClass>your.package.DataClass</dataClass>
<useHashing>true</useHashing>
</personalizationManager>
<securityManager
class="oracle.portal.provider.v2.security.DefaultSecurityManager">
<authLevel>STRONG</authLevel>
</securityManager>
</portlet>
</provider>
```

Migrating from PDK-Java 3.0.9.x to PDK-Java 9.0.x (v2)

This section outlines the necessary changes to an existing (PDK-Java 3.0.9) provider.

Updating Java classes, Servlets or JSPs

For this section, refer to the JavaDoc for the new framework. The JavaDoc is installed with the PDK-Java sample providers and is accessible at

`http://host:port/jpdk/apidoc`

where *host* is the name of the computer on which Oracle9*i*AS Release 2 (9.0.2) is installed and *port* is the port on which the Oracle HTTP Server is listening.

1. Change `import` statements in Java classes and JSPs to reflect the new package organization. The package hierarchy is now more structured with packages organized based on functional areas such as rendering, security, personalization etc.
2. Replace references to `oracle.portal.provider.v1.Provider` with `oracle.portal.provider.v2.ProviderInstance` or `oracle.portal.provider.v2.ProviderDefinition`, depending on the method being called. Review the JavaDoc to determine which of these classes contains the specific method being called.
3. Replace references to `oracle.portal.provider.v1.Portlet` with `oracle.portal.provider.v2.PortletInstance` or `oracle.portal.provider.v2.PortletDefinition`, depending on the method being called. Review the JavaDoc to determine which of these classes contains the specific method being called.
4. Replace references to `oracle.portal.provider.v1.DefaultSecurityManager` with `oracle.portal.provider.v2.security.AuthLevelSecurityManager`
5. Change the type of variables storing `providerId` from long to java.lang.String. (The datatype was changed to allow more flexibility in future releases of the API.)

Updating Provider Definition Files (provider.xml)

1. Replace references to `oracle.portal.provider.v1.DefaultProvider` with `oracle.portal.provider.v2.DefaultProviderDefinition` (or your own class that extends `oracle.portal.provider.v2.ProviderDefinition`).

2. Replace references to `oracle.portal.provider.v1.DefaultPortlet` with `oracle.portal.provider.v2.DefaultPortletDefinition` (or your own class that extends `oracle.portal.provider.v2.PortletDefinition`).
3. Replace references to `oracle.portal.provider.v1.http.JspRenderer` and `oracle.portal.provider.v1.http.Servlet20Renderer` with `oracle.portal.provider.v2.render.http.ResourceRenderer`.
4. Replace JSP / Servlet file references with relative URIs based on the location of the resource (JSP, servlet, or file) within your provider WAR file. See "Packaging and Deploying Your Provider" on page 4-27 for a description of the provider WAR file and how it is used to deploy your provider.
5. (Optional) Replace references to `oracle.portal.provider.v1.render.FileRenderer` with `oracle.portal.provider.v2.render.FileRenderer` or `oracle.portal.provider.v2.render.http.ResourceRenderer`. ResourceRenderer can be used for JSPs, Servlets and static files. However, FileRenderer is recommended for static files because it caches the content of the specified file in memory instead of accessing the file system each time the file needs to be rendered.
6. Replace v1 personalization managers with the equivalent v2 personalization managers.

   Version 2 of the PDK-Java framework includes two personalization managers:

   `oracle.portal.provider.v2.personalize.PrefStorePersonalizationManager`

   and

   `oracle.portal.provider.v2.personalize.DBPersonalizationManager`

   `PrefStorePersonalizationManager` replaces both `FilePersonalizationManager` and `DBPersonalizationManager2`, using the new Preference Store functionality. To use the `PrefStorePersonalizationManager`, you must declare one or more preference stores in your provider definition file. The preference stores can be either file based or database-based. File or database preference stores are compatible with the personalization managers included in PDK-Java 3.0.9.

If you previously used...	Use...
FilePersonalizationManager	PrefStorePersonalizationManager in conjunction with a file-based preference store
DBPersonalizationManager2	PrefStorePersonalizationManager in conjunction with a database preference store
DBPersonalizationManager	`oracle.portal.provider.v2.personalize.DBPersonalizationManager`

Declaring a File-based Preference Store

To declare a file-based preference, use XML similar to that shown below. The `rootDirectory` should be the same as the that used with the FilePersonalizationManager. If you did not specify a root directory when you declared your FilePersonalizationManager, then `rootDirectory` should be the same as the `provider_root` argument passed to the provider servlet. If you want to move your personalization data to a new location, zip or tar the contents of the original root directory (including all the subdirectories) and the unzip or untar into the new location. (The new value for `rootDirectory` will be the path of the directory into which the data was unzipped or untarred.)

```
<preferenceStore
class="oracle.portal.provider.v2.preference.FilePreferenceStore">
<name>prefStore1</name>
<rootDirectory>d:\root\directory</rootDirectory>
<useHashing>true</useHashing>
</preferenceStore>
```

Declaring a Database-based Preference Store

To declare a database-based preference store, use XML similar to that shown below. If you are using the DBPreferenceStore, you must declare a datasource in the `j2ee/home/config/datasources.xml` of the OC4J instance in which your provider will be deployed. The DBPreferenceStore will use JNDI to locate the datasource to use for the preference store.

```
<preferenceStore
class="oracle.portal.provider.v2.preference.DBPreferenceStore">
<name>prefStore2</name>
<connection>oc4jDataSourceName</connection>
<table>databaseTableName</table>
</preferenceStore>
```

> **Note:** Preference stores must be declared inside of the <provider> element at the same level as <portlet> elements.

7. Replace references to

   `oracle.portal.provider.v1.DefaultSecurityManager`

   with

   `oracle.portal.provider.v2.security.AuthLevelSecurityManager`

   The file now resembles the file shown in Example 4–6. The following typographical conventions are used in the example:

   - Bold text indicates that a value has changed. Most of the changed values are class names.
   - Bold italics indicate that the structure of the file has changed.The highlighted section may represent removal, addition or modification.

Example 4–6 Provider Definition File after Upgrade to v2

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?providerDefinition version="3.1"?>
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
<session>false</session>
<containerRenderer
class="oracle.portal.provider.v2.render.DefaultContainerRenderer" />
<preferenceStore
class="oracle.portal.provider.v2.preference.FilePreferenceStore">
<name>prefStore1</name>
<rootDirectory>d:\root\directory</rootDirectory>
<useHashing>true</useHashing>
</preferenceStore>
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
<id>1</id>
<name>samplePortlet</name>
<title>Sample Portlet</title>
<shortTitle>Sample</shortTitle>
<description>PDK-Java version 2 portlet definition</description>
<timeout>10</timeout>
<timeoutMessage>Sample timed out</timeoutMessage>
<hasHelp>true</hasHelp>
<hasAbout>true</hasAbout>
```

```
<showDetails>true</showDetails>
<showEdit>true</showEdit>
<showEditDefault>true</showEditDefault>
<acceptContentType>text/html</acceptContentType>
<renderer class="oracle.portal.provider.v2.render.RenderManager">
<contentType>text/html</contentType>
<showPage class="oracle.portal.provider.v2.render.ResourceRenderer">
<resourcePath>/jsps/showPage.jsp</resourcePath>
</showPage>
<helpPage class="oracle.portal.provider.v2.render.http.FileRenderer">
<appRoot>file path</appRoot>
<name>help.html</name>
</helpPage>
<editPage>/servlet/editServlet</editPage>
<editDefaultsPage>/servlet/editServlet</editDefaultsPage>
<aboutPage>/htdocs/submitServlet/about.html</aboutPage>
<showDetailsPage>/jsps/showDetails.jsp</showDetailsPage>
</renderer>
<personalizationManager
class="oracle.portal.provider.v2.personalize.PrefStorePersonalizationManager" >
<dataClass>your.package.DataClass</dataClass>
</personalizationManager>
<securityManager
class="oracle.portal.provider.v2.security.AuthLevelSecurityManager">
<authLevel>STRONG</authLevel>
</securityManager>
</portlet>
</provider>
```

Packaging and Deploying Your Provider

This section describes the steps necessary to package your provider for deployment. It contains the following topics:

Service Names or Identifiers

With the PDK-Java, you can deploy multiple providers under a single adapter servlet. The providers are identified by a service name or service identifier

(equivalent to the SOAP service identifier). When you deploy a new provider, you must assign a service name to the provider and use that service name when creating the provider WAR file.

After deployment, the correct service name must be used when registering your provider in Oracle Portal to ensure that requests are sent to the correct provider. For older releases of Oracle Portal that do not include a service name, you can choose one of your providers to be the default. If the adapter servlet receives a request that does not specify a service name, the request will be forwarded to the default provider.

WAR and EAR Files

WAR (Web Application Archive) and EAR (Enterprise Application Archive) files are standardized mechanisms used to deploy applications in a J2EE application server such as OC4J. The purpose of the WAR and EAR files is to encapsulate all the components necessary to run an application in a single file. This makes deployment simple and consistent across applications, and reduces errors when applications are moved from development to test to production environments.

WAR files include all the components of a web application, including Java libraries or classes, servlet definitions and parameter settings, JSP files, static HTML files and any other resources the application might need.

An EAR file represents an enterprise application. EAR files provide a grouping mechanism for web applications.

Prepare Working Directories

Follow these steps to prepare directories for your WAR and EAR files:

1. Create a directory named deploy.
2. Create subdirectories named `deploy\ear` and `deploy\war`.
3. Change directories to `deploy\ear`.
4. Unzip the template EAR file into `deploy\ear` with the command:

   ```
   jar -xvf ORACLE_HOME_2/portal/pdkjava/v2/lib/template.ear
   ```

5. Move`deploy\ear\template.ear` to the deploy directory.
6. Move `deploy\ear\template.war to deploy/war`.
7. Change directories to `deploy\war`.
8. Unzip the template WAR file into `deploy\war` with the command:

```
jar -xvf deploy/war/template.war
```

9. Move `deploy\war\template.war` to the deploy directory.

   Two working directories now exist with the structure and files needed to create your own WAR and EAR files.

Specifying the Contents of Your WAR File

To deploy your provider, you first create the WAR file that contains the provider and all the resources it needs to execute. The steps below explain how to do this manually; you can also use a software utility.

1. Copy any jar files that your provider needs into the `deploy\war\WEB-INF\lib` directory. This directory already contains the PDK-Java jar files.
2. If your provider needs any additional Java classes not contained in a .jar file, add them to the `deploy\war\WEB-INF\classes` directory. Ensure that the class files are saved in a directory structure that corresponds to their Java package names.
3. Add static HTML files, JSPs and images to `deploy\war`.

   Note: You can create subdirectories to organize the files. Note that the subdirectories will become part of the URI necessary to access the files. For example, you might create a subdirectory, `html`, and put all of your static HTML files there. To access a file in that directory called `help.html`, you would use the URI html/help.html to reference the file in your provider definition file. There is no restriction on the number or depth of these subdirectories.

4. Create a subdirectory under the providers directory for your provider. The name of the subdirectory will also be the service identifier or name for your provider.
5. Place your provider definition file in the subdirectory created in the previous step.
6. Copy the `_default.properties` file to *`serviceid`*`.properties` and edit it to reflect the provider's configuration.

7. If you have only one provider in your WAR file, edit `_default.properties` so that the configuration settings reflect the default provider (the provider that will be accessed if a service id is not specified in a request from a portal).

   Note: Release 3.0.9 and earlier portals cannot specify a service id, so requests will always be directed to the default provider.

8. If you use servlets to render content, edit `WEB-INF\web.xml` to add your servlets to the list of pre-defined servlets. Be careful not to remove the entries for servlets that are required by the PDK-Java.

Specifying Your Default Service

The default service is the provider that should receive any request that does not specify a service name. This feature is provided to allow you to register your provider on release 3.0.9 of Oracle Portal.

The default provider is specified in the `_default.properties` file in the deployment directory of your WAR file. The `_default.properties` file looks something like this:

```
serviceClass=oracle.portal.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
definition=providers/sample/provider.xml
autoReload=true
```

1. Edit the `definition=` entry so that it points to the provider definition file that for the default provider. The directory path should be based on the contents of the WAR file, not the physical location of the file on the filesystem.
2. If you are not using a provider definition file to define your provider, you must create an implementation of the `ProviderLoader` interface and edit the "loaderClass" entry.

Creating a WAR File

Once you have specified the contents of your WAR file, you are ready to create the WAR file itself. Follow these steps to create the WAR file:

1. Change directories to `deploy\war`.
2. Use the following command to create the WAR file:

   ```
   jar -cvf warfilename.war
   ```

   where warfilename is the name of your WAR file.

Creating an EAR File

Follow the steps below to manually configure an EAR file. You can also use a software utility to create the EAR file.

1. Change directories to `deploy\ear`.
2. Open the `META-INF\application.xml` file (extracted from the EAR file template into the working directory). The file resembles that shown below:

```
<?xml version "1.0">
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE
Application
1.2//EN"
          "http://java.sun.com/j2ee/dtds/application_1_2.dtd">
<application>
    <display-name>this is the display name of the application</display-name>
    <description>this is a description of the application</description>
    <module>
        <web>
            <web-uri>yourwarfile.war</web-uri>
            <context-root>/</context-root>
        </web>
    </module>
</application>
```

3. Change the value of the `<display-name>` element to the display name for your application.
4. Change the value of the `<description>` element so that it describes your application.
5. Change the value of the `<web-uri>` element to the name of your WAR file.
6. Save the `application.xml` file to its original name and location.
7. Copy the WAR file created earlier into the `deploy/ear` directory.
8. Change directories to `deploy/war`.
9. Use the following command to create the EAR file:

   `jar -cvf` *`earfilename`*`.ear`

   where earfilename is the name of your EAR file.

Deploying the Provider on OC4J

Follow these steps to deploy the provider to OC4J:

1. Copy the .ear file into your oc4j applications subdirectory (usually *ORACLE_HOME*\oc4j\j2ee\home\applications).

2. Add the following to *ORACLE_HOME*\oc4j\j2ee\home\config\server.xml:

   ```
   <application name="application name"
    path="../applications/ear file name" />
   ```

   where application name is the name of the application and ear file name is the name of the .ear file containing the provider.

3. Bind the web-app to the default site by adding the following to *ORACLE_HOME*\oc4j\j2ee\home\config\default-web-site.xml:

   ```
   <web-app application="application name"
            name="deployment name"
            path="/{application path}/" />
   ```

   where application name is the name of the application as specified in server.xml and deployment name is the name associated with this deployment of the application.

4. Re-register the provider on Oracle Portal 3.0.9. See "Registering Your Provider" on page 4-33.

5. Start OC4J.

   The application is automatically deployed based on the information specified in Step 3.

6. Access the provider using the following URL:

   ```
   http://host:port/application path/servlet/soaprouter
   ```

   where host is the name of the server hosting the Oracle9*i*AS listener, port is the port for the Oracle9*i*AS listener, and application path is the relative URI for the application defined in Step 3. For example:

   ```
   http://iashost:80/newProvider/servlet/soaprouter
   ```

7. Verify that the provider has been deployed and is accessible. You should see the test page for your default provider (assuming your .properties file specifies

debug=1). To view the test page for specific provider service, append the service name to the above URL, for example:

```
http://iashost:80/newProvider/servlet/soaprouter/sample
```

Registering Your Provider

Web providers have service names or identifiers as well as a URL. The URL represents the location of the adapter servlet and the service name identifies a provider deployed on that adapter.

Since service names did not exist in release 3.0.9 of Oracle Portal, the registration process is slightly different.

- Registering on Oracle Portal 9.0.2

  When registering your web provider on Oracle Portal 9.0.2, the registration wizard now includes fields for both the provider URL and service name. The URL is the URL of the adapter servlet and the service name is the name of the provider service you want to register. You should specify both the URL and the service name. If you omit the service name, you will, in effect, be registering the default provider. You should always use the service name, to eliminate ambiguity.

- Registering on Oracle Portal 3.0.9

  When registering on Oracle Portal 3.0.9, the service name is specified by appending it to the adapter URL For example:

  service name: `sample`

  servlet URL: `http://iashost:80/newProvider/servlet/soaprouter`

  registration URL:
  `http://iashost:80/newProvider/servlet/soaprouter/sample`

  If you omit `/sample` from the registration URL, then any requests will be forwarded to the provider specified in `_default.properties` (see Specifying Your Default Service). This feature is provided so you can upgrade existing providers to the new version of PDK-Java and deploy them using the original URL, without impacting any portals that were using the provider.

Web Cache Cacheability Rules after Mid-Tier Upgrade

Other than the cache setting changes described here, Oracle9*i*AS Release 2 (9.0.2) installs Web Cache by default, and Oracle Portal 9.0.2 uses the features of Web Cache.

When you upgrade a version 3.0.x Portal to Oracle9*i*AS Release 2 (9.0.2) Portal, or when the middle tier is upgraded before the Portal Repository is upgraded to Oracle9*i*AS Release 2 (9.0.2), you must set certain cacheability rules in Web Cache to prevent caching of Portal content. When it is cached, the content is no longer secure. This section explains why you must set the cacheability rules, and explains how to do it.

Note: Migration instructions for the Portal Repository are not included in this document.

Understanding Web Cache Caching Behavior

The default cacheability rules for Web Cache in Oracle9*i*AS Release 2 (9.0.2) include:

- Cache all of the following forever: *.pdf, *.html, *htm, *gif, *jpe, *jpeg, *.js
- Honor standard HTTP headers such as Expires, Last-Modified, etc.

Understanding Portal 3.0.9 Caching Behavior

Portal version 3.0.9 uses standard HTTP headers to cache "Full Page", images and documents from the database and images from the middle tier in the browser for a predetermined length of time.

Caching Behavior Using a Release 2 Mid-Tier with Web Cache and Portal 3.0.9 Repository

When a Release 2 middle tier with Web Cache is used as a front end for a Portal 3.0.9 repository, Web Cache caching rules override Portal caching rules.

- All Portal objects which were intended to be cached in the browser will be cached by Web Cache, thereby compromising security.
- Documents that were not intended to be cached in the browser will be cached in Web Cache, because of the default cacheability rules for Web Cache.
- Images that were supposed to expire after 8 hours, according to Portal caching rules, might be cached forever by Web Cache.

Setting the Cacheabillty Rules

Two cacheability rules must be set in order to prevent caching of Portal content in Web Cache. These rules prevent caching of any request to:

- The mod_plsqsl component of the middle tier. The URL pattern `/pls/`*`DAD name/`* identifies these requests. The default DAD name is `portal30`; it is used in the examples in this section.
- The Page Assembler component of the middle tier. The URL pattern `/servlet/page/` identifies these requests.

Follow these steps to add these cacheability rules:

1. Go to the Web Cache Administrator page.
   a. Log in to the Portal as the Portal Admin.
   b. Click the Builder icon in the top left corner of the page.
   c. Click the Administer tab.
   d. Click the Web Cache Administration link in the Services section.

   A user name and password dialog opens.
2. Enter `administrator` for the user name, and the password given by the Web Cache administrator. (The default password is `adminstrator`.)
3. Click OK.

   The Web Cache Administration page appears.
4. In the Navigation frame, General Configuration section, click the Cacheability Rules link.
5. Click the radio button for the first row of the Site Specific table.
6. Click the Insert Above button (located at the bottom of the table).

   The Create Cacheability Rule window appears.
7. Enter `/pls/portal30` in the URL Expression field.
8. In the HTTP Method(s) field, check the checkboxes for “GET”, “GET with query string”, and “POST”.
9. Enter `.*`(dot asterisk) in the POST Body Expression field.
10. Ensure that the Don’t Cache radio button is selected.
11. Enter "This is a cacheability rule set to ensure that the portal contents doesn’t get accidentally cached in Web Cache. This must be removed as soon as the Portal Repository is upgraded to Release 2. " in the Comment field.
12. Click Submit.

The window closes. The first row of the table contains the values you entered.

13. Click the radio button next to the rule just added.
14. Click the Insert Below button (located at the bottom of the table).

    The Create Cacheability Rule window appears.
15. Enter `/portal/page` in the URL expression field.
16. In the HTTP Method(s) field, click the checkboxes for “GET” and “GET with query string”.
17. Ensure that the Don’t Cache radio button is selected.
18. Enter "This is a cacheability rule set to ensure that the portal contents doesn’t get accidentally cached in Web Cache. This must be removed as soon as the Portal Repository is upgraded to Release 2." in the Comment field.
19. Click Submit.

    The window closes. The second row of the table contains the values you entered.
20. Click the Apply Changes button.

    The new cacheability rules are saved.
21. Click the Restart button.

    Web Cache is restarted, and will not cache the contents of a pre-Release 2 (9.0.2) Portal.

Note: Restarting Web Cache removes all of the cached objects. This might overload the middle tier.

Migrating the SSL Configuration

Migrating the SSL configuration is a manual process, consisting of configuring the SSL connections on the Portal page rendering route. Oracle9*i*AS Portal contains combinations of clients and servers, each of which is secured by an SSL connection. For example, Web Cache and the Parallel Page Engine act as both clients and servers, while the Application Server acts simply as a server.

Migrating (or re-configuring) the SSL connection in Oracle9*i*AS Release 2 (9.0.2) involves the steps listed below. The SSL certificate and wallet for the Oracle HTTP Server were prepared by the Oracle9*i*AS Migration Assistant (see "Migration of SSL

Settings" on page 2-8). You can use the certificate from the previous release, unless it is a Global Site ID (which is not supported in Release 2).

1. Configure Oracle9*i*AS Web Cache to use the wallet.

   See Also: *Oracle9i*AS *Web Cache Administration and Deployment Guide*

2. Configure the Parallel Page Engine (PPE) using the new format for initArgs.

   See Also: Example 4–2, "initArgs Parameter in 9.0.2 Format" on page 4-4.

Figure 4–1 SSL Connections in Oracle9iAS Portal

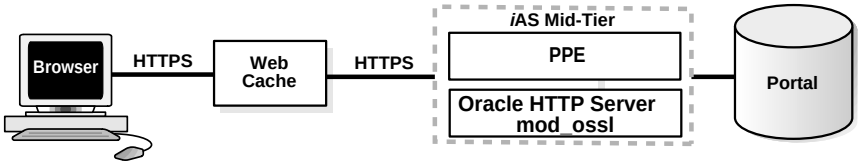


Figure 4–1 shows the communication routes involved in any Portal page rendering. Each SSL connection point is described below:

Browser to Web Cache SSL Connection

Requests for Portal pages are served over this connection. It is secured with an SSL certificate on the Web Cache listener.

Web Cache to Oracle9*i*AS Middle Tier Connection

A request can bypass the Web Cache server if port numbers are suitably configured, so you must configure Oracle9*i*AS for SSL communication.

> **See Also:** *Oracle9i Application Server Security Guide, Oracle9i Application Server Administrator's Guide*

Parallel Page Engine (PPE) to Web Cache Connection

This communication path is secure if Web Cache is secure; however, some configuration is necessary for the PPE to recognize the use of SSL.

Securing Ports to Use Certificates and HTTPS

With HTTPS, you use certificates for ports to increase security. To set this up, edit the `ORACLE_HOME_2/j2ee/OC4J/applications/portal/portal/WEB_INF/web.xmlweb.xml` file.

You must set up HTTPS such that it is used by all ports at all times. The Parallel Page Engine must be aware of which port(s) are operating under HTTPS.

Add the following to the web.xml file:

```
<init-param>
<param-name>httpsports</param-name>
<param-value>433:444</param-value>
</init-param>
```

where the port numbers 433 and 444 are replaced by your HTTPS port configuration. Your server need only have one port, but two are shown here to show the syntax used for multiple entries. Each port in this list operates using the HTTPS protocol, and must have a certificate created on the Oracle HTTP Server on that port.

Troubleshooting Tips for Portal Migration

This section describes configuration settings and files that must be present in order for the migrated Portal to work correctly. If you have problems, ensure that the conditions described in this section have been met.

Configuring Oracle9*i*AS Release 2 (9.0.2) to Serve Release 1 Image Files

In order for Oracle9*i*AS Release 2 (9.0.2) to serve all the static Oracle Portal image files used in your Oracle9*i*AS Release 1 (1.0.2.2.x) install, you will need to make the following changes to *ORACLE_HOME*\Apache\Apache\conf\httpd.conf:

```
# Configuration information added for Oracle Portal 3.0.9
Alias /help/ "/physical/location/of/your/3.0.9/help/files"
Alias /images/ "/physical/location/of/your/3.0.9/image/files "
<Directory "/physical/location/of/your/3.0.9/image/files" >
AllowOverride None
Order allow,deny
Allow from all
ExpiresActive on
ExpiresDefault A28800
<Files *>
Header set Surrogate-Control 'max-age=2592000'
</Files>
</Directory>
<Directory "/physical/location/of/your/3.0.9/help/files" >
AllowOverride None
Order allow,deny
Allow from all
ExpiresActive on
ExpiresDefault A28800
<Files *>
Header set Surrogate-Control 'max-age=2592000'
</Files>
</Directory>
```

Ensuring the Portal EAR File is Present

Ensure that the portal.ear file exists under *ORACLE_HOME_2*\j2ee\OC4J_Portal\applications. If it does not, then an incorrect install type was selected to install Oracle9*i*AS. If you have access to this file, you can just move or copy it to this location; it is not necessary to re-install Oracle9*i*AS.

Ensuring JNI Cache Library is Accessibility

Ensure that the JNI cache library `wwjni.jar` exists in *`ORACLE_HOME_2`*`\portal\jlib`, and that OC4J is configured to use it. Specifically, verify that *`ORACLE_HOME_2`*`\j2ee\home\config\application.xml` has the line

```
"<library_path="../../../lib" />
```

where the path resolves to the path on which `wwjni.jar` resides. If it does not, then an incorrect install type was selected to install Oracle9*i*AS. If you have access to this file, you can just move or copy it to this location; it is not necessary to re-install Oracle9*i*AS.

Migrating Oracle Ultra Search

To migrate from Oracle Ultra Search Release 9.0.1 to Oracle Ultra Search Release 9.0.2, you must run the migration script and perform some manual steps.

The Ultra Search migration script first verifies the version of the current system, then migrates user data. User data includes all dictionary and table data, such as metadata, data sources, mappings, crawler schedules, authentication, and query statistics.

All crawler schedules and jobs created in the 9.0.1 system are disabled before data and system migration. When migration is complete, you should re-activate the crawling schedule to re-index the document. You do not need to reconfigure the system or re-enter any data. Users can still query documents that were crawled and indexed by the previous version.

Migration Approaches for UltraSearch

There are two approaches to migrate UltraSearch user data: the in-place approach and the ETL (extract-transform-load) approach.

In-Place Migration

To migrate using the in-place approach, perform the following steps:

1. Back up the database, since there is no rollback capability in case of a hardware failure during migration.
2. Run the SQL script *`ULTRASEARCH_HOME`*`\admin\wk0upgrade.sql`. It takes the following input parameters:
   - `SYSPW` - password of the user SYS

- WKSYSPW - password of the user WKSYS
- HOST - database host machine
- PORT - database port number
- ORACLE_SID - database SID
- WK_TABLESPACE - tablespace for Ultra Search
- WK_TEMPTABLESPACE - temporary tablespace
- CONN_STRING - database connect string
- ORACLE_HOME - path of the Oracle home
- JAVA_EXE_PATH - Java executable file path
- PATH_SEPARATOR - Java classpath separator; use : (colon) for UNIX or ; (semicolon) for Windows.

The script performs the following functions:

1. Backs up user data.
2. Uninstalls 9.0.1 database objects.
3. Installs 9.0.2 database objects.
4. Recreates user instances.
5. Restores the data.

3. Rebuild the index, using the Ultra Search administration tool to re-activate all crawling schedules.

Extract-Transform-Load Migration

To migrate using the ETL approach, perform the following steps:

1. Install the 9.0.2 system in a new Oracle home, either on the same computer or a different computer.

> **Note:** If the new system is on the same computer, then you must configure the database listener port to be different from that in the 9.0.1 database, so that the new and old database can listen simultaneously.

2. Recreate user instance schemas and related database objects in the new Oracle home. For each table data source created in 9.0.1, if the base table is located in the local database, then you must copy the base table to the new 9.0.2 database. If the table data source base table is set to a remote database table, then you must recreate the database link from the new 9.0.2 database to the remote database.
3. Run the SQL script *ULTRASEARCH_HOME*`\admin\wk0migrate.sql`. It requires the following input parameters:
   - WKSYSPW - password of the user WKSYS
   - CONN_STRING - database connection string
   - SRC_WKSYSPW - password of the 9.0.1 database user WKSYS
   - SRC_CONN_STRING - source database connection string

   The script performs the following functions:

   a. Recreates user instances.

   b. Restores the data.
4. Rebuild the index, using the Ultra Search administration tool to re-activate all crawling schedules.

Migration Logs

The `wk0upgrade.sql` and `wk0migrate.sql` migration scripts for in-place and ETL migration log the actions the migration script has taken. The scripts write the following actions to the log file:

- The current execution step.
- Any error message generated by the stored procedures.
- Number of data records backed up.
- Number of data records copied or migrated.

The log file name for in-place migration is:

ULTRASEARCH_HOME`\admin\wk0upgrade.log`

The log file name for ETL migration is:

ULTRASEARCH_HOME`\admin\wk0migrate.log`

5

Migrating Wireless Components

This chapter explains how to change the necessary configuration files, application deployment files, and metadata schema in order to migrate Wireless components. It contains these sections:

Migrating Oracle9*i*AS Wireless

This section provides instructions for migrating Oracle9*i*AS Wireless. The only supported migration path is from Oracle9*i*AS Wireless 1.0.2.2.0 (the most recent production version) to Release 2 (9.0.2).

Note: In the examples in this document, when a username and password are noted as *orcladmin welcome1*, these are the username and password supplied at the time of installation.

Warning: Back up the Repository before beginning any migration activities.

Migration Scope

The following modifications are required to migrate Oracle9*i*AS Wireless 1.0.2.2.0 to Release 2 (9.0.2).

- **Migrate the database schema**: The database schema has been modified and enhanced significantly in this release.
- **Migrate the model objects**: Model objects are modified because new model objects have been introduced and schema changes have been made. Model objects in the following groups are affected:
  - group
  - role
  - user
  - service list
  - transformer
  - logical device
  - adapters

Migrating User Agent Property Files to Database

Previously, the HTTP user agent header to logical device mapping was stored as a plain Java property file. In this release, the mapping is stored in the database. This information must be migrated from the Java property file to the database.

Upgrade Transformers Due to Table Schema Changes

Because persistent representation between the transformer and logical device has been modified in Release 2 (9.0.2), a separate Java program is used to upgrade to the new table schema.

Migrate Site and Node Configuration Property to Database

The site and node configuration properties stored in the Java property files in Oracle9*i*AS Wireless 1.0.2.2.0 should be migrated to the database.

Migrate Panama User Table to OID

User information stored in the `panama` user table must be migrated to the OID.

Migration Path

This section explains how to migrate from Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) to Oracle9*i*AS Wireless Release 2 (9.0.2). assuming that Release 2 (9.0.2) will always

be installed in a different Oracle home from the prior Oracle 9*i*AS releases (a separate Oracle home is required for the new installation).

1. Back up the Repository.
2. Install Oracle9*i*AS Wireless Release 2 (9.0.2) software as a fresh installation if in-place Repository upgrade is desired. (The in-place Repository upgrade enables you to continue to use the existing database as the Wireless schema Repository. However, migrating the Repository is recommended over an in-place migration). The Repository upgrade occurs on the current Repository instead of on a new one as part of the Oracle9*i*AS Wireless Infrastructure Repository.

Before You Begin

1. Set the environment variables ORACLE_HOME JAVA13_HOME to point to the newly installed Oracle9*i*AS Wireless Release 2 (9.0.2) location, and*ORACLE_HOME_2*\JDK, respectively.
2. Create a named service entry (if one does not exist, in `tnsnames.ora`) corresponding to the Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) Repository database.

IMPORTANT: Ensure that the Oracle9*i*AS Wireless Release 1 (1.0.2.2) database has at least the 8.1.7.1.0 patchset applied. Without this patch, the SSO server will not function after migration.

IMPORTANT: Ensure that the Oracle9*i*AS Wireless Release 9.0.2 OC4J instances are shut down. To do this, issue the command `opmnctl stopall` from the *`ORACLE_HOME_2`*`\opmn\bin` directory. The Oracle Enterprise Manager daemon should, however, be running. This daemon can be started (if not already running) by invoking `emctl` start from *`ORACLE_HOME_2`*`\bin`.

In-place Migration

Before completing an in-place migration, you must delete the Wireless Provisioning Profile Entry.

The Wireless post-installer creates a provisioning profile entry pointing to a default Wireless schema in OID. For an in-place migration, this entry should be deleted by running the following script:

```
ORACLE_HOME/bin/oidprovtool operation=delete ldap_host=<ldap_host> ldap_
port=<ldap_port> ldap_user_dn='cn=orcladmin'
ldap_user_password=<password of cn=orcladmin> application_
dn='orclApplicationCommonName=Wireless1, cn=Wireless,
cn=Products,cn=OracleContext' organization_dn=<default subscriber dn>
```

For example:

```
oidprovtool operation=delete ldap_host=stpc497.us.oracle.com ldap_port=389
ldap_user_dn="cn=orcladmin"
ldap_user_password="welcome1"
application_dn="orclApplicationCommonName=Wireless1, cn=Wireless,
cn=Products, cn=OracleContext" organization_dn="dc=stpc497, dc=com"
```

Note: Type the command as a single line.

Both Installations on Same Computer

1. Using the Oracle Enterprise Manager (EM) console, change the database schema to point to the Repository database of the Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) installation. See the Oracle Enterprise Manager documentation for information on how to change the database schema.

   See Also: *Oracle Enterprise Manager Configuration Guide*, *Oracle Enterprise Manager Administrator's Guide*, and *Oracle Enterprise Manager Concepts Guide* in the Oracle9*i*AS Documentation Library.

2. Connect to the Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) Repository database as the administrative user and run the *ORACLE_HOME_2*`\wireless\sql\aq_grants.sql` script.

   The command syntax is:

   ```
   sqlplus Adminuser/Adminpassword@SID @./aq_grants.sql ias10220_wireless_
   schemaname
   ```

   for example:

   ```
   sqlplus system/manager@0817 @./aq_grants.sql ptg102_user
   ```

3. Run the script `ptgUpgrade.bat` in the *ORACLE_HOME_2*`\wireless\upgrade` directory, supplying the Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) Oracle home, Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) connect string, and Oracle9*i*AS Wireless Release 2 (9.0.2) Oracle home. For example:

   The command syntax is:

   ```
   ptgUpgrade.bat ORACLE_HOME_1 old connect string ORACLE_HOME_2 admin user
   admin password hostname port https port
   ```

   for example:

   ```
   ptgUpgrade.bat d:\wireless1022 ptg1022_user/ptg1022_
   passwd@wirelessdbservicename d:\iasv2 orcladmin welcome1 iasv2.mydomain.com
   7778 4444
   ```

   Note: While running the ptgUpgrade script, several 'Unique Constraint' violations may occur, because the new bootstrap file has user-agent entries that clash with existing user-agent entries. The violations may be safely ignored.

4. A mobile gateway URL must be registered with Oracle Portal. This step is necessary only if Oracle Portal is enabled / configured. Run the portal`Registrar.bat` script in Oracle9*i*AS Wireless Release 9.0.2 *ORACLE_HOME_2*`\wireless\sample` directory as shown below.

   ```
   portalRegistrar.bat ias_admin_user device_portal_url
   ```

   for example:

   ```
   portalRegistrar.bat orcladmin
   http://upgradedv2machine.mycompany.com:7777/ptg/rm
   ```

5. Edit the Oracle Enterprise Manager file *ORACLE_HOME_2*`\sysman\emd\targets.xml` to point to the newly upgraded database repository. Make the changes shown in bold to the target entry corresponding to `oracle_wireless`.

   ```
   <Property NAME="ConfigDBPort" VALUE="port number of upgraded database"/>
   <Property NAME="ConfigDBpassword" VALUE="schema password of upgraded
   database" ENCRYPTED="FALSE"/>
   <Property NAME="MachineName" VALUE="machine name of upgraded database"/>
   ```

```
<Property NAME="ConfigDBSID" VALUE="SID of upgraded database"/>
<Property NAME="ConfigDBMachineName" VALUE="machine name of upgraded
database"/>
<Property NAME="UserName" VALUE="schema name of upgraded database"
ENCRYPTED="FALSE"/>
<Property NAME="Port" VALUE="port number of upgraded database"/>
<Property NAME="SID" VALUE="SID of upgraded database"/>
<Property NAME="ConfigDBUserName" VALUE="schema name of upgraded database"
ENCRYPTED="FALSE"/>
<Property NAME="ORACLE_HOME" VALUE="no modification required"/>
<Property NAME="password" VALUE="schema password of upgraded database"
ENCRYPTED="FALSE"/>
<Property NAME="host" VALUE="fully qualified host name of wireless middle
tier"/>
```

Installations on Separate Computers

If the Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) installation is on computer *mc1* (and its repository database is on *db1*) and the Oracle9*i*AS Wireless Release 2 (9.0.2) installation is on computer *mc2* (and its repository database is on *db2*):

1. Using the EM console, change the database schema to point to the Repository database of the Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) installation.
2. Connect to the Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) Repository database as the administrative user and run the *ORACLE_HOME_2*`\wireless\sql\aq_grants.sql` script.

   The command syntax is:

   ```
   sqlplus Adminuser/Adminpassword@SID @./aq_grants.sql ias10220_wireless_
   schemaname
   ```

   for example:

   ```
   sqlplus system/manager@0817 @./aq_grants.sql ptg102_user
   ```

3. Copy the entire directory structure

   ORACLE_HOME`_1\panama\`

   from machine *mc1* to a temporary directory (`tmp` is used in this example), and modify the file

   ```
   tmp\panama\server\classes\oracle\panama\spatial\spatial.properties
   ```

to replace the ORACLE_HOME_1 prefix for each property with the absolute path to the temporary directory.

For example, if the temporary directory was \tmp, modify the file \tmp\panama\server\classes\oracle\panama\spatial\spatial.properties to change entries of the form:

```
file.providers.config.xml.geocoding = \ORACLE_HOME_
1\panama\server\classes\oracle\panama\spatial\geocoder\Geocoders.xml
```

to the form:

```
file.providers.config.xml.geocoding =
\tmp\panama\server\classes\oracle\panama\spatial\geocoder Geocoders.xml
```

4. On *mc2*, run the ptgUpgrade.bat script, supplying the location of the copied temporary directory, the Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) connect string, and the Oracle9*i*AS Wireless Release 2 (9.0.2) Oracle home. For example:

```
ptgUpgrade.bat d:\tmp ptg1022_user/ptg1022_passwd@wirelessdbservicename
d:\iasv2 orcladmin welcome1 iasv2.mydomain.com 7777 4443
```

> **Note:** While running the ptgUpgrade script, several 'Unique Constraint' violations may occur, because the new bootstrap file has user-agent entries that clash with existing user-agent entries. The violations may be safely ignored.

5. A mobile gateway URL must be registered with Oracle Portal. This step is necessary only if Oracle Portal is enabled / configured. Run the portalRegistrar.bat script in Oracle9*i*AS Wireless Release 9.0.2 ORACLE_HOME\wireless\sample directory as shown below.

```
portalRegistrar.bat ias_admin_user device_portal_url
```

for example:

```
portalRegistrar.bat orcladmin
http://upgradedv2machine.mycompany.com:7777/ptg/rm
```

6. The Oracle Enterprise Manager targets file must be modified to point to the newly upgraded database repository. For this purpose, edit the file *ORACLE_HOME_2*\sysman\emd\targets.xml, to make the following changes (in **bold**) to the target entry corresponding to oracle_wireless.

```
<Property NAME="ConfigDBPort" VALUE="port number of upgraded database"/>
<Property NAME="ConfigDBpassword" VALUE="schema password of upgraded
database" ENCRYPTED="FALSE"/>
<Property NAME="MachineName" VALUE="machine name of upgraded database"/>
<Property NAME="ConfigDBSID" VALUE="SID of upgraded database"/>
<Property NAME="ConfigDBMachineName" VALUE="machine name of upgraded
database"/>
<Property NAME="UserName" VALUE="schema name of upgraded database"
ENCRYPTED="FALSE"/>
<Property NAME="Port" VALUE="port number of upgraded database"/>
<Property NAME="SID" VALUE="SID of upgraded database"/>
<Property NAME="ConfigDBUserName" VALUE="schema name of upgraded database"
ENCRYPTED="FALSE"/>
<Property NAME="ORACLE_HOME" VALUE="no modification required"/>
<Property NAME="password" VALUE="schema password of upgraded database"
ENCRYPTED="FALSE"/>
<Property NAME="host" VALUE="fully qualified host name of wireless middle
tier"/>
```

Migrate the Repository

Repository migration is required if you want to migrate data in the current database to the one installed as part of the Oracle9*i*AS Wireless Release 2 (9.0.2) Infrastructure tier. The current database will be no longer used after Repository migration.

1. Export the existing database information using `exp`. For example:

   ```
   exp system/manager owner=ptg102_user file=exported.dmp log=exported.log
   ```

   An output file, `exported.dmp`, is created.

2. Connect to the Oracle9*i*AS Wireless Release 2 (9.0.2) Repository database as the administrative user and create a new schema using *`ORACLE_HOME`*`_2\wireless\sql\create_aq_user`. For example:

   ```
   SQL>@create_aq_user.sql ptg20_user welcome
   ```

3. Connect as the newly created user with the new username (`ptg20_user`) and password (`welcome`). For example:

   ```
   SQL> connect pt20_user welcome
   ```

4. Execute the create_all SQL script. For example:

   ```
   SQL> @create_all.sql
   ```

5. Use the EM console to change the Wireless connect string to point to this new schema.
6. If the Oracle9*i* database is on a different computer, use ftp to put the `exported.dmp` file on that computer.
7. Import the Oracle9*i* database using `imp`. For example:

```
imp system/manager fromuser=ptg102_user touser=ptg20_user file=exported.dmp
commit=y ingore=n log=imported.log
```

> **Notes:** For some object types the AQ uses, an object ID validation failure may occur. If this happens, you should consider using the TOID_NOVALIDATE parameter to disable validation on those types (when using Oracle Import Utility).
>
> Setting `commit=y` will improve performance since lengthy rollback processing will be avoided.

8. Perform an upgrade by following the steps provided in "Both Installations on Same Computer" or "Installations on Separate Computers", depending on whether the Oracle9*i*AS Wireless Release 1 (1.0.2.2) and Oracle9*i*AS Wireless Release 2 (9.0.2) installations are on the same or different computers.

Migrate Users

> **Note:** This procedure may be skipped if, as part of another migration, users have already been migrated from component-specific repositories to Oracle Internet Directory (OID).

From Oracle9*i*AS Wireless Release 2 (9.0.2), to support Single Sign-on, users' information stored in the Wireless Repository will be migrated to the OID (which is part of Oracle9*i*AS Wireless infrastructure).

> **Note:** In Oracle9*i*AS Release 2 (9.0.2), user information is stored in OID, where user names are *case-insensitive*. This is different from earlier versions of Oracle9*i*AS Wireless, in which user names were *case-sensitive*.

This step is mandatory regardless of whether the migration is in-place or Repository migration, and whether the installations of Oracle9*i*AS Wireless Release 1 (1.0.2.2) and Release 2 (9.0.2) are on the same or different computers.

Oracle Internet Directory does not support all possible password hashing schemes. Currently supported schemes include MD4, MD5, SHA and UNIX Crypt. For an exhaustive list of supported password hashing schemes, see "Appendix F" in *Oracle Internet Directory Administration Guide*. For dealing with non-supported password hashing schemes, see the same section of the above-mentioned guide.

Before migrating users to the Oracle Internet Directory, the default password hashing scheme of the Oracle Internet Directory must be changed to the scheme used in Oracle9*i*AS Wireless Release 1.0.2.2. To perform this step, see "Chapter 17, Directory Storage of User Authentication Credentials" in *Oracle Internet Directory Administration Guide*.

1. Change the Password Policy of Default Subscriber in OID.

   The default password policy for the subscribers in OID requires user passwords to be a minimum of 5 characters long with at least one numeric character. Before uploading existing users to OID the password policy of the default subscriber should be modified to make sure that the passwords of the existing users conform to the password policy of the subscriber. For more information on password policies see Chapter 18, "Password Policies" in *Oracle Internet Directory Administrator's Guide.*

2. Migrate the users from the database repository to OID by running `ptgUpgradeRepository.bat` on the machine on which Oracle9*i*AS Wireless Release 2 (9.0.2) is installed. For example:

   ```
   ptgUpgradeRespoitory.sh ldap_host ldap_port ldap_dn ldap_
   password subscriber_name connect_str
   ```

   where:

   - `ldap_host` is the location of the OID server
   - `ldap_port` is the port of the OID server
   - `ldap_dn` is the DN of the OID admin user
   - `ldap_password` is the password of the OID admin user
   - `subscriber_name` is the subscriber name specified at installation time. This by default is the DNS domain name of the machine on which Oracle9*i*AS is installed.

IMPORTANT: The subscriber name argument to the `ptgUpgradeRepository.bat` script must be the DNS subdomain under which the computer is registered. For example, if the fully qualified DNS name is `mymachine.bar.com`, then the subscriber name argument would be `bar`.

- `connect_str` is the connect string to the old database schema (format is: `user/password@machine_name.domain:port:sid`)

for example:

```
ptgUpgradeRepository.sh myhost.mydomain 389 'cn=orcladmin' welcome1
'mySubscriberName' ptg102_user/ptg102_password@myhost.mydomain:1521:o817
```

Additional Steps for Migrating Customizations

These steps are required only if any customizations in the form of hooks have been introduced in Oracle9*i*AS Wireless Release 1 (1.0.2.2.0):

1. Ensure that the classes for the hooks are copied to computer *mc2* and modify the *`ORACLE_HOME_2`*`\j2ee\OC4J_wireless\config\application.xml` file under the Oracle9*i*AS Wireless Release 2 (9.0.2) installation, adding a library path directive that points to the copied classes. This ensures that they are included in the classpath of the server.
2. If applications or servlets have been added as part of customization and require migration, refer to the Oracle9*i*AS documentation on migrating existing applications.

Migrating with Multiple Middle Tiers

You may choose to migrate your existing installation to an Oracle9*i*AS Release 2 (9.0.2) instance with multiple middle tiers. However, the migration scripts should be run only once from any one middle tier. The typical steps to perform when migrating an Oracle9*i*AS Wireless Release 1 (1.0.2.2.0) instance to an Oracle9*i*AS Release 2 (9.0.2) installation with multiple middle tiers are:

1. Install an Oracle9*i*AS Release 2 (9.0.2) Infrastructure tier.
2. Install as many Oracle9*i*AS Release 2 (9.0.2) middle tiers as necessary to point to this infrastructure tier.

3. Upgrade the Oracle9*i*AS Wireless Release 1 (1.0.2.2) instance from any one middle tier.
4. With the Enterprise Manager console, change the schema for each of the mid-tiers to point to the upgraded database / repository.
5. Edit the `targets.xml` file for each middle-tier instance as detailed in "Both Installations on Same Computer" or "Installations on Separate Computers".

Post-Migration Tasks

After migration, you must complete the following procedures.

Migrating URL Adapter-Based Services

After migrating services that are based on the URL adapter, ensure that the input parameters are still valid. If not, modify them from the Service Designer:

1. Click the radio-button next to the service to select it.
2. Click Edit.
3. Choose Input parameters from the side tab.
4. Enter the new URL.

   URLs that were hosted on the old instance itself, such as `http://mymachine.foo.com:7777/portal/URLtest.jsp` may not be valid any longer. In this particular instance, a solution could be to copy `URLtest.jsp` to

   ```
   [New OracleHome]/j2ee/applications/webtool/webtool-web
   ```

   and modify the service referring to it, following the above-mentioned steps.

Restart Servers

You must restart all servers on both the middle and infrastructure tiers after completing the migration process.

6

Migrating Business Intelligence Components

This chapter explains how to change the necessary configuration files, application deployment files, and metadata schema in order to migrate Business Intelligence components. It contains these major sections:

Migrating Oracle9*i*AS Forms Services

Oracle9*i*AS Release 2 (9.0.2) Enterprise Edition contains Oracle9*i*AS Forms Services Release 9*i*. Oracle9*i*AS Release 1 (1.0.2.2.x) contains Oracle9*i*AS Forms Services Release 6*i*. This section explains how to migrate Oracle9*i*AS Forms Services Release 6*i* deployments to Oracle9*i* Forms Services in Oracle9*i*AS Release 2 (9.0.2).

Oracle9*i*AS Release 1 (1.0.2.2.x) contains Oracle9*i*AS Forms Services Release 6*i*, which supports the following deployment options:

Web Interfaces

- Static HTML files
- Common Gateway Interface (CGI)
- Forms Servlet

Forms Servlet Request dispatcher types (Listeners)

- Forms Listener

- Forms Listener Servlet

In Oracle9*i*AS Release 2 (9.0.2), not all of these options are supported. In Oracle Forms Services 9*i*, the Common Gateway Interface (CGI) is no longer a Web Interface option, and the Forms Listener is no longer an option for Forms Web requests. The Oracle9i Forms Services architecture supports the following Forms Services deployment options:

Web Interfaces

- Forms Servlet

Request Dispatcher (Listener)

- Forms Listener Servlet

This section details the steps necessary to migrate Forms Services from Oracle9*i*AS Release 1 (1.0.2.2.x) to Oracle9*i*AS Release 2 (9.0.2).

This section contains the following sub-sections:

- **Migrating from Oracle9iAS Forms Services Release 6i Common Gateway Interface (CGI) to Forms 9i Servlet**

  This section explains the changes required to migrate to the Forms Servlet from the Forms CGI. Follow these steps if you are using the Forms Services Common Gateway Interface to dynamically render the Forms Applet start HTML file for your application.

- **Migrating Forms 6i Static HTML Start Files to Forms 9i Generic Application HTML Start Files**

  If you are using static HTML deployment files in Oracle Forms Services 6i, follow these steps. Static HTML deployment files are not recommended for use in Oracle9*i*DS Forms Services 9*i*, because the Forms Servlet provides functionality that makes administration and application deployment easier.

- **Migrating from Forms 6i Listener to the Forms Listener Servlet**

  If you are using the Forms Listener to start a Web Forms runtime process, follow these steps. The Forms Listener is a C program that routes Web requests for Forms applications to a runtime process started by the listener. The Forms Listener Servlet does the same, but is completely written in Java. This section explains the differences between the listener types, and how to migrate your applications from Forms Listener to the Forms Listener Servlet.

- **Migrating the Forms 6i Listener Servlet Architecture to Oracle9iAS Forms Services Release 9i**

This section explains the differences between the deployment in Oracle9*i*AS Forms Services Release 9*i* and the deployment in Oracle9*i*AS Forms Services Release 6*i*.

- **Migrating Load Balancing**

  Read this if you are using load balancing with Oracle9*i*AS Forms Services Release 6*i* (the Forms Services Metrics Server or the JServ engine load balancing method).

- **Usage Notes**

  This section contains instructions for configuration and deployment of Oracle9*i*AS Forms Services Release 9*i* when migrating from Oracle9*i*AS Forms Services Release 6*i*. The sections are:

  - "Deploying Icon Images with the Forms Servlet"
  - "Migrating Integrated Calls to Oracle9i Reports to use Reports Services"
  - "Creating Forms Listener Servlet Alias Names in OC4J"
  - "Accessing the Listener Servlet Administration Page"

- **Best Practices For Migrating to Oracle9iAS Forms Services Release 9i**

  This section provides a check list for migrating Oracle Forms Services 6i to Oracle9*i* Forms Services 9*i*. Note that at this time your Forms application source modules should already be upgraded to Forms 9*i*.

Migrating from Oracle9*i*AS Forms Services Release 6*i* Common Gateway Interface (CGI) to Forms 9*i* Servlet

The CGI was introduced in Oracle9*i*AS Forms Services Release 6*i* to allow the Forms Applet start HTML file to be rendered dynamically. The Forms CGI uses the `formsweb.cfg` configuration file and a HTML template to create the application specific start HTML file. The CGI Interface is configured by an entry in the Oracle HTTP Server `httpd.conf` file, which defines a ScriptAlias `dev60cgi` for the directory structure containing the `ifcgi60.exe` file.

The Forms Servlet renders the HTML in the same manner as the CGI, but also provides an automatic browser type detection, supporting Internet Explorer native VM. The Oracle9*i*AS Forms Services Release 9*i* servlet is configured when you install Oracle9*i*AS Release 2 (9.0.2) Enterprise Edition, and is named `f90servlet`.

To access the Forms Servlet, request the following URL:

```
http://<hostname>:port/forms90/f90servlet
```

This URL is similar to the URL used with the CGI Interface in Oracle9*i*AS Forms Services Release 6*i*. To call an application configured as `myapp` in the custom configuration section of the `formsweb.cfg` file, located in the `Forms90/Server` directory, request the following URL:

```
http://<hostname>:port/forms90/f90servlet?config=myapp
```

Configuration is automatic upon installation. The installer creates a virtual path `\forms90\` pointing to the Oracle9*i*AS Forms Services Release 9*i* configuration, `forms90app` and `forms90web`, in the Oracle9*i*AS Containers for J2EE (OC4J) home (*`ORACLE_HOME`*`\j2ee\home`).

To migrate an Oracle9*i*AS Forms Services Release 6*i* CGI environment to an Oracle9*i*AS Forms Services Release 9*i* servlet environment in Oracle9*i*AS Release 2 (9.0.2), perform the following steps:

1. Copy all of the application specific configurations from *`ORACLE_HOME_1`*`\Forms60\Server\formsweb.cfg` and append them to *`ORACLE_HOME_2`*`\Forms90\Server\formsweb.cfg` file.

   Note: It is not safe to replace the 9*i* `formsweb.cfg` file in ORACLE_HOME_2 with the 6i `formsweb.cfg` file from ORACLE_HOME_1, because the 9*i* file is different from the 6*i* file. You must copy the application configuration lines from the 6*i* file to the 9*i* file.

2. Configure `Forms90_Path` in the `default.env` file in `Forms90\Server` to point to your migrated Oracle9*i*AS Forms Services Release 9*i* application modules.

   Note: You can create your own environment file by copying `default.env`, modifying it for use with a particular application, and adding `envFile=`*`your created environment file`* to the custom application section in the `formsweb.cfg` file.

3. If you changed the Forms 6*i* HTML template files, then make the same changes to the Oracle9*i*AS Forms Services Release 9*i* HTML template files.

Note: You must make the changes in all three files: `basejini.htm`, `basejpi.htm` and `baseie.htm`, because the servlet supports JInitiator, Java Plugin and Internet Explorer native VM.

Migrating Forms 6*i* Static HTML Start Files to Forms 9*i* Generic Application HTML Start Files

Every application deployed by Oracle9*i*AS Forms Services has a custom application definition, configured in the `formsweb.cfg` configuration file. It automatically inherits the general system settings, such as the JInitiator version used or the names and locations of the base HTML template files.

The name of the custom application definition becomes a part of the Forms URL. The following custom settings define two different applications:

```
[MyHR_app]
IE=Jinit
serverURL=/forms90/l90servlet
Form = hr_main.fmx
lookAndFeel=oracle
Otherparams=myParam1=12
Userid=scott/tiger@orcl
```

The following URL invokes this application:

```
http://<hostname>:<port>/forms90/f90servlet?config=MyHR_app
```

Another custom application definition might look like this:

```
[booking_app]
IE=native
ServerURL=/forms90/l90servlet
Form = book.fmx
lookAndFeel=oracle
Otherparams= Userid=
```

The following URL invokes this application:

```
http://<hostname>:<port>/ forms90/f90servlet?config=booking_app
```

For each static HTML file, you must create a custom application definition. Part of the static HTML file is the `archive` parameter holding at least `f90all.jar` in Oracle9*i*AS Forms Services Release 9*i*. If you added a custom archive file,

then the archive parameter directive looks similar to `Archive=f90all.jar, custom.jar`. Using the Forms servlet and the `formsweb.cfg` file, the archive settings are defined under the User Parameter section. All custom application settings inherit these values, so you don't have to explicitly set this parameter, unless you add your own `custom.jar` file required by one of your applications.

If `custom.jar` was added, then you add the following lines to the custom application definition (the example assumes that you are using Jinitiator or another VM but not IE native).

```
[booking_app]
archive_jini=f90all_jinit.jar, custom.jar
archive=f90all.jar, custom.jar
ServerURL=/forms90/l90servlet
Form = book.fmx
lookAndFeel=oracle
Otherparams=
Userid=
```

To migrate, perform these steps:

1. Edit the `default.env` file in `Forms90\Server`, adding the location of the Forms9*i* application modules to the `Forms90_Path`.
2. Edit the `Forms90\Server\formsweb.cfg file` and append a custom application section for each static HTML application file that you want to replace.
3. Give each custom application section a name, containing no spaces and enclosed in square brackets [ ] (as shown in the examples above).
4. Start the application using:

   `http://<hostname>:<port>/forms90/f90servlet?config=`*`name`*

Using Static HTML Files with FormsOracle9*i*AS Forms Services Release 9*i*

If you need to, you can continue to use static HTML files in Oracle9*i*AS Release 2 (9.0.2). However, with static HTML files, some Release 2 functionality, such as Single Sign-On support, is unavailable to your Forms applications.

The Forms Listener Servlet is automatically set up to `\forms90\l90servlet` after installation. To use static HTML files in Release 2, you must modify each static start HTML file to include a value for the `serverURL` parameter. The `serverPort` and `serverHost` parameters are no longer used, and can be left undefined. Oracle9*i*AS

Forms Services Release 9*i* uses JInitiator version 1.3.x, so you must also change those settings. The required values are found in the `\forms90\server\formsweb.cfg` file.

The migration steps for this configuration are:

1. Configure `Forms90_Path` in the `\forms90\server\default.env` file to point to the migrated Oracle9*i*AS Forms Services Release 9*i* application modules.
2. Create virtual directories in the *`ORACLE_HOME_2`*`\Apache\Apache\conf\httpd.conf` file to point to the location of the static HTML start files.
3. Modify the application start HTML files as follows:
   a. Add the `serverURL` value `/forms90/l90servlet`.
   b. Change the Jinitiator version number.
4. Change the codebase parameter to `forms90/java`.
5. Edit the *`ORACLE_HOME_2`*`\j2ee\OC4J_BI_Forms\applications\forms90app\form90web\WEB-INF\web.xml` file.
6. Set the `envFile` initialization parameter for the LIstenerServlet to point to the environment file (usually *`ORACLE_HOME_2`*`\forms90\server\default.env`).

After editing, the entry in the `web.xml` file for the Forms listener servlet should resemble that shown in Example 6–1.

Example 6–1 Forms listener entry in web.xml

```
<!--Forms 9i listener servlet-->
<servlet>
  <servlet-name>l90servlet</servlet-name>
     <servlet-class>oracle.forms.servlet.ListenerServlet</servlet-class>
  <init-param>
    <param-name>envFile</param-name>
    <param-value>ORACLE_HOME_2/forms90/server/default.env</param-value>
  </init-param>
</servlet>
```

Migrating from Forms 6*i* Listener to the Forms Listener Servlet

The Forms 6*i* Listener is a C program that starts a Forms runtime process on behalf of an incoming Forms Web request. The Forms Web runtime process then is directly

accessed by the Forms client Applet, using either a direct socket or a HTTP socket connection. The Forms Listener is then no longer involved in the application Web client-server communication process, and free to handle other incoming Web requests.

The Forms Listener Servlet, a Java program, also takes incoming Web requests for a Forms application and starts the Forms server-side Web runtime process. But, unlike the Forms Listener, the Forms Listener Servlet remains between the Forms application Applet-server communication.

While the Forms 6*i* Listener listens on a specific port (by default 9000), the Forms Servlet doesn't need an extra port and is accessed by the HTTP listener port. The Forms Listener Servlet was introduced in Forms 6i patch 4 and is the only listener supported in Forms 9*i* Services.

The Forms Listener Servlet is configured during installation.

The installer creates a virtual path `/forms90/` that points to the Oracle9*i*AS Forms Services Release 9*i* Services. Oracle9*i*AS Containers for J2EE (OC4J) is the servlet environment.

Request the Forms Listener Test page with the following URL:

`http://`*`hostname`*`:`*`port`*`/forms90/f90servlet/admin`

This page indicates that the Forms Listener Servlet is configured and ready for you to use. `f90servlet` is the access name configured for the Forms Servlet during installation. The name of the Listener Servlet is `l90servlet`.

If the Forms Listener Servlet is accessed with the Forms Servlet, then only the custom application settings from the `Forms60\server\formsweb.cfg` file need to be appended to the `Forms90\server\formsweb.cfg` file. All application configurations automatically inherit the serverURL parameter value `/forms90/l90servlet` from the global system parameter settings.

Note: It is not safe to replace the 9*i* `formsweb.cfg` file in ORACLE_HOME_2 with the 6*i* `formsweb.cfg` file from ORACLE_HOME_1, because the 9*i* file is different from the 6*i* file. You must copy the application configuration lines from the 6*i* file to the 9*i* file.

To change a Forms application deployment from the Forms Listener architecture to the Listener Servlet architecture, you need only to supply a value for the serverURL

parameter in the `formsweb.cfg` file. During installation, this parameter is set to `/forms90/l90servlet`.

To migrate, perform these steps:

1. Copy your Forms application files to a new directory and migrate them to Oracle9*i*AS Forms Services Release 9*i* modules.
2. Edit the file in the `Forms90\server\default.env` directory, adding the location of the migrated Oracle9*i*AS Forms Services Release 9*i* application modules to the `Forms90_Path` variable.
3. Copy the custom application settings from the Oracle9*i*AS Forms Services Release 6*i* `formsweb.cfg` file to the Oracle9*i*AS Forms Services Release 9*i* `formsweb.cfg` file.

   Note: It is not safe to replace the 9*i* `formsweb.cfg` file in ORACLE_HOME_2 with the 6*i* `formsweb.cfg` file from ORACLE_HOME_1, because the 9*i* file is different from the 6*i* file. You must copy the application configuration lines from the 6*i* file to the 9*i* file.

4. If an applications requires its own environment file, then instead of defining a Servlet alias for the Listener Servlet, you add this information into the custom application definition section of the application. For example:

   ```
   envFile=myEnvFile.env
   ```

   where the `myEnvFile.env` is located in the `Forms90\server` directory
5. If you changed the Oracle9*i*AS Forms Services Release 6*i* HTML template files, then make the same changes to the Oracle9*i*AS Forms Services Release 9*i* HTML template files.

Note: If you need to change the underlying base HTML files, you should make a copy of the provided template files before editing them. Save the edited HTML files under a different name and leave the default templates that were provided with the installation unchanged. This prevents overwriting of your custom HTML template files when patch sets are applied to your application.

To use your own template files with applications, use these parameters in the system section, or one of your custom application definitions:

```
baseHTML=your base template.htm
baseHTMLJinitiator=your base jinit.htm
baseHTMLie=your base ie.htm
```

6. Start the application using:

```
http://hostname:port/forms90/ 90servlet?config=application
```

Migrating the Forms 6*i* Listener Servlet Architecture to Oracle9*i*AS Forms Services Release 9*i*

In Oracle9*i*AS Forms Services Release 6*i* the Listener Servlet, if not aliased, is accessed by `oracle.forms.servlet.ListenerServlet`. The Listener Servlet configuration exists in the `jserv.properties` and the `zone.properties` files.

In Oracle9*i*AS Forms Services Release 9*i*, the Forms Listener Servlet is the same except for the servlet names: `f90servlet` and `l90servlet`, and the servlet container, which is now Oracle9*i*AS Containers for J2EE (OC4J). As in Release 1, the configuration is performed during installation. The Listener Servlet configuration in OC4J is stored in `Forms_OC4J_Home\applications\forms90app\forms90web\web-inf\web.xml`. Some initialization parameters, like the envFile parameter, need no longer be configured with the servlet engine because they are moved into the formsweb.cfg file.

Installing Oracle9*i*AS Release 2 (9.0.2) configures the Forms Listener servlet.

The installer creates a virtual path `\forms90\` that points to the Oracle9*i*AS Forms Services. Oracle9*i*AS Containers for J2EE (OC4J) is the servlet environment.

After installation, the following URL invokes the Forms Listener test page:

```
http://<hostname>:<port>/forms90/f90servlet/admin
```

This page indicates that the Forms Listener Servlet is configured. `f90servlet` is the access name configured for the Forms Servlet during installation; the name of the Listener Servlet is `l90servlet`.

To migrate, perform these steps:

1. Copy your Forms application files to a new directory and migrate them to Forms 9*i* modules.
2. Edit the `default.env` file in the `Forms90\server` directory, adding the location of the migrated Forms9*i* application modules to the Forms90_Path variable.
3. Copy the custom application settings in the Oracle9*i*AS Forms Services 6i `formsweb.cfg` file and add them to the Forms9*i* `formsweb.cfg` file.

   Note: It is not safe to replace the 9*i* `formsweb.cfg` file in ORACLE_HOME_2 with the 6i `formsweb.cfg` file from ORACLE_HOME_1, because the 9*i* file is different from the 6*i* file. You must copy the application configuration lines from the 6*i* file to the 9*i* file.

4. If an application requires its own environment file, then instead of defining a servlet alias for the Listener Servlet, you add this information into the custom application definition section of the application. For example:

   ```
   envFile=myEnvFile.env
   ```

   where the `myEnvFile.env` is located in the `Forms90\server` directory
5. If you changed the Oracle9*i*AS Forms Services Release 6*i* HTML template files, then make the same changes to the Oracle9*i*AS Forms Services Release 9*i* HTML template files.
6. Start the application using:

   `http://`*`hostname`*`:`*`port`*`/forms90/f90servlet?config=`*`application`*

Migrating Load Balancing

The method of load balancing in Oracle9*i*AS Forms Services Release 6*i* depends on the deployment type used.

- With the Forms 6*i* listener, the Metrics Server (a separate process) performs load balancing.
- With the Forms 6*i* servlet, load balancing is configured with the JServ servlet engine using a round robin load balancing among JServ engines.

In Oracle9*i*AS Forms Services Release 9*i* in Oracle9*i*AS Release 2 (9.0.2), the load balancing is performed by mod_oc4j. mod_oc4j binds Web requests to the servlet container processing the Forms Servlet and the Forms Listener Servlet.

For information on setting up clusters for load balancing, see the *Oracle9i Application Server Administrator's Guide*.

Usage Notes

This section contains instructions for configuration and deployment of Oracle9*i*AS Forms Services Release 9*i* migrated from Oracle9*i*AS Forms Services Release 6*i*.

Deploying Icon Images with the Forms Servlet

Using static HTML start files in Oracle9*i*AS Forms Services Release 6*i* allowed you to store images used by an application in a location relative to the start HTML file. The Forms Servlet in Oracle9*i*AS Forms Services Release 9*i* does not support this.

The alternative is to use the `imagebase` parameter with a value of `codebase` as the location for the icon images for applications. The `codebase` value refers to the `Forms90\java` directory, which contains all the Forms client java archive files. For performance reasons, it is not a good idea to store images here.

Instead, Oracle recommends bundling icons into a separate archive file, which improves performance because archives are cached permanently on the client. The instructions below explain how to create the archive file and place the images there.

1. Verify that the `jar` command succeeds. If it doesn't, then you need to make sure that there is a JDK installed in your system with the appropriate path environment variable entry (pointing to the \bin directory).
2. Navigate to the directory containing the application images and issue the command:

   ```
   jar -cfv application_images.jar *.extension
   ```

   where *`application`* is the name of the application and *`extension`* is the extension of the image file (e.g., `.gif`).

   A jar file, *`application`*`_images.jar` is created in the current directory.

3. Copy the *application*_images.jar file to the Forms90\Java directory.
4. Edit the formsweb.cfg file, adding the imageBase=codebase to the custom application section defined for the application.
5. Add the *application*_images.jar file to the archive path used by the application by adding the following lines to the custom application section:

```
archive_jini=f90all_jinit.jar,application_images.jar
archive=f90all.jar,application_images.jar
archive_ie=f90all.jar,application_images.jar
```

> **Note:** archive_ie should contain f90all.cab, which is the better archive format when using IE5 native VM. Because you can't mix archives and cab files, you must either create cab file for the images or use f90all.jar for Forms applications (the former is recommended).

See *Deploying Oracle Forms Applications* (Deploying Forms to the Web chapter) in the Oracle9*i*AS documentation library for more information on deploying custom icon files with Forms Services.

Migrating Integrated Calls to Oracle9*i* Reports to use Reports Services

In Oracle9*i*AS Release 2 (9.0.2), integrated calls to Oracle Reports in Forms are no longer handled by a client-side background engine. Oracle9*i*AS Forms Services Release 9*i* requires that applications use the Run_Report_Object built-in, calling Reports 9*i* Services to process integrated Reports. The Reports Services are set up on your system. To migrate the call, follow the steps below.

1. Change all occurrences of Run_Product(Reports,...) to the equivalent call using Run_Report_Object().
2. Add the location of the application's Reports modules to the Reports90_Path of the Reports Services.
3. Change Run_Report_Object to reference Reports Services.

> **See Also:** Run_Report_Object() built-in and Reports Services white paper at http://otn.oracle.com

Creating Forms Listener Servlet Alias Names in OC4J

In Oracle9*i*AS Forms Services Release 6*i*, before patch 8, it was necessary to create alias names for the Forms servlet in the *ORACLE_HOME*\Apache\Jserv\conf\zone.properties file in order to use individual environment files for different applications. The Forms servlet in Oracle9*i*AS Forms Services Release 9*i* does not require this. You can set the environment file name in the formsweb.cfg file using the envFile parameter, shown below.

```
EnvFile=myApp.env
```

Alias names for the Forms servlet in Forms 9*i* are no longer created in:

```
ORACLE_HOME\Apache\JServ\conf\jserv.properties
```

Instead, they are created in

```
ORACLE_HOME\Forms_OC4J_Home\applications\forms90app\forms90web\web-inf\web.xml
```

Simply copy the content between the <servlet> and </servlet> tags and change the servlet's name. To create a URL mapping for the new servlet alias name add the following to the file.

```
<servlet-mapping>
<servlet-name>New_Servlet_Name</servlet-name>
<url-pattern>/New_URL_Name*</url-pattern>
</servlet-mapping>
```

Accessing the Listener Servlet Administration Page

You can display a test page for the Listener Servlet in Oracle9*i*AS Forms Services Release 6*i* with the following URL:

```
http://hostname:port/servlet/oracle.forms.servlet.ListenerServlet
```

The information displayed depends on the value of the initialization parameter TestMode. This is set in the *ORACLE_HOME*\Apache\JServ\etc\zone.properties configuration file.

You can display the test page for Forms 9*i* Services with the following URL:

```
http://hostname:port/forms90/f90servlet/admin
```

The information displayed depends on the value of the initialization parameter TestMode. This is set in the ORACLE_HOME\Forms_OC4J_

`Home\applications\forms90app\forms90web\web-inf\web.xml` configuration file. An example is shown below.

```
<init-param>
<!-- Display sensitive options on the /admin page ? -->
   <param-name>testMode</param-name>
   <param-value>true</param-value>
</init-param>
```

Best Practices For Migrating to Oracle9*i*AS Forms Services Release 9*i*

This section provides general recommendations and considerations for migrating Forms applications to Oracle9*i*AS Release 2 (9.0.2).

- Keep your Oracle9*i*AS Release 1 (1.0.2.2.x) installation until you successfully deploy your applications on Oracle9*i*AS Release 2 (9.0.2).
- Deploy Oracle9*i*AS Forms Services Release 6*i* using the new Listener Servlet architecture.
- Use the Forms servlet rather than the Forms CGI.
- Replace `Run_Product` calls to integrated Reports with `Run_Report_Object` calls to Reports Services (or use the PL/SQL conversion utility later in Forms9*i*).
- Install Oracle9*i*AS Release 2 (9.0.2) and configure the `formsweb.cfg` file in the `Forms90\server` directory with the information used by your applications. Copy the environment files used by your Forms applications to the same directory. Copy the migrated Forms application module files to the machine running Oracle9*i*AS Release 2 (9.0.2) if it is not the same machine.
- After starting Oracle9*i*AS Release 2 (9.0.2), access the Forms 9*i* Services Listener Servlet test page with the following URL:

  `http://`*`hostname`*`:`*`port`*`/forms90/f90servlet/admin`

- Verify that the application setting is added to the `formsweb.cfg` file and that the environment variable `Forms90_Path` contains the directory of the application modules.
- Verify that the database is accessible from using a SQL*Plus connect.
- Type `http://`*`hostname`*`:`*`port`*`/forms90/f90servlet?config=your app` to invoke your application.

- Test applications deployed on Forms9*i* Services before decommissioning Oracle9*i*AS Release 1 (1.0.2.2.x).
- Migrate source files first, and completely back up and secure application files.

Migrating Oracle9*i*AS Reports Services

This section explains how to migrate a 6i Reports Server configuration from Oracle9*i*AS Release 1 (1.0.2.2.x) to Oracle9*i*AS Release 2 (9.0.2). It contains the following sections:

Migrating Reports Configuration Files

If you want to reuse the 6i Reports server persistent file and server configuration file, copy the file

ORACLE_HOME_1\reports60\server*report server name*.ora

to:

ORACLE_HOME_2\reports\conf*report server name*.ora

and, if present, copy this file

ORACLE_HOME_1\reports60\server*report server name*.dat

to:

ORACLE_HOME_2\reports\server*report server name*.dat

If you want to reuse the 6i Reports server `cgicmd.dat` file, copy it from:

ORACLE_HOME_1\reports60\server\cgicmd.dat

to

ORACLE_HOME_2\reports\conf\cgicmd.dat

Security features have been improved for Oracle9*i*AS Reports Services. With Oracle 6*i*Reports, you may have placed connect string information in the `cgicmd.dat` file. For improved security in Oracle9*i*AS Reports Services, you should consider moving this connect string information into the Oracle Internet Directory (OID).

See Also: *Oracle9iAS Reports Services Publishing Reports to the Web*

Using Oracle9*i*AS Portal with Oracle9*i*AS Reports Services

To facilitate a smooth transition to Oracle9*i*, your Oracle9*iAS* Reports Server can use Oracle9*i*AS Portal 9.0 or 3.0.9 as its security repository and the destination for report content you are pushing into a page group or content area. This compatibility is extremely useful if you are currently using Oracle9*i*AS Portal 3.0.9 and Oracle Reports 6*i* integration in Oracle9*i*AS Release 1 (1.0.2.2.x) and want to continue to use Oracle9*i*AS Portal 3.0.9 while upgrading to Oracle9*i* Reports.

To configure Oracle9*i* Reports in this way, the Reports Server needs to be able to use an Oracle9*i*AS Portal 3.0.9 connection as its security and destination for reports. When the Reports Server detects that the security system is an Oracle9*i*AS Portal 3.0.9 instance instead of an Oracle9*i*AS Portal 9.0 instance, it will run in 6*i* security compatible mode. The Reports Servlet also uses this mode and behaves appropriately for any report request that utilizes the Oracle9*i*AS Portal 3.0.9 integration.

Simultaneously, you can have another Oracle9*i*AS Reports Server that uses Oracle9*i*AS Portal 9.0 for its security and destination element. This Oracle9*i*AS Reports Server runs with all of the latest functionality, including support for Single Sign-on and OID for authentication. The Reports Servlet also uses this mode and behaves appropriately for any report request that utilizes the Oracle9*i*AS Portal 9.0 integration. The servlet will also use Single Sign-on if configured to do so.

Note: You only need one instance of the Oracle9*i*AS Reports Servlet to service your Oracle9*i*AS Reports Servers, regardless of whether they are operating with an Oracle9*i*AS Portal 3.0.9 instance or an Oracle9*i*AS Portal 9.0 instance. The servlet will switch between 6*i* and 9*i* modes as necessary depending upon the Reports Server to which it is sending the request.

To ensure that your Oracle9*i*AS Reports Server operates correctly, do the following:

- To set up the Oracle9*i*AS Reports Server to work with Oracle9*i*AS Portal 3.0.9, enter the 3.0.9 Portal's database username/password@tnsname for the securityUserid property under the security element and portalUserid property under the destination element in the Reports Server configuration file *servername*.conf. For example:

```
<security id="rwSec" class="oracle.reports.server.RWSecurity">
    <property name="securityUserid" value="portal309_id/portal309_
    password@portal_schema" confidential="yes" encrypted="no"/>
</security>
```

- To set up the Oracle9*i*AS Reports Server to work with Oracle9*i*AS Portal 9.0, enter the 9.0 Portal's database username/password@tnsname for the securityUserid property under the security element and portalUserid property under the destination element in the Reports Server configuration file *servername*.conf. For example:

```
<security id="rwSec" class="oracle.reports.server.RWSecurity">
   <property name="securityUserid" value="portal90_id/portal90_
   password@portal_schema" confidential="yes" encrypted="no"/>
</security>
```

See Also: *Oracle9iAS Reports Services Publishing Reports to the Web*

Oracle Graphics Migration

Oracle9*i*AS Reports Services has a new graphing engine that replaces Oracle Graphics from earlier releases. You cannot migrate Oracle Graphics graphs to the new graphing engine, but Oracle9*i*AS Reports Services can use the Oracle Graphics engine for backward compatibility.

To make the Oracle Graphics engine backward-compatible on Windows, ensure that the Oracle Graphics 6*i* home is in the system path.

The following restrictions apply to the use of Oracle Graphics in Oracle9*i*AS Reports Services:

- You can only use Oracle Graphics graphs in paper layouts. They will not work in the new JSP Web layouts. If you want to put an Oracle Graphics graph into a Web format, you must run the paper layout to a Web destination format (e.g., HTML or HTMLCSS).
- Reports that use Oracle Graphics must be saved in the Reports Definition File (RDF) format. They cannot be saved in any of the other formats, such as XML.

- You cannot add new Oracle Graphics charts in Oracle9*i*AS Reports Services. You can only run existing Oracle Graphics charts. All new charts will use the new charting engine that comes with Oracle9*i*AS Reports Services.

Deprecated Features in Oracle9*i*AS Reports Services

Deprecated features in Oracle9*i*AS Reports Services are listed below. Existing reports using these features will continue to function without modification, but these features are no longer documented and further use is discouraged.

- User exits
- rwcgi web executable
- Command line options: CURRENCY, THOUSANDS, DECIMAL, PROFILE, ERRFILE, LOGFILE, BACKGROUND, KEYIN, KEYOUT
- SRW.SET_ATTR built-in
- OLE2 object support

The following features are removed entirely:

- rwrunc character mode runtime
- client/server GUI report previewer in rwrun
- rwrbe60 background engine
- rwows60 OAS cartridge
- obe60 query builder
- obs60 schema builder

> **See Also:** http://otn.oracle.com for more information about deprecated and obsolete features and alternatives to their use.

Migrating Oracle9*i*AS Discoverer

This section explains how to migrate Oracle9*i*AS Discoverer from Oracle9*i*AS Release 1 (1.0.2.2.x) to Oracle9*i*AS Release 2 (9.0.2). This process involves the following tasks:

- Migrating Preferences
- Updating the End User Layer
- Updating URL References

- Configuring Session Timeout
- Migrating Viewer Customizations

Before you begin the migration process:

1. Confirm that the Oracle9*i*AS Release 2 (9.0.2) Discoverer installation was successful. Run the Discoverer demonstrations with the links available from:

   `http://hostname:port/discoverer/intro/html/disc_demo_intro.htm`

Migrating Preferences

This section explains how to migrate default user and user-level preferences.

Migrating Default User Preferences

To migrate default user preferences, follow these steps:

1. Compare the original version 4.1 file

   ORACLE_HOME_1`\discwb4\util\pref.txt`

   to

   ORACLE_HOME_2`\discoverer902\util\pref.txt`

2. If changes were made to the *ORACLE_HOME_1*`\..\pref.txt` file, edit the `ORACLE_HOME_2..\pref.txt` file, so that it contains the same changes as the original version 4.1 file.

3. To apply the default preferences, issue the following command:

   ORACLE_HOME_2`/discoverer902/util/applypreferences.bat`

Migrating User Level Preferences From Discoverer 4.1 to Discoverer 9.0.2

If you are migrating from one Oracle home to another on the same computer, then to migrate user level preferences from Discoverer 4.1:

1. Issue the following command:

   ORACLE_HOME_2/Discoverer902/Util/MIGRATEPREFS.BAT

   This command copies user level preferences from the Windows registry to a file called reg_key.dc.

If you are migrating from one computer to another computer, use the instructions provided on the Oracle Technology Network. To access these:

1. Go to the Oracle Technology Network at http://otn.oracle.com.
2. Click on Products, then Oracle9*i* Application Server, then Documentation. Refer to the Release Notes Addendum for Windows NT.

Updating the End User Layer

The version of Discoverer that ships with Oracle9*i*AS Release 2 (9.0.2) requires an End User Layer created by Discoverer Administrator 9.0.2 (which is shipped with Oracle9*i* Developer Suite Release 2 (v9.0.2). If you have an existing Discoverer v4.1 End User Layer, you must upgrade the End User Layer from v4.1 to v5.1 using Discoverer Administrator 9.0.2.

> **See Also:** *Oracle9i Discoverer Administrator Administration Guide* in the Oracle9*i* Developer Suite Documentation Library.

Updating URL References

All Discoverer Viewer and Discoverer Plus URL references have changed. These include, but are not limited to, links within the Web site and client bookmarks. You must manually replace all occurrences of old URLs with the new URLs, according to the tables below:

Discoverer Plus Release 1 URL	Release 2 URL	Example
http://hostname/Discwb4/*html*/english/ms_ie/start_ie.htm or http://hostname/Discwb4/*html*/english/netscape/start_nn.htm	http://hostname/discoverer/plus	Change http://*host*:*port*/Discwb4/english... to http://*host*:*port*/discoverer/plus

Discoverer Viewer Release 1 URL	Release 2 URL	Example
http://hostname/Discoverer4i/Viewer	http://hostname/discoverer/viewer	Change http://*host*:*port*/Discoverer4i/Viewer to http://*host*:*port*/discoverer/viewer

Configuring Session Timeout

The `session-timeout` parameter has moved from

ORACLE_HOME_1\Apache\JServ\servlets\discoverer4i.properties

to

ORACLE_HOME_2\j2ee\OC4J_BI_Forms\discoverer\web\WEB-INF\web.xml

The `session-timeout` parameter controls the http session timeout, the number of minutes the Discoverer servlet waits for a browser to make another http or https request before terminating the user's http session.

The `session-timeout` parameter must be added to the `web.xml` file, directly following the line that reads `</servlet>`, before the bottom of the `<web-app>` section of the file, as shown below:

```
<web-app>
   <servlet>
   ...
   </servlet>
   <session-config>
      <session-timeout>15</session-timeout>
   </session-config>
</web-app>
```

Migrating Viewer Customizations

A number of files control the appearance and behavior of Oracle9*i*AS Discoverer Viewer. Those files might have been modified to customize Discoverer Viewer to meet certain requirements. Installing Oracle9*i*AS Release 2 (9.0.2) installs new files with different names. Specifically, `disco4iv.xml` has been renamed `configuration.xml` and `viewer_config.xml` has been renamed `ui_config.xml`. If you have modified the original configuration files and .xsl files, then you must edit the new files and reapply the changes from your original version 4.1 files.

> **Note:** Do not simply replace the new files with the original version 4.1 files (do not simply rename the files). Follow the steps in this section to migrate only the customizations, not all of the differences, from the original version 4.1 files to the new files.

1. For each original version 4.1 file in *ORACLE_HOME_1*\Apache\Apache\htdocs\discwb4\disco4iv\htm, perform the following tasks:

   a. Use the table below to identify the file that contains the equivalent information.

Release 1 File	Release 2 File
disco4iv.xml	web\web-inf\configuration.xml
viewer_config.xml	web\common\xsl\ui_config.xml

   b. Compare the original version 4.1 file with the equivalent Release 2 file in the *ORACLE_HOME_2*\J2EE\OC4J_BI_Forms\discoverer\web\directory.

   c. Edit the equivalent Release 2 file to incorporate any customizations found in the original version 4.1 file.

> **Note:** If you modify `configuration.xml`, consider using Oracle Enterprise Manager to make the changes. Oracle Enterprise Manager provides a user interface for editing `configuration.xml` and includes useful information about each setting.

Upgrading JInitiator

You must use Oracle9*i*AS Release 2 (9.0.2) Discoverer with the supplied version of JInitiator (1.3.x) or a later version. The version of JInitiator is specified by the `jvm` element in the

ORACLE_HOME_2\j2ee\OC4J_BI_Forms\discoverer\webplus\xsl\plus_config.xml

file. Do not attempt to run Oracle9*i*AS Release 2 (9.0.2) Discoverer with the version of JInitiator supplied with Oracle9*i*AS Release 1 (1.0.2.2.x).

7

Migrating Management Components

This chapter explains how to change the necessary configuration files, application deployment files, and metadata schema in order to migrate Management components. It contains these major sections:

Migrating Oracle Enterprise Manager

> **Note:** This migration procedure applies only if you have installed the Oracle Management Server as part of an Oracle9*i*AS Release 2 (9.0.2) Infrastructure installation, and you previously used Oracle Enterprise Manager to manage Oracle9*i*AS Release 1 (1.0.2.2.x). The procedure in this guide does not apply to the Oracle Enterprise Manager Web Site, which provides Web-based management tools for Oracle9*i*AS Release 2 (9.0.2), and requires no migration. For information about these Enterprise Manager components, see the *Oracle9i Application Server Administrator's Guide.*

Migrating Oracle9iAS Single Sign-On

Migrating Oracle Internet Directory

Migrating Oracle Enterprise Manager

This section explains how to upgrade an Oracle Enterprise Manager Release 2.x repository to a Release 9*i* repository. Existing pre-9*i* repositories are not upgraded automatically during installation. To upgrade, you must run the Oracle Enterprise Manager Configuration Assistant manually after the installation.

The Oracle Enterprise Manager Configuration Assistant takes an existing Release 2.x repository and upgrades it directly to a Release 9*i* repository.

Preparing for Migration

This section outlines the steps you must perform before you migrate Oracle Enterprise Manager.

Stop Management Servers and Enterprise Manager Applications

Before you attempt to perform an upgrade, you must first stop all Management Servers and Oracle Enterprise Manager applications that are using this repository. If any Management Server is currently using this repository, upgrading the repository causes a server error.

If you are using an Oracle Enterprise Manager version prior to Release 2.2, or if you are using any separately ordered management pack, see the *Oracle Enterprise Manager Configuration Guide* for important information about migrating your Enterprise Manager Repository.

Back Up the Repository

Before you attempt to upgrade the repository, you must first back up the database or repository schema using the standard export mechanism. The EXPORT utility is a base utility shipped with the Oracle database server.

Note: A repository created under the SYS user cannot be exported.

If there is a failure during a repository upgrade, the repository will no longer be usable. The failed repository would no longer appear in the list of repositories that could be upgraded.

Upgrading the Repository

To upgrade the repository, perform the steps in the following sections.

> **Note:** All job and event details in the repository are stored in binary fields to keep the information secure. Because of the way the data is encrypted, you cannot export data from one schema to another schema with a different name.

1. Start the Enterprise Manager Configuration Assistant by performing the following steps:

   Start the Enterprise Manager Configuration Assistant from the Windows Start Menu, or

   Start the Enterprise Manager Configuration Assistant from the command line using the command:

   ```
   emca
   ```

   > **Note:** You must have write access to the `omsconfig.properties` file in the `ORACLE_HOME\sysman\config` directory to run the `emca` command.

2. Click Next on the Welcome page.

   The Configuration Operation page appears.

3. Select "Upgrade an existing repository" from the list of configuration operations and press Next to continue.

   The Select Database for Repository page appears.

4. Log in to the database that contains the repository you want to upgrade.

> **Note:** In order to upgrade a repository, you must connect to the database as a user with DBA privileges. The repository schema user created by the Enterprise Manager Configuration Assistant will not have the necessary DBA privileges for this step. To avoid potential security issues, do not grant more privileges to your repository schema user than necessary. Connect to the database as a different user with DBA privileges instead.

The "Select Repository" page appears. If you are selecting a repository to upgrade, the Select Repository page shows only Release 2.0, 2.1, and 2.2 repositories. The Enterprise Manager Configuration Assistant does not display Release 9.0.1 repositories in this situation (they are already at the current version and do not need to be upgraded). The page shows the following information about the repository:

Username: The username of the repository.

Version: The version of the repository.

Type: The type of repository. Type can be either "Enterprise" or "Standalone". An Enterprise repository is used by the Oracle Enterprise Manager connected to a Management Server. A Standalone repository is required by certain applications when you use Oracle Enterprise Manager not connected to a Management Server.

5. Select a repository and click Next.

> **Note:** If the specified database does not contain any Release 2.x repositories, the list of repositories is empty and grayed out, and a message that "No repositories were found in the database" appears. Click Cancel to exit the Enterprise Manager Configuration Assistant or click Back to return to previous pages and connect to a different database.

The Repository Login Information page appears.

In order to perform a repository upgrade, you must log on to the repository database as the repository owner (the user so designated during repository creation). The repository user name is carried forward from the previous page, but you must enter the password.

6. Enter the repository user password.
7. Click Next to continue.

   The Upgrade Repository Summary page appears.

   All of the information you supplied is summarized on this page. Click Finish to perform the repository upgrade, or click Back to return to previous pages if you need to change any of the information.

   The Configuration Assistant Progress window appears, showing the processing performed and the processing steps that comprise the operation being performed. Each processing step is shown by a line of text.
8. To view detailed information in a text area, click Show Details. You can close the text area by clicking Hide Details.

   You can cancel the requested operation before it completes by clicking the Cancel button. However, if you cancel the operation, the repository will become unusable.

   The Cancel button changes to a Close button when processing completes, whether it is successful or not.

   When all of the steps have been completed without error, the "Processing completed." message appears.
9. Click Close.

   During the Configuration Assistant upgrade operation, the Oracle Management Service will be created (if it does not already exist) only if the repository being upgraded is the one actually being used by the local Management Server.
10. At the DOS prompt, issue this command:

    ```
    oemctl import registry ORACLE_HOME\sysman\admin\OMSIASRegistry.registry
    ```

Controlling the Management Server After Configuration

Once configured, the Management Server provides distributed control between clients and managed nodes. A central engine for notification, it processes all system management tasks and administers the distribution of these tasks across the enterprise.

> **See Also:** *Oracle9i Application Server Administrator's Guide* in the Oracle9*iAS* Documentation Library.

Migrating Oracle9*i*AS Single Sign-On

This section describes the process of migrating standalone Oracle9*i*AS Release 1 (1.0.2.2.x) applications to enable them for Single Sign-On in Oracle9*i*AS Release 2 (9.0.2).

1. Install the Oracle9*i*AS Release 2 (9.0.2) Infrastructure option, which installs the Single Sign-On (SSO) Server and Oracle Internet Directory (O*i*D).
2. Migrate user data for Oracle9*i*AS Release 1 (1.0.2.2.x) applications to the O*i*D Release 9.0.2 in order for SSO Server Release 9.0.2 to recognize them as valid users. If there are multiple user repositories in use, then you must first reconcile the multiple account names that a single corporate user may have, issue a unique account/username to each user, and then migrate these unique accounts into the O*i*D. See the *Oracle Internet Directory Migration Guide* and *OiD* Release 9.0.2 documentation for details on user provisioning & directory information tree (DIT).
3. Migrate the application logic to Oracle9*i*AS Containers for J2EE (OC4J).

   See Also: Chapter 3, "Migrating Internet Applications Components"

   See Also: *Oracle 9iAS Single Sign-On Administration Guide*

   See Also: *Oracle 9iAS Single Sign-On Developer's Guide*

   In Release 2, most applications are integrated with mod_sso (the Single Sign-On module for Oracle HTTP Server) and the OC4J security infrastructure.

After this process is complete, Release 1 applications can use the security infrastructure provided in Release 2.

Note: SSO Server Release 2 does not include migration of the server from Release 1.

Migrating Oracle Internet Directory

This section explains how to migrate Oracle Internet Directory (OID) Release 9.0.2.1.0 from the following Oracle Internet Directory releases:

- Oracle Internet Directory 2.1.1.x

- Oracle Internet Directory 3.0.1.x

Throughout these instructions, *ORACLE_HOME_1* represents the Oracle home of the existing Oracle Internet Directory and *ORACLE_HOME_2* represents the Oracle home of the newly installed Oracle Internet Directory 9.0.2.1.0. The migration and upgrade process is described below:

1. The supported version of OID is available in ORACLE_HOME_1.
2. An installation of OID 9.0.2.1.0 takes place in a different Oracle home, ORACLE_HOME_2, through the Infrastructure Installation type.
3. A migration of the database from the old OID installation must take place. This includes upgrading the database to the latest version and changing its parent Oracle home from ORACLE_HOME_1 to ORACLE_HOME_2. The binaries running the database will be coming from ORACLE_HOME_2 after this step.
4. The OID schema is upgraded to version 9.0.2.1.0.
5. Upgrade is complete.

Upgrade Considerations

The following conditions and procedures are important in planning and executing a successful upgrade.

- During the OID 9.0.2.1.0 installation in step 2., a database instance will be created in ORACLE_HOME_2. This database should NOT be used by other Oracle9*i*AS components as the Infrastructure. It should be discarded and deleted using Database Configuration Assistant (DBCA).
- At the end of upgrade, ORACLE_HOME_1 can be de-installed. Be careful if you choose to clean up the file system after de-installation. The location of the database and control files do not change after the upgrade, although it has been migrated to run from ORACLE_HOME_2. Make sure you leave these files alone when cleaning up.
- ORACLE_HOME_2 should only be used to support the upgraded OID. ORACLE_HOME_2 is not itself a complete Oracle9*i*AS Infrastructure. However, the OID running in ORACLE_HOME_2 can be used by other Oracle9*i*AS 9.0.2.1.0 installation that requires a 9.0.2.1.0 OID.

Pre-Upgrade Tasks

Before you begin the upgrade, you must perform these steps:

1. In ORACLE_HOME_1, where the existing OID is installed, stop all OID processes (OID Monitor, OID Server, Replication Server, Directory Integration Server). The corresponding database instance and the listener should remain running.

   Note: Oracle Corporation strongly recommends that you back up the database in ORACLE_HOME_1 before proceeding, thereby retaining existing schema information and data.

2. Remove the Oracle Internet Directory service from the Windows registry with the following command:

   ```
   ORACLE_HOME_1\bin\oidmon remove
   ```

3. Install OID 9.0.2.1.0 from the Oracle9*i*AS Release 2 (9.0.2) Installation CD into ORACLE_HOME_2. Ensure that you have applied all the required patches from the Patch CD in ORACLE_HOME_2.

4. When the installation completes, stop all of the OID processes running in ORACLE_HOME_2, as well as the corresponding infrastructure database and listener in ORACLE_HOME_2.

Upgrading in a Single Node Environment

The following table outlines the steps for upgrading OID in a single node environment.

OID Version	Database Version	Migration From ORACLE_HOME_1 to ORACLE_HOME_2	Database Migration to Version 9.0.1	9.0.1.3.0 Database Patch	OID Schema Upgrade
2.1.1.x	8.1.7.x	Done by Oracle Data Migration Assistant (ODMA)	Done by Oracle Data Migration Assistant	Manual Step before launching OIDCA (see details below)	OIDCA
3.0.1.x	9.0.1.x	Done by Oracle Internet Directory Configuration Assistant (OIDCA)	N / A	Manual Step – during execution of OIDCA (see details below)	OIDCA

Upgrading from 2.1.1.x to 9.0.2.1.0

This section describes the procedures you must perform to upgrade from OID version 2.1.1.x.

Migrating the Database from 8.1.7.x to 9.0.1.0.0

If you are upgrading from Oracle Internet Directory 3.0.1.x only, proceed to "Upgrading from 3.0.1.x to 9.0.2.1.0".

1. In ORACLE_HOME_2, launch the Oracle Data Migration Assistant (ODMA)by opening the **Start** menu, selecting **Programs**-->**ORACLE_HOME_2**-->**Configuration and Migration Tools**-->**Data Migration Assistant**. Follow the wizard to migrate the existing 8.1.7.x database in ORACLE_HOME_1, on which the old 2.1.1.x OID is running. Make sure the correct SID is specified and that you choose to migrate the database listener. For more information on Oracle Data Migration Assistant, please refer to the 9*i* (9.0.1.0.0) Database Migration Documentation.

Applying the 9.0.1.3.0 database patch

Ensure that the database is brought up in ORACLE_HOME_2. To apply the 9.0.1.3.0 database patch set, please follow the instructions in the “Post Install Actions section” in *ORACLE_HOME_2*`\rdbms\notes\patch_note.htm`.

The post-install steps in the RDBMS bundled patch must also be applied on the migrated database. Refer to the corresponding Readme file for information.

Oracle Internet Directory Configuration Assistant

In ORACLE_HOME_2, do the following:

1. Launch the Oracle Internet Directory Configuration Assistant (OIDCA) by executing *ORACLE_HOME_2*`\bin\oidca`.

   The Welcome screen appears.

2. Click Next.

3. Select the Upgrade an existing OID option and click Next.

   The Database Migration screen appears.

4. Supply the information about the database you are upgrading (the database on which the OID you intend to upgrade was running):

   - Database SID of the old OID database

- Passwords for the database users, SYSTEM and ODS
- Oracle home of the old OID version
- Location of the `init.ora` file for the old OID database (for example, `C:\oracle\database\initoiddb.ora`)
- Listener port for the OID database.
- Connect string for the OID database.

5. Click Next.

   The Upgrade in Progress window appears.

6. On the Oracle Internet Directory Credentials screen, supply the following information about the OID server:

   - Non-SSL port on which the OID server must be started. The default value specified is 389.
   - SSL port on which the OID server must be started. The default value specified is 636.
   - The super-user Distinguished Name (DN).
   - The corresponding super-user password.

7. Click Next.

   The Root Oracle context and the Directory Integration Platform-related information are upgraded.

8. On the Upgrading Subscriber screen, please supply the Distinguished Name (DN) that identifies the root of your organization (e.g. o=acme, dc=com). This domain will become the default subscriber node. A subscriber Oracle Context will be created under this subscriber.

   Note: Oracle Corporation recommends that you choose a domain with no existing Oracle Context or with an Oracle Context version 9.0.0.0.0 or higher. A 9.0.0.0.0 or newer Oracle Context can be upgraded. However, an 8.1.6.0.0 Oracle Context will not be upgraded.

9. Click Next.

   The User Data Migration screen appears.

10. Select Yes if you want to perform a user data migration as part of the OIDCA. It is strongly recommended you do this as a post-upgrade step if you have a large directory (>10,000 users). See "User Data Upgrade" for more details on performing user migration outside of OIDCA.

    The upgrade is complete.

11. Exit from Oracle Internet Directory Configuration Assistant.

    The OID is running, listening to the specified non-SSL and SSL ports.

Upgrading from 3.0.1.x to 9.0.2.1.0

This section describes the procedures you must perform to upgrade from OID version 3.0.1.x.

1. Launch the Oracle Internet Directory Configuration Assistant (OIDCA) by executing *ORACLE_HOME_2*\bin\oidca.

   The Welcome screen appears.

2. Click Next.

3. Select the Upgrade an existing OID option and click Next.

4. The Database Migration screen appears.

5. Supply the information about the database you are upgrading (the database on which the OID you intend to upgrade was running):

   - Database SID of the old OID database
   - Passwords for the database users, SYSTEM and ODS
   - Oracle home of the old OID version
   - Location of the INIT.ORA file for the old OID database (for example, C:\oracle\database\initoiddb.ora)
   - Listener port for the OID database.
   - Connect String for the OID database.

6. Click Next.

   The OID database is migrated from ORACLE_HOME_1 to ORACLE_HOME_2.

7. Apply the 9.0.1.3.0 patch by performing the steps below in a separate window.

a. If you are upgrading from OID 3.0.1.x, stop the database in ORACLE_HOME_1 if it is still running, and bring it up from the ORACLE_HOME_2 environment to complete the database migration.

b. Stop the listener in ORACLE_HOME_1.

c. Copy the listener entry for the old OID database from *ORACLE_HOME_1*\network\admin\tnsnames.ora to ORACLE_HOME_2\network\admin\tnsnames.ora.

d. Copy the listener entry for the old OID database from *ORACLE_HOME_1*\network\admin\listener.ora to *ORACLE_HOME_1*\network\admin\listener.ora.

e. Start the listener in ORACLE_HOME_2.

f. To apply the 9.0.1.3.0 database patch set, follow the instructions in the "Post Install Actions" section in *ORACLE_HOME_2*\rdbms\notes\patch_note.htm.

g. The post-install steps in the RDBMS bundled patch must also be applied on the migrated database. Refer to the corresponding Readme file for information.

h. Click Next.

The Upgrade in Progress window appears.

8. On the Oracle Internet Directory Credentials screen, supply the following information about the OID server:

- Non-SSL port on which the OID server needs to be started. The default value specified is 389.
- SSL port on which the OID server needs to be started. The default value specified is 636.
- The super-user Distinguished Name (DN)
- The corresponding super-user password.

9. Click Next.

The Root Oracle context and the Directory Integration Platform-related information are upgraded.

10. On the Upgrading Subscriber screen, supply the Distinguished Name (DN) that identifies the root of your organization (e.g. o=acme, dc=com). This domain will

become the default subscriber node. A subscriber Oracle Context will be created under this subscriber.

Note: Oracle Corporation recommends that you choose a domain with no existing Oracle Context or with an Oracle Context version 9.0.0.0.0 or higher. A 9.0.0.0.0 or newer Oracle Context can be upgraded. However, an 8.1.6.0.0 Oracle Context will not be upgraded.

---

11. Click Next.

    The User Data Migration screen appears.

12. Select “yes” if you want to perform a user data migration as part of the OIDCA. Oracle Corporation recommends that you do this as a post-upgrade step if you have a large directory (>10,000 users). See "User Data Upgrade" for more details on performing user migration outside of OIDCA.

    The upgrade is complete.

13. Exit from Oracle Internet Directory Configuration Assistant.

    The OID server is running, listening to the specified non-SSL and SSL ports.

Upgrading in a Multi-Node Environment

Upgrading a multi-node OID system in a replication environment requires special attention. This section discusses the two ways to upgrade a multi-node OID system.

Upgrading One Node at a Time

This method avoids a complete shutdown of the replicated network. An upgrade on one node can take place, while the other nodes remain available. Note the following:

- The upgrade is not complete until all the nodes are upgraded. However, during this period, all network nodes, except the one being upgraded, remain available.
- You must perform the upgrade on the master definition site (MDS) before you upgrade the other sites.
- During the upgrade, only one node should be in the Read-Write mode. The rest should all be set to the Read-Only mode.

1. Change a node to become Read-Only by doing the following:
   a. Create an input file `inputfile.ldif` as follows:

      ```
      dn:
      changetype: modify
      replace: orclservermode
      orclservermode:r
      ```

   b. Use the following command to change the node to Read-Only.

      ORACLE_HOME_1`/bin/ldapmodify -D` *super-user DN* `-w` *super-user password* `-h` *hostname* `-p` *port* `-f inputfile.ldif`

2. Perform the upgrade on each node, repeating the following steps on each node:
   a. Stop all OID processes.
   b. Delete ASR push jobs temporarily by running *ORACLE_HOME_1*`\ldap\admin\delasrjobs.sql`. This script deletes Oracle9*i* Replication jobs on other master sites that push changes to the MDS. Deleting these jobs temporarily removes the current node from the replication environment so that no changes can be applied to it. Other nodes, however, remain operational and continue replicating changes.
3. Upgrade the node to OID 9.0.2.1.0 by following the steps outlined in "Upgrading One Node at a Time" on page 7-13 and "Backward Compatibility in a Replicated Environment" on page 7-14.
4. After the upgrade, ensure that OID database, listener and OID processes (OID monitor, OID server, Directory Integration Server) are all running.
5. Create ASR push jobs again by running *ORACLE_HOME_2*`/ldap/admin/delasrjobs.sql`. The ASR jobs deleted will be restored. Changes in the upgraded node will be pushed to the rest of the network and vice versa. The upgraded node is now back in the replicated network.

Backward Compatibility in a Replicated Environment

As discussed previously, the upgrade is not complete until all nodes are upgraded.

When an existing Directory Replication Group (DRG) is being upgraded, some of the changes made on the newly upgraded OID 9.0.2.1.0 will not replicate to nodes of the older version that have not been upgraded. These changes will eventually be replicated successfully when the consumer node is upgraded to 9.0.2.1.0.

If possible, apply the following restrictions to the DRG during upgrade:

- Do not perform any LDAP operations on an upgraded node in the DRG unless all the nodes in DRG are upgraded.
- If you do need to perform updates on the upgraded node, DO NOT push the changes to the other nodes unless they are upgraded.

To do this, you can temporarily disable the pushing of changes by a particular node by bringing up the replication server in a special mode (-o FALSE). To start the replication server in this mode, execute the following:

```
ORACLE_HOME\bin\oidctl connect=connect string server=oidrepld instance=1
flags="-p port -h host -o FALSE " start
```

Note that any the changes made on an older node can be successfully replicated to any newer v.9.0.2.1.0 node.

Upgrading All Nodes at the Same Time

Backward compatibility issues can be avoided by upgrading all the nodes at one time. However, this method requires a complete shutdown of the replication network, which leads to downtime. If downtime is acceptable, this method is preferable.

This method requires all the nodes to become Read-Only, and an eventual shutdown of all the OID processes on all the nodes. This replicated network becomes unavailable during the upgrade process.

1. Change a node to become Read-Only by doing the following:
   a. Create an input file `inputfile.ldif` as follows:

      ```
      dn:
      changetype: modify
      replace: orclservermode
      orclservermode:r
      ```

   b. Use the following command to change the node to Read-Only.

      ```
      ORACLE_HOME_1\bin\ldapmodify -D super-user DN -w super-user password -h
      hostname -p port -f inputfile.ldif
      ```

2. When the change log queue is empty on each node, stop the OID server and all OID processes and the corresponding database on each node.
3. Upgrade each node to OID 9.0.2.1.0 by following the steps outlined in "Upgrading One Node at a Time" on page 7-13 and "Backward Compatibility in a Replicated Environment" on page 7-14.

Post-Upgrade Tasks

After the upgrade, perform the following tasks:

Set the JOB_QUEUE_PROCESSES Parameter

On all the nodes in the DRG, please make sure that the *JOB_QUEUE_PROCESSES* parameter in the init.ora file of the database is set to the following value:

- For a single-node environment, it should be set to at least 1.
- For a multi-node environment, it should be set to the number of nodes.

User Data Upgrade

You must do this if you are performing a single-node upgrade. In a multi-node environment, this only needs to be done on the Master Definition Site (MDS). Skip this section if you have performed user data migration in OIDCA.

Password Conversion

The password format in OID 9.0.2.1.0 is base-64. The older passwords stored in hexadecimal must be converted. To perform the conversion, follow these steps:

1. Use the command below to perform an `ldapsearch` to output all the encrypted user passwords to a file. In this case, *ORACLE_HOME_2*`\ldap\install\pwdin.ldif` is used as the output file.

   ```
   ORACLE_HOME_2\bin\ldapsearch –L –h OID host name -p OID Non-SSL port -D OID
   Super User DN-w OID Super User Password -b “” –s sub “objectclass=*” dn
   userpassword > ORACLE_HOME_2\ldap\install\pwdin.ldif
   ```

2. Issue the command below to use the passwordconvert tool to convert the userpasswords in *ORACLE_HOME_2*`\ldap\install\pwdin.ldif` and output it to *ORACLE_HOME_2*`\ldap\install\pwdout.ldif`.

   ```
   ORACLE_HOME_2\bin\passwordconvert –m hex2base64 –f modify ORACLE_HOME_
   2\ldap\install\pwdin.ldif  ORACLE_HOME_2\ldap\install\pwdout.ldif
   ```

3. Issue the command below to use ldapmodify to upload the BASE-64 encoded userpasswords in *ORACLE_HOME_2*`\ldap\install\pwdout.ldif` back into OID.

   ```
   ORACLE_HOME_2\bin\ldapmodify –h OID host name -p OID Non-SSL port -D OID
   Super User DN-w OID Super User Password> -f ORACLE_HOME_
   2\ldap\install\pwdout.ldif
   ```

Oracle Context Configuration

You can use Oracle Directory Manager (ODM) to perform the Oracle Context configuration.

Root Oracle Context Configuration

The following information needs to be added in the Root Oracle Context under the DN *"cn=Common, cn=Products, <Root Oracle Context DN>"*. By default, the Root Oracle Context DN is "*cn=OracleContext*". The following attribute values are needed:

- `orclSubscriberSearchBase` - Identifies the base node in the DIT when searching for subscribers.
- `orclSubscriberNickNameAttribute` - Identifies the nickname attribute to be used when searching for a subscriber under the subscriber search base.
- `orclDefaultSubscriber` - Identifies the root of your organization. It should be the same value as the one specified in "Upgrading from 3.0.1.x to 9.0.2.1.0" on page 7-11.

Default Subscriber Oracle Context Configuration

The following information must be added in the subscriber-specific Oracle Context under the DN *"cn=Common, cn=Products, cn=oracleContext, <Subscriber DN>"*.

- `orclCommonUserSearchBase` - Identifies the base node under the subscriber when searching for users. During upgrade, this attribute value is set as the subscriber DN.

  Note: If this attribute is not set, then the password policy under the Root Oracle Context will be applied.

- `orclCommonNickNameAttribute` - Identifies the base node under the subscriber when searching for groups.
- `orclCommonGroupSearchBase` - Identifies the node in the DIT under which all the groups are placed.

  See Also: *Oracle Internet Directory Administrator's Guide*

Password Policy Configuration

If a password policy exists in the older version of OID (which would be located under the DN `cn=pwdpolicyentry, cn=oracle internet directory`), this policy will be applied to both the Root Oracle Context and the default Subscriber Oracle Context. The original DN containing the policy, `cn=pwdpolicyentry, cn=oracle internet directory` will be removed in the earlier version.

Otherwise, the default password policy is set up as part of the Subscriber Oracle Context creation. By default, the password policy for the default subscriber is set to the following values:

- The user passwords expire in 60 days (pwdmaxage=5184000)
- Account locked out after 10 successive failed login attempts (pwdlockout=1 and pwdmaxfailure=10)
- Password syntax checking is enabled and the minimum length of userpasswords is 5 (pwdchecksyntax=1 and pwdminlength = 5)
- User passwords must contain at least 1 numeric character (orclpwdalphanumeric=1)

  (You can find the above attribute values in the entry `cn=PwdPolicyEntry, cn=Common,cn=Products,cn=oracleContext,`*`subscriber DN`*)

The password policy under Root Oracle Context applies to all entities under the root DSE. However, it does not apply to entities under the Root Oracle Context.

See Also: *Oracle Internet Directory Administrator's Guide*

If the upgraded OID is integrating with other Oracle9*i*AS Release 2 (9.0.2) components, appropriate access control policies (ACPs) will need to be set up to grant necessary privileges to the components.

See Also: *Oracle9i Application Server Security Guide*

Post-Upgrade Manual Tasks and Database Migration Alternatives

This section describes manual updates you must perform after upgrading, and an alternative to migrating the database.

Server Manageability

After the upgrade, the OID target in `targets.xml` must be updated manually to specify the correct ORACLE_SID, and the ods and emd administrator user passwords.

See Also: *Oracle Internet Directory Release Notes*

Directory Integration Server

During the upgrade, the Directory Integration Platform (DIP) server is not brought up. In order to use the Directory Integration Platform, you need to explicitly register and start the DIP server.

See Also: *Oracle Internet Directory Administrator's Guide*

Post-Upgrade Step for *i*Planet Synchronization

Integration Profiles for *i*Planet synchronization, namely `iPlanetImport` and `iPlanetExport`, are created as part of the upgrade. These profiles must be added to `configset1` to be configured and used for synchronization.

If the profiles are available in `Configset 1` as part of upgrade, it will be shown in ODM under Integration Server as part of Configuration Set1. If the profiles are not available, add them using `LDAPMODIFY` as below:

```
ldapmodify -h OID Host -p OID Port -D OID Super-user -w OID Super-user password
-f ORACLE_HOME/ldap/install/upgdip.ldif
```

Database Import/Export Style Upgrade

An upgrade procedure using database import/export is also available. It offers a more flexible way of migrating OID from version to version. Database migration is not required.

See Also: *Oracle Internet Directory Administrator's Guide*

8

Migrating e-Business Integration Components

This chapter explains how to migrate e-business integration components. It contains these sections:

Migrating Oracle9*i*AS InterConnect

This section explains how to migrate Oracle9*i*AS InterConnect. (The schema is in the new Release 2 (v9.0.2) database.)

Migrating Hub Components

1. Configure the Release 2 installation of Oracle9*i*AS InterConnect to match the Release 1 (v1.0.2.2) installation. Note that:
   - The infrastructure database of Release 2 corresponds to the hub database for Release 1.

- The Oracle9*i*AS middle tier Oracle home corresponds to the Oracle home in which the repository in Oracle9*i*AS Release 1 (1.0.2.2.x) is installed.

2. Install the Oracle9*i*AS InterConnect Hub in the Oracle9*i*AS middle tier Oracle home.

   The Oracle9*i*AS InterConnect Repository, Workflow, and Workflow Communication are now installed. By default, these components point to the schemas in the infrastructure database.

Migrating Metadata

1. Run the `oaiexport` script provided with the Release 1 installation. Supply values for repository name, file name, system password, and connect string in the command:

   ```
   ORACLE_HOME_1\oai\4.1\repository\repository name\oaiexport file name
   system/system password connect string
   ```

   The metadata is exported to a file in the current directory.

2. Run the oaiimport script provided with the Release 2 installation. Supply values for: repository name, file name, from user (the user id of the user whose metadata is being imported), system password, oaihub902 schema password, and connect string in the command:

   ```
   ORACLE_HOME_2\oai\9.0.2\repository\oaiimport file name from user
   system/system password oaihub902 schema password connect string
   ```

   The file is imported into the Oracle9*i*AS Release 2 Infrastructure Database.

3. Using SQLPlus, connect to the hub schema in the Infrastructure Database and execute the commands:

   ```
   update emd set type='AQ' where type='XML'
   commit;
   ```

Migrating Adapters

1. Install the Oracle9*i*AS InterConnect 902 adapter, which corresponds to the existing adapter that you have. The prompts for hub database information refer to the Oracle9*i*AS Release 2 Infrastructure Database.

2. When prompted for all other information, provide the values from your existing configuration.

Migrating *i*Studio and SDKs

1. Install the Oracle9iAS InterConnect Developer Kits 902. This includes the new version of iStudio and the SDKs. As you create an iStudio project, the prompts for hub database information refer to the Oracle9*i*AS Release 2 Infrastructure Database.

Migrating Management

1. Install the Oracle Enterprise Manager Console 902.

   A correctly configured Oracle Enterprise Manager for Oracle9*i*AS InterConnect is now available.

Migrating Oracle Workflow

To migrate Oracle Workflow from Oracle9*i*AS Release 1 (v1.0.2.2) to Release 2 (v 9.0.2), perform the following steps:

1. Install Oracle Workflow with Oracle9*i*AS Release 2 (9.0.2), including all pre- and post- installation steps as described in the *Oracle9i Application Server Installation Guide*.

   The installation updates the Oracle Workflow server version in the database. See the *Oracle9i Application Server Installation Guide* for detailed instructions.

2. Perform all Workflow setup steps for your Release 2 installation as described in the Setting Up Oracle Workflow chapter of the Oracle Workflow Guide. In particular, ensure that you:

   - Set your global Workflow preferences appropriately
   - Set up a directory service for Oracle Workflow
   - Create a new configuration file for the Notification Mailer that contains the parameters with which you want to run the Notification Mailer.

3. Copy any customized files from *ORACLE_HOME_1* to *ORACLE_HOME_2*. Such files may include the following:

   - Workflow process definition files (`.wft` files located in *ORACLE_HOME_1*`\wf\res\`*lang*)
   - Business Event System definition files (`.xml` files located in *ORACLE_HOME_1*`\wf\res\`*lang*)
   - SQL scripts (`.sql` files located in *ORACLE_HOME_1*`\wf\sql`)

- Custom help files (`.htm` files located in *`ORACLE_HOME_1`*`\wf\doc\`*`lang`*`\wfcust` or `.hlp` files located in *`ORACLE_HOME_1`*`\wf\res\`*`lang`*)
- Any other files containing customizations

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X