

Oracle® Application Server 10g

High Availability Guide

10g (9.0.4)

Part No. B10495-02

March 2004

Oracle Application Server 10g High Availability Guide, 10g (9.0.4)

Part No. B10495-02

Copyright © 2003, 2004 Oracle. All rights reserved.

Primary Author: Kai Li, Thomas Van Raalte

Contributor: Jay Feenan, Shari Yamaguchi, Ashesh Parekh, Susan Kornberg, Pradeep Bhat, Ashish Prabhu, Mukul Paithane, Wei Hu, Wayne Milsted, Jerry Bortveldt, Michael Moon, David Rowlands, Paul Mackin, Wes Root

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Send Us Your Comments	ix
Preface	xi
Intended Audience	xi
Documentation Accessibility	xi
Organization	xii
Related Documents	xii
Conventions	xiii
1 Introduction	
What is High Availability	1-1
High Availability in Oracle Application Server 10g	1-1
Types of Failures	1-3
Organization of this Guide	1-4
High Availability Information in Other Documentation	1-4
2 Middle Tier High Availability	
OracleAS Middle Tier Overview	2-1
OracleAS Middle Tier Terminology	2-2
Services Available	2-3
J2EE	2-3
HTTP	2-3
Portal	2-3
Business Intelligence	2-3
Oracle Application Server Forms Services	2-4
Single Sign-On	2-4
Caching	2-4
Features and Components for Middle Tier High Availability	2-5
Oracle Application Server Instance High Availability	2-5
Oracle Application Server Clusters	2-6
Types of Oracle Application Server Clusters	2-7
Cluster-Wide Configuration for Oracle Application Server Clusters that are Managed Using a Repository	2-9
Requirements for Oracle Application Server Instances to Join Oracle Application Server Clusters that are Managed Using a Repository	2-10

Properties of Oracle Application Server Instances in Oracle Application Server Clusters that are Managed Using a Repository	2-10
Oracle Application Server Web Cache Clusters	2-11
OC4J Islands.....	2-11
Web Application Session State Replication with OC4J Islands	2-12
Web Application Session State Protecting Against Software Problems	2-12
Web Application Session State Replication Protecting Against Hardware Problems.....	2-13
Configuring OC4J Islands With High Availability.....	2-14
Stateful Session EJB High Availability Using EJB Clustering	2-14
JNDI Namespace Replication.....	2-15
OC4J Distributed Caching Using Java Object Cache	2-15
Process Monitoring and Restart.....	2-15
Oracle Process Manager.....	2-16
Oracle Notification System.....	2-16
High Availability Through Distributed Configuration.....	2-16
Other High Availability Components.....	2-17
Improving Availability with an External Load Balancer	2-17
Types of External Load Balancers	2-17
High Availability Benefits of External Load Balancing.....	2-18
Improving Availability with Operating System Clusters	2-18
HTTP Service High Availability	2-18
Web Cache and Oracle HTTP Server High Availability Summary	2-18
OC4J Load Balancing Using mod_oc4j	2-19
OC4J Load Balancing Using Local Afinity and Weighted Routing Options	2-20
Choosing a mod_oc4j Routing Algorithm.....	2-21
J2EE High Availability	2-21
EJB Client Routing	2-21
Oracle Application Server Portal High Availability	2-22
Oracle Application Server Wireless High Availability.....	2-23
Business Intelligence High Availability.....	2-24
Oracle Application Server Reports Services High Availability.....	2-24
High Availability Solution.....	2-24
Oracle Application Server Discoverer High Availability.....	2-25
Oracle Application Server Forms Services High Availability.....	2-25
Oracle Application Server Integration High Availability.....	2-26
Middle Tier Recovery Solutions.....	2-27
Restarting Processes.....	2-27
Restoring from Cold Backup	2-27
Restoring from Online Backup.....	2-27
Disaster Recovery.....	2-28
DCM Archive/Recover	2-28
Configuration Cloning	2-29
3 Infrastructure High Availability	
Oracle Application Server 10g Infrastructure Overview	3-1
Oracle Application Server 10g Infrastructure Components.....	3-2

Oracle Application Server Metadata Repository	3-2
When to Use Oracle Application Server Metadata Repository	3-3
Oracle Identity Management	3-3
Oracle Internet Directory	3-4
Oracle Application Server Single Sign-On	3-4
Oracle HTTP Server	3-4
Oracle Application Server Containers for J2EE (OC4J)	3-5
Oracle Enterprise Manager - Application Server Console	3-5
High Availability Configurations for Infrastructure	3-5
Oracle Application Server Cold Failover Clusters	3-7
Terminology	3-7
Hardware Cluster	3-7
Failover	3-7
Primary Node	3-7
Secondary Node	3-7
Logical or Virtual IP	3-7
Virtual Hostname	3-8
Shared Storage	3-8
Architecture (UNIX)	3-8
Architecture (Windows)	3-10
Middle Tier on OracleAS Cold Failover Cluster Nodes	3-12
Oracle Application Server Active Failover Cluster (UNIX)	3-14
Load Balancer Configuration	3-16
4 Managing and Operating Middle Tier High Availability	
Middle Tier High Availability Configuration Overview	4-1
Configuration Overview OracleAS Clusters Managed Using a Repository	4-2
Oracle Application Server Clusters Managed Using Database Repository	4-2
Oracle Application Server Clusters Managed Using File-Based Repository	4-2
Common Tasks for OracleAS Cluster Configuration	4-3
Manually Configured OracleAS Clusters Configuration Overview	4-3
OracleAS Web Cache Cluster Overview	4-4
Managing and Configuring OracleAS Clusters	4-4
Creating and Managing OracleAS Clusters	4-4
Associating an Instance with a Farm	4-5
Associating an Instance to be Managed Using a Database Repository	4-5
Associating an Instance to be Managed Using a File-Based Repository	4-5
Creating OracleAS Clusters Using Application Server Console	4-5
Managing OracleAS Clusters Using Application Server Console	4-6
Managing Application Server Instances in an OracleAS Cluster	4-7
Adding an Application Server Instance to an OracleAS Cluster	4-7
Removing an Application Server Instance from an OracleAS Cluster	4-8
Using a File-Based Repository with OracleAS Clusters	4-9
Initializing File-Based Repository Host and Adding Instances to a Farm	4-9
Testing an Instance With whichFarm and Leaving a Farm	4-9
Initializing the Repository Host Instance for a File-Based Repository	4-11
Joining a Farm Managed Using a File-Based Repository	4-11

Managing Instances in a Farm That Uses a File-Based Repository	4-12
Managing Oracle Application Server Instances and Clusters With a File-Based Repository	4-12
Availability Issues for OracleAS Clusters With a File-Based Repository	4-12
Exporting and Importing Configuration Information With a File-Based Repository ...	4-13
Moving an Instance Between Repositories	4-14
Moving to a Database-Based Repository	4-14
Moving to Another File-Based Repository	4-15
Enabling SSL For Communication Between Instances That are Using a File-Based Repository	4-15
Generating the Keystore	4-16
Shutdown Oracle Application Server Processes on Each Instance	4-16
Set Up the Keystore Information File on Each Instance in the Farm	4-16
Enable SSL By Configuring dcmCache.xml.....	4-16
Verify that Configuration Changes are Effected	4-17
Start Each Instance in the Farm	4-17
Adding a New Instance to a SSL-Enabled Farm.....	4-17
OC4J Configuration with an OracleAS Cluster	4-18
Overview of OracleAS Cluster Configuration for OC4J Instances.....	4-18
Cluster-Wide Configuration Changes and Modifying OC4J Instances	4-19
Creating or Deleting OC4J Instances on OracleAS Clusters.....	4-19
Deploying Applications on OracleAS Clusters	4-20
Configuring Web Application State Replication for OracleAS Clusters	4-20
Configuring EJB Application State Replication for OracleAS Clusters	4-21
Configuring Stateful Session Bean Replication for OracleAS Clusters.....	4-22
End of Call Replication	4-22
JVM Termination Replication	4-23
Configuring OC4J Instance-Specific Parameters.....	4-23
Configuring OC4J Islands and OC4J Processes.....	4-23
Configuring Port Numbers and Command Line Options.....	4-24
Oracle HTTP Server Configuration with OracleAS Clusters	4-25
mod_oc4j Load Balancing With OracleAS Clusters	4-25
Load Balancing Overview	4-25
Setting Load Balancing Options	4-26
Configuring Oracle HTTP Server Instance-Specific Parameters.....	4-26
Security – Configuring Single Sign-On.....	4-27
Advanced Clustering Configuration	4-29
Routing Between Instances in Same Farm	4-29
Routing Between Instances Across Firewalls.....	4-30
Opening Intranet Communication through the OracleAS Port Tunnel	4-31
Opening OracleAS Ports To Communicate Through Intranet.....	4-32
5 Managing Infrastructure High Availability	
Oracle Application Server Cold Failover Clusters	5-1
Starting Up	5-1
Stopping.....	5-2
Oracle Application Server Active Failover Cluster (UNIX).....	5-3

Starting Up	5-3
Shutting Down.....	5-5
Monitoring	5-5
Failing Over During an Outage	5-6
Restoring Resiliency After an Outage	5-7
Synchronizing Configuration Files Using the Oracle Application Server Active Failover Cluster Runtime Control Utility (afcctl)	5-7
Setting Up afcctl	5-8
Obtain the afcctl Utility	5-8
Install the afcctl Utility	5-8
Using afcctl.....	5-8
Setting the Default Baseline Timestamp.....	5-9
Synchronizing Files From a Node to Other Nodes in an OracleAS Active Failover Cluster	5-9
Listing Modified Files on a Node Since the Last Synchronization.....	5-10
Excluding Specific Configuration Files from Synchronization.....	5-11
Example	5-11
Best Practises for Using afcctl.....	5-12

6 Oracle Application Server Disaster Recovery

Oracle Application Server 10g Disaster Recovery Solution	6-2
Terminology.....	6-2
Requirements	6-3
Topology.....	6-4
Setting Up the OracleAS Disaster Recovery Environment	6-5
Planning and Assigning Hostnames	6-6
Physical Hostnames.....	6-8
Logical Hostnames.....	6-9
Virtual Hostname.....	6-9
Configuring Hostname Resolution.....	6-9
Using Local Hostnaming File Resolution.....	6-10
Using DNS Resolution	6-11
Additional DNS Server Entries for Oracle Data Guard	6-13
Secure Shell (SSH) Port Forwarding.....	6-14
Installing Oracle Application Server 10g Software	6-14
Setting Up Oracle Data Guard	6-15
Enable ARCHIVELOG Mode for Production Database.....	6-16
Identifying the Production Database Datafiles	6-18
Make a Copy of the Production Database.....	6-18
Create a Control File for the Standby Database	6-19
Prepare the Initialization Parameter File to be Copied to the Standby Database	6-19
Copy Files from the Production System to the Standby System	6-19
Set Initialization Parameters for the Physical Standby Database	6-21
Create a Windows Service (for Microsoft Windows systems)	6-21
Create a New Password File on the Standby System	6-21
Configure Listeners for the Production and Standby Databases.....	6-22
Enable Dead Connection Detection on the Standby System	6-22

Create Oracle Net Service Names.....	6-22
Create a Server Parameter File for the Standby Database	6-23
Start the Physical Standby Database	6-24
Enable Archiving to the Physical Standby Database.....	6-24
Start Remote Archiving.....	6-24
Verify the Physical Standby Database	6-24
Synchronizing Baseline Installation with Standby Site	6-26
Backing Up Production Site	6-26
Shipping Infrastructure Database Archive Logs	6-27
Backing Up Configuration Files (Infrastructure and Middle Tier).....	6-28
Restoring to Standby Site	6-29
Restoring Configuration Files (Infrastructure and Middle Tier)	6-29
Restoring the Infrastructure Database - Applying Log Files	6-30
Scheduled Outages	6-31
Site Switchover Operations.....	6-32
Unplanned Outages	6-35
Site Failover Operations	6-35
Setting Up the New Standby Database.....	6-37
Wide Area DNS Operations	6-37
Using a Wide Area Load Balancer.....	6-37
Manually Changing DNS Names	6-38

A Setting Up a DNS Server

Index

Send Us Your Comments

Oracle Application Server 10g High Availability Guide, 10g (9.0.4)

Part No. B10495-02

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

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

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

- Electronic mail: appserverdocs_us@oracle.com
- FAX: 650-506-7375 Attn: Oracle Application Server Documentation Manager
- Postal service:

Oracle Corporation
Oracle Application Server Documentation
500 Oracle Parkway, M/S 10p6
Redwood Shores, CA 94065
USA

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

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

Preface

This preface contains these topics:

- Intended Audience
- Documentation Accessibility
- Organization
- Related Documents
- Conventions

Intended Audience

Oracle Application Server 10g High Availability Guidet is intended for administrators, developers, and others whose role is to deploy and manage Oracle Application Server 10g with high availability requirements.

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

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

Organization

The following chapters make up this guide:

Chapter 1, "Introduction"

This chapter provides an introduction to high availability with Oracle Application Server 10g.

Chapter 2, "Middle Tier High Availability"

This chapter provides a description of high availability in the Oracle Application Server 10g middle tier.

Chapter 3, "Infrastructure High Availability"

This chapter describes the high availability solutions available for the Oracle Application Server 10g Infrastructure.

Chapter 4, "Managing and Operating Middle Tier High Availability"

This chapter provides instructions to manage and operate the middle tier high availability environment.

Chapter 5, "Managing Infrastructure High Availability"

This chapter provides instructions to set up and manage the Infrastructure high availability solutions.

Chapter 6, "Oracle Application Server Disaster Recovery"

This chapter describes the disaster recovery solution for OracleAS. The solution covers both the middle and Infrastructure tiers.

Appendix A, "Setting Up a DNS Server"

This appendix provides instructions for setting up a DNS server relevant to the Oracle Application Server Disaster Recovery solution.

Related Documents

For more information, see these Oracle resources:

- Oracle Application Server Documentation Library
- Oracle Application Server Platform-Specific Documentation on Oracle Application Server Disk 1

Printed documentation is available for sale in the Oracle Store at

<http://oraclestore.oracle.com/>

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

<http://otn.oracle.com/membership/>

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

<http://otn.oracle.com/documentation/>

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 for Microsoft Windows Operating Systems

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 .
<i>Italics</i>	Italic typeface indicates book titles or emphasis.	<i>Oracle9i Database Concepts</i> Ensure that the recovery catalog and target database do <i>not</i> 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.
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 italic monospace (fixed-width) font	Lowercase italic monospace font represents placeholders or variables.	You can specify the <i>parallel_clause</i> . Run <i>Old_release</i> .SQL where <i>old_release</i> 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 (<i>digits</i> [, <i>precision</i>])
{ }	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]
...	Horizontal ellipsis points indicate either: <ul style="list-style-type: none"> 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 <i>subquery</i> ; SELECT <i>col1</i> , <i>col2</i> , ... , <i>coln</i> 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;
<i>Italics</i>	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/ <i>system_password</i> DB_NAME = <i>database_name</i>
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;

Conventions for Microsoft Windows Operating Systems

The following table describes conventions for Microsoft Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Oracle Database Configuration Assistant, choose Start > Programs > Oracle - <i>HOME_NAME</i> > Configuration and Migration Tools > Database Configuration Assistant.

Convention	Meaning	Example
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt "\"system32 is the same as C:\WINNT\SYSTEM32
C:\>	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the <i>command prompt</i> in this manual. The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	C:\oracle\oradata> C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\" C:\>imp SYSTEM/password FROMUSER=scott TABLES=(emp, dept)
HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	C:\> net start OracleHOME_ NAMETNSListener

Convention	Meaning	Example
<i>ORACLE_HOME</i> and <i>ORACLE_BASE</i>	<p>In releases prior to Oracle8i release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level <i>ORACLE_HOME</i> directory that by default used one of the following names:</p> <ul style="list-style-type: none"> ■ C:\orant for Windows NT ■ C:\orawin95 for Windows 95 ■ C:\orawin98 for Windows 98 <p>This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level <i>ORACLE_HOME</i> directory. There is a top level directory called <i>ORACLE_BASE</i> that by default is C:\oracle. If you install Oracle9i release 1 (9.0.1) on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is C:\oracle\ora90. The Oracle home directory is located directly under <i>ORACLE_BASE</i>.</p> <p>All directory path examples in this guide follow OFA conventions.</p> <p>Refer to <i>Oracle9i Database Getting Starting for Windows</i> for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.</p>	Go to the <i>ORACLE_BASE\ORACLE_HOME\rdbms\admin</i> directory.

Introduction

In this release of Oracle Application Server 10g, 10g (9.0.4), work has been done to improve and extend the high availability solutions for Oracle Application Server. Several new solutions for the Oracle Application Server 10g Infrastructure have been tested and are described in this book. All of these solutions seek to ensure that applications that you deploy on Oracle Application Server 10g meet the required availability to achieve your business goals. The solutions and procedures described in this book seek to eliminate single points of failure of any Oracle Application Server components with no or minimal outage in service.

This chapter explains high availability and its importance from the perspective of Oracle Application Server.

What is High Availability

The availability of a system or any component in that system is defined by the percentage of time that it works normally. A system works normally when it meets its correctness and performance specifications. For example, a system that works normally for twelve hours per day is 50% available. A system that has 99% availability is down 3.65 days per year on average. System administrators can expect critical systems to have 99.99% or even 99.999% availability. This means that the systems experience as little as four to five minutes of downtime per year.

Availability may not be constant over time. For example, availability may be higher during the daytime when most transactions occur, and lower during the night and on weekends. In the event of an unexpected disaster, such as a fire or earthquake, a system may go down suddenly for a period of time. However, because the Internet provides a global set of users, it is a common requirement that systems always be available.

Redundant components can improve availability, but only if a spare component takes over immediately for a failed component. If it takes ten minutes to detect a component failure and twenty additional minutes to start the spare component, then the system experiences a 50% reduction in availability for that hour of service.

Oracle Application Server is designed to provide a wide variety of high availability solutions, ranging from load balancing and basic clustering to providing maximum system availability during catastrophic hardware and software failures.

High Availability in Oracle Application Server 10g

Oracle Application Server consists of many components that can be deployed in distributed topologies. The underlying paradigm used to enable high availability for Oracle Application Server is clustering, which unites various Oracle Application

Server components in certain permutations to offer scalable and unified functionality, and redundancy should any of the individual components fail.

Before you continue, we recommend that you read the book *Oracle Application Server 10g Concepts* to gain an understanding of the different components in Oracle Application Server. The descriptions there will allow you to understand the rest of the text in this guide more efficiently.

Oracle Application Server has several solutions and techniques to achieve high availability, which are all described in this guide. They allow you to achieve the following goals:

- **Redundancy**

A highly available system requires its sub-systems to be redundant. All Oracle Application Server components can be deployed redundantly using the procedures and solutions described in this book. Depending on the type of components, they can be deployed in an active-active configuration or active-passive configuration.

In active-active configuration, multiple instances of a component service client requests at the same time. If one instance fails, the requests being serviced by that instance can be fulfilled by other active instances; the failure and failover of that instance is transparent to clients. An active-active configuration can usually be achieved by clustering instances of components together.

In active-passive configuration, requests are usually serviced by one instance of a component. Upon failure of that component, another instance is made active to respond to the request workload.

- **Death Detection and Auto Restart**

Software processes belonging to Oracle Application Server components, local or distributed, are managed by a central process management system. This system is able to detect the death of processes and restart them even if they are distributed over multiple machines. The system allows customization of parameter values that define process death and restart (such as number of heartbeats). The processes implementing the process management system are themselves redundant as each has a shadow process.

- **Clustering**

Clustering components of a system together allows the components to be viewed functionally as a single entity from the perspective of a client. A cluster increases the scalability, availability, and manageability of the components.

Several types of clusters can exist with Oracle Application Server components. Procedures to create and configure these clusters are comprehensively documented in this book.

- **State Replication and Routing**

For stateful client requests, Oracle Application Server can replicate client state in order to enable stateful failover of requests in the event that processes servicing these requests fail. For J2EE requests, replicating client state for J2EE applications can be done declaratively or programmatically, depending on the mechanism being used. For most other components, state-based routing using cookies is available.

- **Connection Failure Management**

Clients often connect to services on the server and reuse these connections. When a process implementing one of these services on the server is restarted, the connection may need to be re-established.

Oracle Application Server components ensure that if a reused connection fails, the connection is retried before a failure condition is propagated to the rest of the system. This allows clients to be transparent to any failures.

- **Backup and Recovery**

Oracle Application Server provides facilities for backing up system state and using this backup to recover from failures. In certain circumstances, a component or system failure may not be repairable. The Oracle Application Server Backup and Recovery Tool can be used to back up the system at certain intervals and restore a backup when an unrepairable failure occurs.

For specific problems localized to the HTTP listener and J2EE container, a runtime configuration management system allows these components to be check pointed quickly and also allows for undo operations for configuration errors.

- **Disaster Recovery**

Natural and physical disasters can happen to areas where an Oracle Application Server site hosting critical applications is physically located. A solution for recovering from such disasters is documented in this guide. This solution is a site-to-site recovery solution that allows the backing up of the state of an entire Oracle Application Server site and recovering it to another site that is physically distant from the first.

Types of Failures

Table 1–1 depicts the various types of failures that are possible with the Oracle Application Server system and the strategies that are used to prevent or solve the failures. For the purpose of discussion, maintenance activities during planned downtime is also included.

Table 1–1 System downtime, failures, and availability solutions

Downtime Type	Failure Type	Solution
Unplanned Downtime	System Failure	Load balancers, Farm, Oracle Process Management and Notification, Oracle Application Server Active Failover Cluster, Oracle Application Server Cold Failover Clusters
	Data Failure and Disaster	Remote Site, Backup and Recovery, Oracle Data Guard
	Human Error	Backup and Recovery, Oracle Data Guard
Planned Downtime	System Maintenance	Distributed and Dynamic Configuration
	Data Maintenance	No downtime required as data is stored in Oracle database. Backup and Recovery tool for configuration files in filesystem.

As depicted, solutions exist to prevent or recover from unplanned system failures to unintentional human errors. These solutions enable Oracle Application Server to be robust and reliable, and offer high availability to the applications that it hosts.

Organization of this Guide

This guide has been organized into several chapters using the layers of the middle tier and Oracle Application Server Infrastructure as a baseline. When the term "middle tier" is mentioned in this book, the reference is made generically to the Oracle Application Server middle tier installation types. However, where Oracle Application Server Clusters are discussed, only the J2EE and Web Cache installation type is inferred as this is the only middle tier installation type that can be part of an Oracle Application Server Cluster.

Chapters 2 and 3 contain the description and configuration of the middle tier for high availability, respectively. Chapters 3 and 5 have the similar organization of information but for the Infrastructure. Chapter 6 contains the setup and operational information for the site-to-site Oracle Application Server Disaster Recovery solution.

High Availability Information in Other Documentation

The following table provides a list of cross-references to high availability information in other documents in the Oracle library. This information mostly pertains to high availability of various Oracle Application Server components.

Table 1–2 Cross-references to high availability information in Oracle documentation

Component	Location of Information
Overall high availability concepts	In the high availability chapter of <i>Oracle Application Server 10g Concepts</i> .
Oracle installer	In the chapter for installing in a high availability environment in <i>Oracle Application Server 10g Installation Guide</i> .
Oracle Application Server Backup and Recovery Tool	In the backup and restore part of <i>Oracle Application Server 10g Administrator's Guide</i> .
Oracle Application Server Web Cache	<i>Oracle Application Server Web Cache Administrator's Guide</i>
Identity Management service replication	In "Advanced Configurations" chapter of <i>Oracle Application Server Single Sign-On Administrator's Guide</i> .
Identity Management high availability deployment	In "Directory Replication and High Availability" chapter of <i>Oracle Internet Directory Administrator's Guide</i> . In "Oracle Identity Management Deployment Planning" chapter of <i>Oracle Identity Management Concepts and Deployment Planning Guide</i> .
Database high availability	Oracle High Availability Architecture and Best Practices
Distributed Configuration Management commands	<i>Distributed Configuration Management Reference Guide</i>
Oracle Process Management and Notification commands	<i>Oracle Process Manager and Notification Server Administrator's Guide</i>
OC4J high availability	<i>Oracle Application Server Containers for J2EE Services Guide</i> <i>Oracle Application Server Containers for J2EE User's Guide</i> <i>Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide</i>
Java Object Cache	<i>Oracle Application Server Web Services Developer's Guide</i>
Load balancing to OC4J processes	<i>Oracle HTTP Server Administrator's Guide</i>
Oracle Application Server Wireless high availability	<i>Oracle Application Server Wireless Administrator's Guide</i>

Table 1–2 (Cont.) Cross-references to high availability information in Oracle documentation

Component	Location of Information
Oracle Application Server Reports Services high availability	<i>Oracle Application Server Reports Services Publishing Reports to the Web</i>
Oracle Application Server Discoverer high availability	<i>Oracle Application Server Discoverer Configuration Guide</i>
Oracle Application Server Forms Services high availability	<i>Oracle Application Server Forms Services Deployment Guide</i>
Oracle Application Server InterConnect ini file information	<i>Oracle Application Server InterConnect User's Guide</i>

In addition, references to these and other documentation are noted in the text of this guide, where applicable.

Middle Tier High Availability

This chapter describes solutions that are available to protect the Oracle Application Server middle tier from failures. It is organized into the following main sections:

- OracleAS Middle Tier Overview
- Features and Components for Middle Tier High Availability
- HTTP Service High Availability
- J2EE High Availability
- Oracle Application Server Portal High Availability
- Oracle Application Server Wireless High Availability
- Business Intelligence High Availability
- Oracle Application Server Forms Services High Availability
- Oracle Application Server Integration High Availability
- Middle Tier Recovery Solutions

OracleAS Middle Tier Overview

OracleAS middle tier consists of components that provide front-end application services to clients. The middle tier can be created with one of three installation types:

- *J2EE and Web Cache*

This installation type installs components for Oracle Application Server Containers for J2EE (OC4J) and Oracle Application Server Web Cache (OracleAS Web Cache). These components provide J2EE 1.3 runtime containers, and static and dynamic content caching functionality. Of the three installation types, J2EE and Web Cache installations are the only ones that can be clustered together as OracleAS Clusters.
- *Portal and Wireless*

This installation type installs Oracle Application Server Portal and Oracle Application Server Wireless components in addition to the components in the J2EE and Web Cache installation type.
- *Business Intelligence and Forms*

This installation installs components for Oracle Application Server Forms Services, Oracle Application Server Reports Services, Oracle Application Server Discoverer, Oracle Application Server Personalization, and components in the Portal and Wireless installation type.

Note: Oracle Application Server ProcessConnect and Oracle Application Server InterConnect have their own installers that are run from their own CD-ROMs.

See Also: *Oracle Application Server 10g Installation Guide* for your platform for complete information on Oracle Application Server installation types.

For running J2EE applications, the J2EE and Web Cache installation type provides the core functionality to do so. This includes HTTP services through Oracle HTTP Server. J2EE applications can be enabled with single sign-on functionality if the middle tier is configured to use the Oracle Identity Management framework in the OracleAS Infrastructure.

Note: The Portal and Wireless, and Business Intelligence and Forms installation types always require the Oracle Application Server Metadata Repository and Oracle Identity Management services, because components in these middle tier types need to access their schemas in the Oracle Application Server Metadata Repository.

OracleAS Middle Tier Terminology

The following terms are useful to review before discussing Oracle Application Server middle tier high availability:

- **Oracle Application Server Instance:** An Oracle Application Server instance (also called an application server instance and OracleAS instance) is the set of processes required to run the configured components within an application server installation. There can be only one application server instance per application server installation. The terms installation and instance are sometimes used interchangeably; however, note that an installation is the set of files installed into an Oracle home, while an instance is a set of processes associated with those files.
- **Component Instance:** Component instances include a single Oracle HTTP Server process or multiple Oracle Application Server Containers for J2EE (OC4J) instances (see Figure 2-2).
- **Oracle Application Server Cluster:** An Oracle Application Server Cluster (OracleAS Cluster) is a collection of application server instances with identical configurations and application deployments. Oracle Application Server Clusters enforce homogeneity between instances that are part of the cluster so that the cluster appears and functions as a single instance. With appropriate front-end load balancing, any instance in an Oracle Application Server Cluster can serve client requests. This simplifies configuration and deployment across multiple instances and provides high availability for applications deployed to Oracle Application Server Clusters. Oracle Application Server Clusters can be managed using a database or file-based repository. They can also be manually configured without a repository. See "Oracle Application Server Clusters" on page 2-6 for more details.
- **OC4J Island:** An Oracle Application Server Containers for J2EE (OC4J) island is a group of OC4J processes that replicate session state, for stateful Web applications, among each OC4J process that is part of the island. OC4J processes in an island can be on multiple nodes. When an OC4J island shares the same name across OracleAS instances, session state is replicated across the OracleAS instances.

- **Oracle Application Server Farm:** A Farm is a collection of application server instances that share the same Oracle Application Server Infrastructure or that use the same application server instance for their file-based repository host. An Oracle Application Server Farm can include application server instances that are in an OracleAS Cluster, as well as those that are not (Figure 2-2).

See Also: *Oracle Application Server 10g Concepts*

Services Available

The three installation types mentioned above provide the following services to application clients:

J2EE

J2EE services are provided by OC4J. OC4J is a J2EE-compliant container providing a JSP, Servlet, and EJB runtime environment. OC4J can be clustered together to provide failover and redundancy to clients. See "Oracle Application Server Clusters" on page 2-6 and Chapter 4, "Managing and Operating Middle Tier High Availability".

HTTP

Oracle HTTP Server provides HTTP support for Oracle Application Server. It is based on the open source Apache HTTP Server (version 1.3.27) with several standard Apache and OracleAS-specific modules. Oracle HTTP Server is a component of OracleAS Instances in the middle tier. HTTP support for the Infrastructure tier is also provided by Oracle HTTP Server (refer to the section "Oracle HTTP Server" on page 3-4).

Additionally, Oracle Application Server Web Cache (OracleAS Web Cache) provides a cache for requested HTTP objects. See the section called "Caching" on page 2-4.

See Also: *Oracle HTTP Server Administrator's Guide*

Portal

OracleAS provides an out-of-the-box enterprise portal that does not require extensive programming and maintenance. You can use Oracle Application Server Portal (OracleAS Portal) and its associated components to build, deploy, and maintain self-service and integrated enterprise portals. OracleAS Portal allows for self-service content management and publishing, wizard-based development, and Web services deployment and publishing in an extensible framework. Oracle Application Server Portal Developer Kit, Oracle Ultra Search, and Oracle Application Server Syndication Services support OracleAS Portal to provide these functionalities. Refer to "Oracle Application Server Portal High Availability" on page 2-22 for high availability details.

See Also: *Oracle Application Server Portal Configuration Guide*

Business Intelligence

Oracle Application Server 10g provides business intelligence services through several components:

- Oracle Application Server Reports Services
 - Oracle Application Server Reports Services publish high quality, dynamically generated reports on a scalable, secure platform. These reports can be delivered through Web-browsers or non browser interface.
- Oracle Application Server Discoverer

Oracle Application Server Discoverer enables you to perform dynamic, ad-hoc query reporting and analysis for Web browser delivery.

- **Oracle Application Server Personalization**

Oracle Application Server Personalization dynamically serves personalized content recommendations to both registered and anonymous visitors as they browse your site.

Refer to "Business Intelligence High Availability" on page 2-24 for a discussion of high availability for these components.

Oracle Application Server Forms Services

Oracle Application Server Forms Services (OracleAS Forms Services) is a comprehensive application framework optimized to deploy OracleAS Forms Services applications in a multi-tiered environment. It is a middle tier application framework for deploying complex, transactional forms applications.

You can build new applications with Forms Developer and deploy them to the Internet with OracleAS Forms Services. Developers can also take current applications that were previously deployed in the legacy client-server model and move them to a three-tier architecture without changing the application code.

At runtime, Oracle Application Server Forms Services has two components: a servlet and a separate runtime process. Refer to "Oracle Application Server Forms Services" on page 2-4 for information on how they can be made highly available.

Single Sign-On

Single sign-on service in Oracle Application Server 10g is provided by Oracle Application Server Single Sign-On (OracleAS Single Sign-On), which is part of the Oracle Identity Management framework. This framework allows all applications deployed on Oracle Application Server and its components, such as OracleAS Portal and OracleAS Reports Services, to have a centralized authentication and authorization system for users. Users need only log in once to access any of the applications and resources they are authorized for. The credentials for users are stored in an LDAP version 3-compliant repository (Oracle Internet Directory).

On the middle tier, the Apache module, `mod_ossso`, allows single sign-on requests to be forwarded to the single sign-on server in the OracleAS Infrastructure where the other components of the framework reside. These components are Oracle Internet Directory, and Oracle Application Server Certificate Authority. They are further discussed in Chapter 3, "Infrastructure High Availability".

See Also: *Oracle Application Server 10g Concepts* for a discussion on Oracle Identity Management.

Caching

Two main caching mechanisms are available in OracleAS:

- *OracleAS Web Cache*

OracleAS Web Cache is a HTTP-level cache deployed in front of Oracle HTTP Server. It caches both static (HTML, GIF, and JPEG) and dynamic (generated by servlets and JSPs) content. OracleAS Web Cache can be configured to perform as a load balancer for Oracle HTTP Server instances. Additionally, they can be clustered together to provide failover, redundancy, and improved scalability for cached content. Refer to "Oracle Application Server Web Cache Clusters" on page 2-11 for details.

- *OC4J Java Object Cache*

Java Object Cache is an in-process caching service for Java application use. It stores frequently accessed or resource-expensive (to create) objects in memory or on disk. Objects can be distributed across OC4J processes that have the same applications deployed in them (for example, OC4J processes in the same OracleAS Cluster). The distributed objects are coordinated and synchronized. Hence, failure of one process does not reduce the availability of cached objects. Java Object Cache enables Oracle Application Server 10g to retrieve content faster and reduce load on Java applications, thereby increasing application availability. See "Oracle Application Server Clusters" on page 2-6 for more information.

Features and Components for Middle Tier High Availability

The middle tier architecture involves several features and constructs that allow for high availability. These are:

- Oracle Application Server Instance High Availability
- Oracle Application Server Clusters
- Oracle Application Server Web Cache Clusters
- OC4J Islands
- Other High Availability Components
- High Availability Through Distributed Configuration
- Process Monitoring and Restart

Oracle Application Server Instance High Availability

The Oracle Application Server architecture supports high availability in the middle tier that in many cases can prevent unplanned down time for deployed applications. This section provides an overview of the architecture of an Oracle Application Server instance and shows some of the mid-tier high availability features.

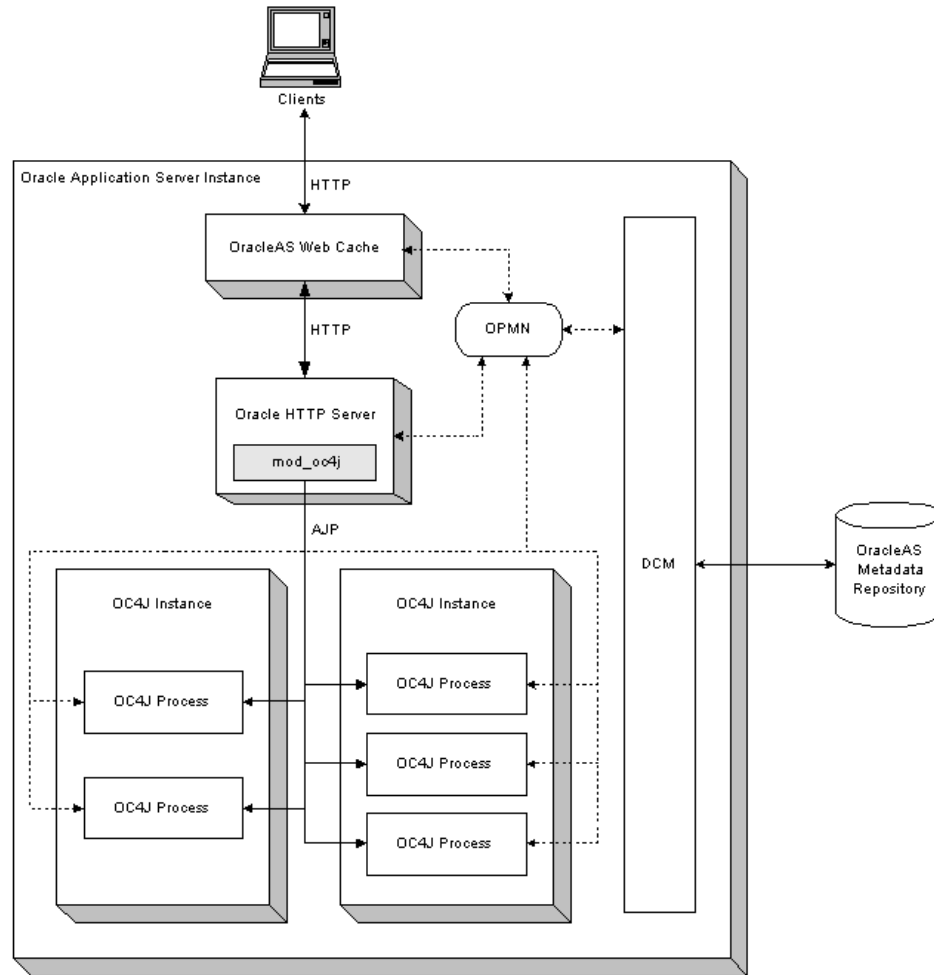
Within each Oracle Application Server instance, the following features provide high availability within the instance, and for any clusters that the instance is a part of:

- Process Monitoring – Using the Oracle Process Management and Notification system provides for process death detection and process restarting in the event that problems are detected for monitored processes.
- Configuration Cloning – Using the Distributed Configuration Management features that uses a Oracle Application Server Metadata Repository for configuration information provides distributed and managed configuration for Oracle Application Server instances and for Oracle Application Server instances that are part of a cluster.
- Data Replication – Using OC4J instances with OC4J islands that provide Web application level stateful session replication, and using EJB sessions, data is replicated across processes within an Oracle Application Server instance and across different Oracle Application Server instances that may reside on different hosts when using Oracle Application Server Clusters. This allows stateful session based applications to remain available even when processes within an Oracle Application Server instance become unavailable or fail.
- Smart Routing – Oracle Application Server Web Cache and Oracle HTTP Server (`mod_oc4j`) provide configurable and intelligent routing for incoming requests.

Requests are routed only to processes and components that `mod_oc4j` determines to be alive, through communication with the Oracle Process Management and Notification system.

Figure 2–1 shows the architecture of an Oracle Application Server instance, including the features listed above that provide redundant processes and automatic recovery within an instance.

Figure 2–1 Oracle Application Server Instance Architecture



Oracle Application Server Clusters

An Oracle Application Server Cluster (OracleAS Cluster) is a set of application server instances configured to act in concert to deliver greater scalability and availability than a single instance. Using OracleAS Clusters removes the single point of failure that a single host poses. While a single application server instance leverages the operating resources of a single host, a cluster can span multiple hosts, distributing application execution over a greater number of CPUs. A single application server instance is vulnerable to the failure of its host and operating system, but a cluster continues to function despite the loss of an operating system or a host, hiding any such failure from clients.

This section covers the following:

- Types of Oracle Application Server Clusters

- Cluster-Wide Configuration for Oracle Application Server Clusters that are Managed Using a Repository
- Requirements for Oracle Application Server Instances to Join Oracle Application Server Clusters that are Managed Using a Repository
- Properties of Oracle Application Server Instances in Oracle Application Server Clusters that are Managed Using a Repository

Types of Oracle Application Server Clusters

There are two types of Oracle Application Server Clusters, Oracle Application Server Clusters managed using a file-based or database repository and Oracle Application Server Clusters that are manually configured:

Note: Only OracleAS instances of the J2EE and Web Cache installation type can be clustered as an OracleAS Cluster.

- **Oracle Application Server Clusters managed using a file-based or database repository** contain a collection of application server instances with identical configurations and application deployments. Oracle Application Server Clusters enforce homogeneity between instances that are part of the cluster so that the cluster appears and functions as a single instance. Oracle Application Server Clusters that are managed using a repository propagate configuration information across all application server instances in the cluster, which simplifies configuration and deployment.

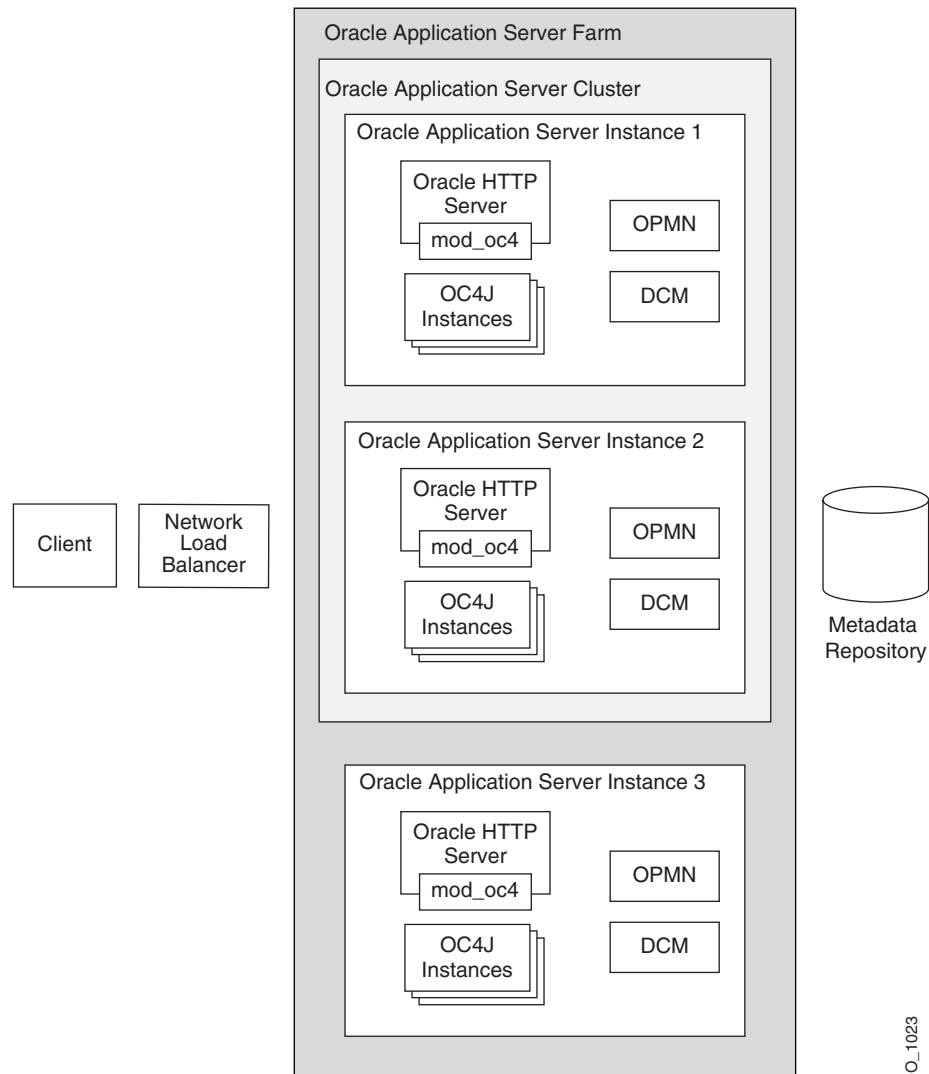
There are two types of Oracle Application Server Clusters managed using a repository: Oracle Application Server Clusters managed using a database repository and Oracle Application Server Clusters managed using a file-based repository:

- **Oracle Application Server Clusters managed using a database repository**
These clusters use a database to store metadata and configuration information. This type of Oracle Application Server Cluster requires the Oracle Application Server Infrastructure since metadata and configuration information is stored in a Oracle Application Server Metadata Repository that resides on an Infrastructure host.
- **Oracle Application Server Clusters managed using a file-based repository**
These clusters designate an application server instance as the **repository host**. The repository host uses its file system to store the Oracle Application Server Metadata Repository that retains the metadata and configuration information for the cluster.

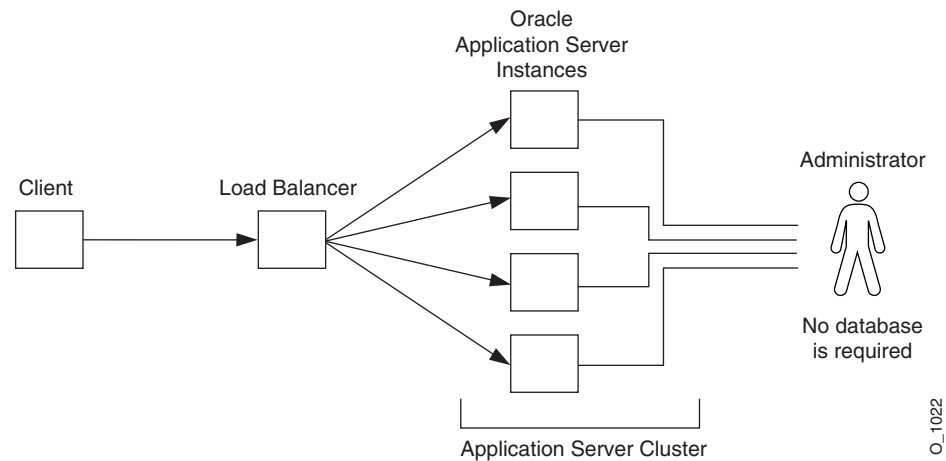
Figure 2–2 shows an example of an Oracle Application Server Cluster managed using a database repository. Figure 2–2 shows three application server instances. All three application server instances share the same Oracle Application Server Metadata Repository. Thus, all three application server instances in the cluster are part of the same Farm.

Application server instances 1 and 2 are part of an Oracle Application Server Cluster managed using a database repository. In front of the cluster is a front-end load balancer, this may be Oracle Application Server Web Cache or a hardware load balancer appliance. Included within each application server instance are its manageability features—Oracle Process Management and Notification (OPMN) and Distributed Configuration Management (DCM)—and its installed components—Oracle HTTP Server and Oracle Application Server Containers for J2EE (OC4J).

Figure 2–2 OracleAS Cluster Architecture



- **Manually Configured Oracle Application Server Clusters** rely on the administrator to manually configure each instance within the cluster (Figure 2–3). With manually configured Oracle Application Server Clusters, it is the administrator’s job to make a group of application server instances function as a cluster. Maintaining the configuration and application deployment information on these Oracle Application Server Clusters can be a difficult task. Manually configured Oracle Application Server Clusters provide scalability and availability, but not manageability. The administrator has the responsibility to synchronize the configuration of the application server instances across the cluster.

Figure 2-3 Manually Configured OracleAS Clusters**See Also:**

- "High Availability Through Distributed Configuration" on page 2-16
- "Process Monitoring and Restart" on page 2-15
- "Using a File-Based Repository with OracleAS Clusters" on page 4-9

Cluster-Wide Configuration for Oracle Application Server Clusters that are Managed Using a Repository

Oracle Application Server Clusters that are managed using a repository contain a collection of application server instances with identical configuration information. Oracle Application Server propagates configuration information across all application server instances that are in an Oracle Application Server Cluster. Each application server instance in a cluster uses the same **base configuration**. The base configuration is defined by **cluster-wide configuration** information. When an application server instance joins an Oracle Application Server Cluster, the Distributed Configuration Management system assures that the base configuration is applied to the new instance so that the new instance uses the same cluster-wide configuration.

Using either Application Server Console or `dcmtcl` to deploy an application on an instance, or to modify an application server instance, cluster-wide configuration modifications are propagated to all other application server instances across Oracle Application Server Clusters.

Cluster-wide configuration excludes certain **instance-specific parameters**. The instance-specific parameters are not propagated to all of the application server instances across a cluster. If you modify an instance-specific parameter, it is not propagated as it is only applicable to the specific application server instance where the change is made.

See Also: "Cluster-Wide Configuration Changes and Modifying OC4J Instances" on page 4-19

Requirements for Oracle Application Server Instances to Join Oracle Application Server Clusters that are Managed Using a Repository

In order for an application server instance to join Oracle Application Server Clusters, the application server instance must be **clusterable**. For an application server instance to be clusterable, the following must be true:

1. The application server instance must be part of the Farm where the Oracle Application Server Cluster resides. You can associate application server instances with a OracleAS Metadata Repository either during installation time or after installation using Application Server Console.
2. Each application server instance in a cluster must be installed on the same type of operating system, such as UNIX.
3. Each application server instance can contain only one Oracle HTTP Server.
4. Each application server instance can contain one or more OC4J instances.

See Also: "Adding an Application Server Instance to an OracleAS Cluster" on page 4-7

Properties of Oracle Application Server Instances in Oracle Application Server Clusters that are Managed Using a Repository

Once application server instances join a cluster, they have the following properties:

- Each application server instance uses the same cluster-wide configuration. That is, if you modify any cluster-wide parameters, the modifications are propagated to all application server instances in the cluster.
- If you deploy an application to one application server instance, it is propagated to all application server instances in the cluster. The application is actually deployed to an OC4J instance in the application server instance and propagated to the same OC4J instance in the other application server instances in the cluster. You can change some of the configuration for the deployed application, and this change is propagated to the same OC4J instance in the other application server instances across the cluster.
- Most clustering management, configuration, and application deployment is handled through Oracle Enterprise Manager. If you want to use a command-line tool, you can use the Distributed Configuration Management command-line tool `dcmctl`.
- The base configuration is created from the first application server instance to join a cluster.
- You can remove application server instances from the cluster. The application server instance is stopped when removed from the cluster. When the last application server instance is removed, the cluster still remains. You must delete the cluster itself for it to be removed.

See Also:

- "Managing Application Server Instances in an OracleAS Cluster" on page 4-7
- *Distributed Configuration Management Reference Guide* for information on `dcmctl` commands

Oracle Application Server Web Cache Clusters

Two or more OracleAS Web Cache instances can be clustered together to create a single logical cache. Physically, the cache can be distributed amongst several nodes. If one node fails, a remaining node in the same cluster can fulfill the requests serviced by the failed node. The failure is detected by the remaining nodes in the cluster who take over ownership of the cacheable content of the failed member. The load balancing mechanism in front of the OracleAS Web Cache cluster, for example, a hardware load balancing appliance, redirects the requests to the live OracleAS Web Cache nodes.

OracleAS Web Cache clusters also add to the availability of OracleAS instances. By caching static and dynamic content in front of the OracleAS instances, requests can be serviced by OracleAS Web Cache reducing the need for the requests to be fulfilled by OracleAS instances, particularly for Oracle HTTP Servers. The load and stress on OracleAS instances is reduced, thereby increasing availability of the components in the instances.

Oracle Application Server Web Cache can also perform a stateless or stateful load balancing role for Oracle HTTP Servers. Load balancing is done based on the percentage of the available capacity of each Oracle HTTP Server, or, in other words, the weighted available capacity of each Oracle HTTP Server. If the weighted available capacity is equal for several Oracle HTTP Servers, OracleAS Web Cache uses round robin to distribute the load. Refer to *Oracle Application Server Web Cache Administrator's Guide* for the formula to calculate weighted available capacity.

In the case of failure of a Oracle HTTP Server, OracleAS Web Cache redistributes the load to the remaining Oracle HTTP Servers and polls the failed server intermittently until it comes back online. Thereafter, OracleAS Web Cache recalculates the load distribution with the revived Oracle HTTP Server in scope.

See Also:

- "HTTP Service High Availability" on page 2-18
- *Oracle Application Server Web Cache Administrator's Guide*

OC4J Islands

Oracle Application Server provides several strategies for ensuring high availability with OC4J instances, both within an application server instance and across a cluster that includes multiple application server instances.

This section covers the following:

- Web Application Session State Replication with OC4J Islands
- Stateful Session EJB High Availability Using EJB Clustering
- JNDI Namespace Replication
- OC4J Distributed Caching Using Java Object Cache

Besides the high availability features described in this section, other Oracle Application Server features enable OC4J processes to be highly available, including the load balancing feature in Oracle HTTP Server and the Oracle Process Management and Notification system that automatically monitors and restarts processes.

See Also:

- "HTTP Service High Availability" on page 2-18
- "Process Monitoring and Restart" on page 2-15

Web Application Session State Replication with OC4J Islands

When a stateful Web application is deployed to OC4J, multiple HTTP requests from the same client may need to access the application. However, if the application running on the OC4J server experiences a problem where the OC4J process fails, the state associated with a client request may be lost. Using Oracle Application Server, there are three ways to guard against such failures:

- State safe applications save their state in a database or other persistent storage system, avoiding the loss of state when the server goes down. Obviously, there is a performance cost for continually writing the application state to persistent storage.
- Stateless applications do not have a state that needs to be carried between requests, and so, stateless applications do not have state integrity considerations when a server goes down. Another active server can handle the request. High availability for stateless applications is easier to achieve than for state safe or stateful applications.
- Stateful applications can use OC4J session state replication, with OC4J islands, to automatically replicate the session state across multiple processes in an application server instance, and in a cluster, across multiple application instances which may run on different nodes.

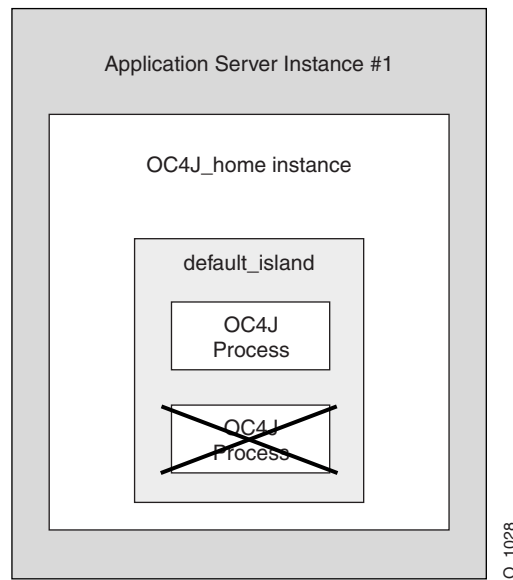
OC4J processes can be grouped into islands to support session state replication for high availability of Web applications. Using OC4J islands together with Oracle HTTP Server `mod_oc4j` request routing provides stateful failover in the event of a software or hardware problem. For example, if an OC4J process that is part of an island fails, `mod_oc4j` is notified of the failure by OPMN and routes requests to another OC4J process in the same island.

Web Application Session State Protecting Against Software Problems To guard against software problems, such as OC4J process failure or hang, you can configure an OC4J instance to run multiple OC4J processes in the same OC4J island. The processes in the OC4J island communicate their session state between each other. Using this configuration provides failover and high availability by replicating state across multiple OC4J processes running on an application server instance.

In the event of a failure, Oracle HTTP Server forwards requests to active (alive) OC4J process within the OC4J island. In this case, the Web application state for the client is preserved and the client does not notice any loss of service.

Figure 2–4 shows this type of software failure within an application server instance.

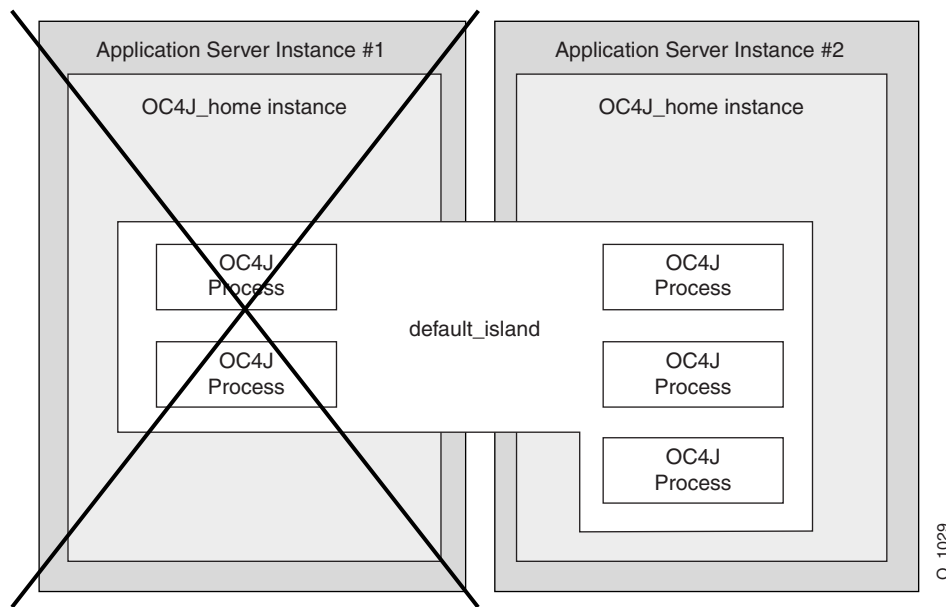
Figure 2–4 Web Application Session State Failover Within An OC4J Island in an OC4J Instance



Web Application Session State Replication Protecting Against Hardware Problems To guard against hardware problems, such as the failure of the node where an application server instance runs, you can configure OC4J islands across application server instances that are in more than one node in an OracleAS Cluster. By configuring an OC4J island that uses the same name across multiple application server instances, the OC4J processes can share session state information across the OracleAS Cluster. When an application server instance fails or is not available, for example, when the node it runs on goes down, Oracle HTTP Server forwards requests to an OC4J process in an application server instance that is available. Thus, Oracle HTTP Server forwards requests only to active (alive) OC4J processes within the cluster.

In this case, the Web application state for the client is preserved and the client does not notice any irregularity.

Figure 2–5 depicts an OC4J island configured within an OracleAS Cluster.

Figure 2–5 Web Application Session State Failover Within An OracleAS Cluster

Configuring OC4J Islands With High Availability To protect against software or hardware failure while maintaining state with the least number of OC4J processes, you need to configure at least two OC4J processes in the same island on multiple application server instances running on separate nodes. For example, if you have two application server instances, instance 1 and instance 2, you can configure two OC4J processes in the `default_island` on each application server instance. With this configuration, stateful session applications are protected against hardware and software failures, and the client maintains state if either of the following types of failures occurs:

- If one of the OC4J processes fails, then the client request is redirected to the other OC4J process in the `default_island` on the same application server instance. State is preserved and the client does not notice any irregularity.
- If application server instance 1 terminates abnormally, then the client is redirected to the OC4J process in the `default_island` on application server instance 2. The state is preserved and the client does not notice any irregularity.

See Also: "Configuring OC4J Islands and OC4J Processes" on page 4-23

Stateful Session EJB High Availability Using EJB Clustering

Using OC4J, stateful session EJBs can be configured to provide state replication across OC4J processes running within an application server instance or across an OracleAS Cluster. This EJB replication configuration provides high availability for stateful session EJBs by using multiple OC4J processes to run instances of the same stateful session EJB.

Note: Use of EJB replication (EJB clusters) for high availability is independent of OracleAS Clusters and can involve multiple application server instances installed across nodes that are or are not part of OracleAS Clusters.

EJB clusters provide high availability for stateful session EJBs. They allow for failover of these EJBs across multiple OC4J processes that communicate over the same multicast address. Thus, when stateful session EJBs use replication, this can protect against process and node failures and can provide for high availability of stateful session EJBs running on Oracle Application Server.

See Also:

- "Configuring EJB Application State Replication for OracleAS Clusters" on page 4-21
- *Oracle Application Server Containers for J2EE User's Guide*
- *Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide*

JNDI Namespace Replication

When EJB clustering is enabled, JNDI namespace replication is also enabled between the OC4J instances in an OracleAS Cluster. New bindings to the JNDI namespace in one OC4J instance are propagated to other OC4J instances in the OracleAS Cluster. Re-bindings and unbindings are not replicated.

The replication is done outside the scope of OC4J islands. In other words, multiple islands in an OC4J instance have visibility into the same replicated JNDI namespace.

See Also: *Oracle Application Server Containers for J2EE Services Guide*

OC4J Distributed Caching Using Java Object Cache

Oracle Application Server Java Object Cache provides a distributed cache that can serve as a high availability solution for applications deployed to OC4J. The Java Object Cache is an in-process cache of Java objects that can be used on any Java platform by any Java application. It allows applications to share objects across requests and across users, and coordinates the life cycle of the objects across processes.

Java Object Cache enables data replication among OC4J processes even if they do not belong to the same OC4J island, application server instance, or Oracle Application Server Cluster.

By using Java Object Cache, performance can be improved since shared Java objects are cached locally, regardless of which application produces the objects. This also improves availability; in the event that the source for an object becomes unavailable, the locally cached version will still be available.

See Also: The Java Object Cache chapter in the *Oracle Application Server Web Services Developer's Guide* for complete information on using Java Object Cache

Process Monitoring and Restart

In an application server instance and across OracleAS Clusters, Oracle Process Management and Notification (OPMN) monitors Oracle Application Server components, including Oracle HTTP Server, OC4J, OracleAS Web Cache, and Oracle Application Server Reports Services (OracleAS Reports Services).

The OPMN system, which is itself an Oracle Application Server component, assists in making Oracle Application Server highly available by monitoring and automatically restarting Oracle Application Server processes that fail. When a process becomes unavailable, OPMN notifies certain other Oracle Application Server components that the process is unavailable. For example, in an OracleAS Cluster, when an OC4J process

fails, the Oracle HTTP Server `mod_oc4j` modules are notified of the failure and do not send requests to the failed OC4J process until OPMN uses an event to notify the modules that the OC4J process has been restarted.

OPMN consists of the following sub-components:

- Oracle Process Manager
- Oracle Notification System

Oracle Process Manager

The Oracle Process Manager is responsible for starting, restarting, shutting down, and detecting the death of Oracle Application Server processes. The Oracle Process Manager can start or stop processes through one of the following ways:

- directives in the `opmn.xml` configuration file
- by selecting Application Server Console operations, such as start or stop for components such as OC4J instances
- by using the `opmnctl` command line utility.

Oracle Notification System

The Oracle Notification System is the communication mechanism for failure, recovery, startup, and other related notifications between components. The notification system operates according to a subscriber-publisher model, wherein any component that wishes to receive an event of a certain type subscribes to the Oracle Notification System. When such an event is published, the Oracle Notification System sends it to all relevant subscribers.

In an Oracle Application Server Cluster, the Oracle HTTP Servers communicate using the Oracle Notification System and are aware of the active OC4J processes across the Oracle Application Server Cluster. This communication mechanism enables each Oracle HTTP Server to know the live OC4J processes in the Oracle Application Server Cluster so that incoming requests can be load balanced to them.

See Also: *Oracle Process Manager and Notification Server Administrator's Guide*

High Availability Through Distributed Configuration

Oracle Application Server uses the Distributed Configuration Management (DCM) system to manage the cluster-wide configuration in Oracle Application Server clusters. DCM provides supports the following actions:

- When new application server instances join a cluster, DCM automatically replicates the base configuration to all instances in the cluster.
- DCM propagates application deployments and configuration changes to all application server instances across a cluster.

For Oracle Application Server high availability, when a system in an Oracle Application Server cluster is down, there is no single point of failure for DCM. DCM remains available on all the available nodes in the cluster.

Using DCM helps reduce deployment and configuration errors in a cluster; these errors could, without using DCM, be a significant cause of system downtime.

Enterprise Manager uses DCM commands to perform application server configuration and deployment. You can also issue DCM commands manually using the `dcmtcl` command.

DCM controls the following configuration commands

- Create or remove a cluster
- Add or remove application server instances to or from a cluster
- Synchronize configuration changes across application server instances

Note the following when making configuration changes to a cluster or deploying applications to a cluster:

- If Enterprise Manager is up and managing the cluster, you can invoke the DCM command-line tool from any host where a clustered application server instance is running. DCM informs Enterprise Manager of the requested function and Enterprise Manager then interfaces with the other DCM management features on the other application server instances in the cluster to complete the cluster-wide configuration or application deployment.
- If Enterprise Manager is not up and managing the cluster, if you want configuration changes to be applied dynamically across the cluster, the DCM daemon must be running on each cluster. To start the DCM daemon, run the DCM command-line tool, `dcmtcl`, on each application server instance in the cluster.

See Also: *Distributed Configuration Management Reference Guide*

Other High Availability Components

Several external components can be used to improve the availability of Oracle Application Server. These components are discussed in the following sections:

- Improving Availability with an External Load Balancer
- Improving Availability with Operating System Clusters

Improving Availability with an External Load Balancer

You can use an external load balancer to improve the availability of both clustered and non-clustered Oracle Application Server instances.

Clients access the cluster through a load balancer, which hides the cluster configuration. The load balancer can send requests to any application server instance in the cluster, as any instance can service any request. An administrator can raise the capacity of the system by introducing additional application server instances to the cluster. These instances can be installed on multiple nodes to allow for redundancy in case of node failure.

You can also use a load balancer to increase the availability of non-clusterable Oracle Application Server instances, such as Portal and Wireless, when they are installed on multiple nodes. As long as the load balancer is configured to serve a set of nodes, it will route requests accordingly.

Types of External Load Balancers There are three types of load balancers you can use with Oracle Application Server instances: hardware load balancers and network load balancers. Table 2–4 summarizes these:

Table 2–1 Types of External Load Balancers Summary

Load Balancer Type	Description
Hardware Load Balancer	Hardware load balancing involves placing a hardware load balancer, such as Big-IP or Alteon, in front of a group of Oracle Application Server instances or OracleAS Web Cache. The hardware load balancer routes requests to the Oracle HTTP Server or OracleAS Web Cache instances in a client-transparent fashion.
Windows Network Load Balancer (applicable to Windows version of Oracle Application Server)	With some Windows operating systems, you can use the operating system to perform network load balancing. For example, with Microsoft Advanced Server, the NLB functionality allows you to send requests to different machines that share the same virtual IP or MAC address. The servers themselves do not need to be clustered at the operating system level.

Note: Check <http://metalink.oracle.com> for information on supported external load balancers.

High Availability Benefits of External Load Balancing There are three main benefits of using clusters: scalability, availability, and manageability. Load balancing improves scalability by providing an access point through which requests are routed to one of many available instances. Instances can be added to the group that the load balancer serves to accommodate additional users.

Load balancing improves availability by routing requests to the most available instances. If one instance goes down, or is particularly busy, the load balancer can send requests to another active instance.

Load balancing improves the system manageability by routing application deployment and system configuration requests to the most available instances. If one instance goes down, or is particularly busy, the load balancer can send requests to another active instance.

Improving Availability with Operating System Clusters

Using operating system clusters involves installing Oracle Application Server on a hardware cluster created through the operating system or other clustering system solutions, such as HP MC Service Guard. Operating system clustering is supported in Oracle Application Server 10g (9.0.4).

HTTP Service High Availability

Oracle HTTP Server and Oracle Application Server Web Cache provide HTTP and HTTPS request handling for Oracle Application Server requests. Each HTTP request is met by a response from Oracle HTTP Server or from Web Cache if the content requested is cached.

This section covers the following topics:

- Web Cache and Oracle HTTP Server High Availability Summary
- OC4J Load Balancing Using mod_oc4j

Web Cache and Oracle HTTP Server High Availability Summary

Table 2–2 summarizes some of the Oracle Application Server high availability features for the Oracle HTTP Server and OracleAS Web Cache components.

Table 2–2 Oracle HTTP Server and OracleAS Web Cache high availability characteristics

Component	Protection from Node Failure	Protection from Service Failure	Protection from Process Failure	Automatic Re-routing	State Replication	Configuration Cloning
OracleAS Web Cache	OracleAS Web Cache cluster protects from single point of failure. An external load balancer should be deployed in front of this cluster to route requests to live OracleAS Web Cache nodes.	In an OracleAS Web Cache cluster, pings are made to a specific URL in each cluster member to ensure that the URL is still serviceable.	OPMN monitors OracleAS Web Cache processes and restarts them upon process failure	OracleAS Web Cache members in a cluster ping each other to verify that peer members are alive or have failed.	OracleAS Web Cache clustering manages replicated objects	OracleAS Web Cache cluster maintains uniform configuration across cluster
Oracle HTTP Server	OracleAS Cluster protects from single point of failure. A load balancer should be deployed in front of Oracle HTTP Server instances. This can be an external load balancer or OracleAS Web Cache.	Load balancer in front of Oracle HTTP Server sends request to another Oracle HTTP Server if first one doesn't respond or is deemed failed through URL pings. Load balancer can be either OracleAS Web Cache or hardware appliance.	OPMN monitors Oracle HTTP Server processes and restarts them upon process failure. Each Oracle HTTP Server is also notified by OPMN when another Oracle HTTP Server process in the OracleAS Cluster fails.	Load balancer in front of Oracle HTTP Server auto re-routes to another Oracle HTTP Server if first does not respond.	None.	OracleAS Cluster allows configuration to be replicated across to other Oracle HTTP Servers in the cluster through DCM.

See Also:

- "Oracle Application Server Web Cache Clusters" on page 2-11
- "Process Monitoring and Restart" on page 2-15
- "Types of External Load Balancers" on page 2-17

OC4J Load Balancing Using mod_oc4j

The Oracle HTTP Server module, `mod_oc4j` provides intelligent routing for HTTP requests that are handled by OC4J. The intelligent routing that `mod_oc4j` provides is an import Oracle Application Server high availability feature. If an OC4J process fails Oracle Process Management and Notification detects the failure and `mod_oc4j` does not send requests to the failed OC4J process until the OC4J process is restarted.

Generally, `mod_oc4j` deals with stateless HTTP requests, since stateful HTTP requests are forwarded to the OC4J process that served the previous request (unless `mod_oc4j` determines, through communication with Oracle Process Management and Notification that the process is not available, in which case `mod_oc4j` forwards the request to an available OC4J).

Using `mod_oc4j` configuration options you can specify different load balancing routing algorithms, depending on the type and complexity of routing you need.

Table 2–3 summarizes the routing styles that `mod_oc4j` provides. For each routing style, Table 2–3 lists the different algorithms that you can configure to modify the routing behavior. These `mod_oc4j` configuration options determine the OC4J process where `mod_oc4j` sends incoming HTTP requests to be handled.

See Also:

- "mod_oc4j Load Balancing With OracleAS Clusters" on page 4-25
- *Oracle HTTP Server Administrator's Guide* for information on using weighted routing and selecting local affinity with `mod_oc4j` load balancing options.

Table 2–3 mod_oc4j Routing Algorithms Summary

Routing Method	Description
Round Robin	Using the simple round robin configuration, all OC4J processes, remote and local to the application server instance running the Oracle HTTP Server, are placed in an ordered list. Oracle HTTP Server then chooses an OC4J process at random for the first request. For each subsequent request, Oracle HTTP Server forwards requests to another OC4J process in round robin style. The round robin configuration supports local affinity and weighted routing options.
Random	Using the simple random configuration, all OC4J processes, remote and local to the application server instance running the Oracle HTTP Server, are placed in an ordered list. For every request, Oracle HTTP Server chooses an OC4J process at random and forwards the request to that instance. The random configuration supports local affinity and weighted routing options.
Metric-Based	Using the metric-based configuration OC4J processes, remote and local to the application server instance running the Oracle HTTP Server, are placed into an ordered list. OC4J processes then regularly communicate to Oracle HTTP Server how busy they are and Oracle HTTP Server uses this information to send requests to the OC4J processes that are less busy. The metric-based configuration supports a local affinity option.

OC4J Load Balancing Using Local Affinity and Weighted Routing Options

Using `mod_oc4j` options, you can select a routing method for routing OC4J requests. If you select either round robin or random routing, you can also use local affinity or weighted routing options. If you select metric-based routing, you can also use the local affinity option.

Using the weighted routing option, a weight is associated with OC4J processes on a node, as configured in `mod_oc4j`, on a node by node basis. During request routing, `mod_oc4j` then uses the routing weight to calculate which OC4J process to assign requests to. Thus, OC4J processes running on different nodes can be assigned different weights.

Using the local affinity option, `mod_oc4j` keeps two lists of available OC4J processes to handle requests, a local list and a remote list. If processes are available from the local list then requests are assigned locally using the random routing method or, for metric-based routing using metric-based routing. If no processes are available in the local list, then `mod_oc4j` selects processes randomly from the remote list when random method, using a round robin method for the round robin method, or using metric-based routing with the metric-based method.

Choosing a mod_oc4j Routing Algorithm

Table 2-3 summarizes the available routing options. To select a routing algorithm to configure with mod_oc4j, you need to consider the type of environment where Oracle HTTP Server runs. Use the following guidelines to help determine which configuration options to use with mod_oc4j:

- For a Oracle Application Server cluster setup, with multiple identical machines running Oracle HTTP Server and OC4J, the round robin with local affinity algorithm is preferred. Using this configuration, an external router distributes requests to multiple machines running Oracle HTTP Server and OC4J. In this case Oracle HTTP Server gains little by using mod_oc4j to route requests to other machines, except in the extreme case that all OC4J processes on the same machine are not available.
- For a tiered deployment, where one tier of machines contains Oracle HTTP Server and another contains OC4J instances that handle requests, the preferred algorithms are simple round robin and simple metric-based. To determine which of these two is best in a specific setup, you may need to experiment with each and compare the results. This is required because the results are dependent on system behavior and incoming request distribution.
- For a heterogeneous deployment, where the different application server instances run on nodes that have different characteristics, using the weighted round robin algorithm is preferred. Tune the number of OC4J processes running on each application server instance may allow you to achieve the maximum benefit. For example, a machine with a weight of 4 gets 4 times as many requests as a machine with a weight of 1, but if the system with a weight of 4 may not be running 4 times as many OC4J processes.

See Also:

- "mod_oc4j Load Balancing With OracleAS Clusters" on page 4-25
- *Oracle HTTP Server Administrator's Guide* for information on using weighted routing and selecting local affinity with mod_oc4j load balancing options.

J2EE High Availability

J2EE requests are fulfilled by OC4J, and involve OracleAS Web Cache and Oracle HTTP Server (mod_oc4j). Hence, the high availability of the J2EE service requires that these components are highly available. The high availability of Oracle HTTP Server and OracleAS Web Cache is discussed in the section "HTTP Service High Availability" on page 2-18. The high availability of OC4J is presented in the section "Features and Components for Middle Tier High Availability" on page 2-5.

EJB Client Routing

In EJB client routing, EJB classes take on the routing functionality that mod_oc4j provides for Oracle HTTP Server. Using the Active Components for Java (AC4J) architecture, EJBs can interact in a loosely-coupled fashion. This provides support for reliable asynchronous, disconnected, one-way request and response interactions, without the complexity of JMS programming. It automatically routes service requests to the appropriate service provider, and provides automatic security context propagation, authorization and identity impersonation. It also provides automatic exception routing and handling, which is integrated into the EJB framework.

Oracle Application Server Portal High Availability

An OracleAS Portal request's lifecycle is serviced by a number of OracleAS components. These are:

- OracleAS Web Cache
- Oracle HTTP Server and the following modules:
 - `mod_oc4j` (on middle and Infrastructure tiers)
 - `mod_osso` (on Infrastructure tier to access OracleAS Single Sign-On)
 - `mod_plsql` (on middle tier with OracleAS Portal DAD and Infrastructure tier with ORASSO DAD)
 - `mod_oradav` (on middle tier)
- OC4J (the Portal Page Engine runs as a stateless servlet)
- OracleAS Portal repository (contains OracleAS Portal schemas and also caches group memberships of users after their retrieval from Oracle Internet Directory)
- OracleAS Single Sign-On
- Oracle Internet Directory (including Oracle Delegated Administration Services and Oracle Directory Integration and Provisioning)
- Various web and database portlet providers

In order for OracleAS Portal to be highly available, all these components must be highly available individually. Of particular importance is the availability of Oracle Identity Management because OracleAS Portal uses it for portlet security and management functions.

Reference the following table to find out where you can find high availability information for each of the components mentioned above.

Table 2–4 High availability information for components involved in an OracleAS Portal request

Component	Where to find information
OracleAS Web Cache	See "Oracle Application Server Web Cache Clusters" on page 2-11.
Oracle HTTP Server	See: "Oracle Application Server Clusters" on page 2-6 "HTTP Service High Availability" on page 2-18
OC4J	See: "J2EE High Availability" on page 2-21 Note: The Portal Page Engine is stateless.
OracleAS Portal repository	See: Chapter 3, "Infrastructure High Availability" and Chapter 5, "Managing Infrastructure High Availability" of this book. <i>Oracle Application Server Portal Configuration Guide</i>
OracleAS Single Sign-On	See: Chapter 3, "Infrastructure High Availability" and Chapter 5, "Managing Infrastructure High Availability" of this book. <i>Oracle Identity Management Concepts and Deployment Planning Guide</i>

Table 2–4 (Cont.) High availability information for components involved in an OracleAS Portal request

Component	Where to find information
Oracle Internet Directory	See: Chapter 3, "Infrastructure High Availability" and Chapter 5, "Managing Infrastructure High Availability" of this book. <i>Oracle Internet Directory Administrator's Guide</i> <i>Oracle Identity Management Concepts and Deployment Planning Guide</i>
Web Provider	See: "Oracle Application Server Clusters" on page 2-6 "HTTP Service High Availability" on page 2-18 "OC4J Islands" on page 2-11 "Stateful Session EJB High Availability Using EJB Clustering" on page 2-14 "Oracle Application Server Web Cache Clusters" on page 2-11 (OracleAS Web Cache could be providing access to the provider)
Database Providers	For providers using <code>mod_plsql</code> , Oracle HTTP Server high availability is relevant. See "HTTP Service High Availability" on page 2-18 and "Oracle Application Server Clusters" on page 2-6. For database high availability, see: "Oracle Application Server Cold Failover Clusters" on page 3-7 "Oracle Application Server Active Failover Cluster (UNIX)" on page 3-14 See also: "Oracle Application Server Web Cache Clusters" on page 2-11 (OracleAS Web Cache could be providing access to the provider)

See Also: *Oracle Application Server Portal Configuration Guide*

Oracle Application Server Wireless High Availability

Typical Oracle Application Server Wireless (OracleAS Wireless) deployments in the enterprises, and particularly in telecom operator infrastructure, have very high availability and fault tolerance requirements. Oracle Application Server provides a framework and the mechanism to address these requirements.

OracleAS Wireless is integrated with this framework to extend these features to wireless deployments. Since OracleAS Wireless components are deployed as OC4J applications, Oracle HTTP Server can be configured to provide high availability to OracleAS Wireless applications. In addition, the OracleAS Wireless runtime is designed to handle session state replication so that client sessions failover transparently among multiple OC4J containers. Typical high availability deployments involve several network components, which in turn lead to considerations of configuration topology, performance, and security.

Refer to the "Wireless Gateway Configuration" chapter and the "Load Balancing and Failover" chapters of the *Oracle Application Server Wireless Administrator's Guide* for details.

Business Intelligence High Availability

The business intelligence components in Oracle Application Server include Oracle Application Server Reports Services and Oracle Application Server Discoverer. The following sections discuss the high availability of each:

- Oracle Application Server Reports Services High Availability
- Oracle Application Server Discoverer High Availability

Oracle Application Server Reports Services High Availability

At runtime, Oracle Application Server Reports Services (OracleAS Reports Services) consists of the following components shown in Table 2–5.

Table 2–5 Oracle Application Server Reports Services runtime components

Component	Characteristics
Reports Servlet	Translates client requests between HTTP and the Reports Server. It runs in-process in OC4J, and hence, is subject to the failures and high availability solutions for OC4J.
Reports Server	Processes client requests and forwards them to the Reports Engine. It runs as a standalone process and is stateful. Its state is not replicated to other Reports Server processes.
Reports Engine	Fetches requested data from data sources, formats the reports, and notifies the Reports Server that jobs are complete. It runs as a separate process from the Reports Server but is spawned by the latter to service requests. The Reports Engine is stateless. Failure of a Reports Engine process has minimal impact on the overall Reports Services as the Reports Server can spawn new Engine processes.

Of the three components described in the table above, the Reports Server process is the critical process that will adversely affect availability of OracleAS Reports Services if it fails. As it maintains state for client requests and the state is not replicated, its failure will cause client sessions to be lost.

Reports Server processes *are* monitored by OPMN. When OracleAS Reports Services is installed, it is registered with OPMN by default. Hence, OPMN can monitor and restart a Reports Server process if it fails. If you add more Reports Server processes after installation, you need to add them to the `opmn.xml` and `targets.xml` (for Application Server Console). See the book *Oracle Application Server Reports Services Publishing Reports to the Web* for instructions as well as for more high availability information.

The Reports Server makes database connections to the OracleAS Portal repository in the OracleAS Infrastructure as well as to Oracle Internet Directory. If any of these connections fail, Reports Server retries them before throwing an exception. If a successful connection is made, Reports Server need not be restarted. This also applies to Reports Servlet, which makes connections to Oracle Internet Directory.

High Availability Solution

To eliminate the single point of failure of the Reports Server process, perform multiple installations of OracleAS Reports Services (Business Intelligence and Forms Installation Type). These installations should also be installed on multiple nodes to protect from node failure.

For the OracleAS Infrastructure that is used by the OracleAS Reports Services installations, inclusive of Oracle Identity Management, use the Oracle Application Server Cold Failover Clusters or OracleAS Active Failover Cluster high availability solutions. These are described in Chapter 3, "Infrastructure High Availability".

See Also: *Oracle Application Server Reports Services Publishing Reports to the Web* ("Clustering Reports Servers" chapter)

Oracle Application Server Discoverer High Availability

Oracle Application Server Discoverer achieves high availability in the following ways:

- *Process monitoring and restart*
OPMN is configured to monitor and restart Oracle Application Server Discoverer processes on each middle tier node. See Chapter 4 of *Oracle Application Server Discoverer Configuration Guide*.
- *Load balancing*
OracleAS Web Cache can be set up to perform as a load balancer for Oracle Application Server Discoverer requests. See Chapter 5 of *Oracle Application Server Discoverer Configuration Guide*.

Oracle Application Server Forms Services High Availability

At runtime, Forms Services consist of the components listed in Table 2–6.

Table 2–6 Runtime Forms Services components

Component	Function
Forms Servlet	The Forms Servlet handles the initial application request and dynamically generates the start HTML file for the Forms generic Java Applet. If using OracleAS Single Sign-On, the Forms Servlet is also used to verify users' authentication.
Forms Listener Servlet	The Forms Listener Servlet is a dispatcher servlet that handles the communication between the Forms Java client in the client browser and the Forms runtime process in the middle tier server. The Forms Listener Servlet starts a Forms runtime process for each application request and user.
Forms Runtime Engine	The Forms Runtime Engine interprets the Forms application modules (fmx files) and executes the contained business logic. The Forms Runtime Engine also makes the database connection using SQLNet.

Forms Services doesn't exist as a dedicated server process on the middle tier, and therefore, all that is required to request and run a Forms application on the Web is the availability of a servlet container (OC4J) that is configured to run Forms Services.

Because Forms Services launches a dedicated Forms Runtime process for each user there is no transparent application failover. Once a user session is interrupted, the user has to restart the Forms Web application by issuing a new request to the Forms Servlet.

If a middle tier server crashes or a servlet session is interrupted, recover from either failure by restarting the application. To set up high availability for Forms, the following components can be used:

mod_oc4j - Handling the failure of an OC4J instance, Forms can be setup to load balance application requests between different OC4J instances. This ensures that an

application request can be routed to the next available OC4J instance if the current OC4J instance fails.

OracleAS Web Cache - Using OracleAS Web Cache as a HTTP load balancer allows you to distribute Forms requests between many Oracle Application Server instances that may or may not share the same Infrastructure installation. If one instance fails, then the next Forms application request gets routed to the next available instance. Each instance can also use `mod_oc4j` to load balance Forms sessions between OC4J instances.

Hardware load balancers - A hardware load balancer can be deployed in front of OracleAS Web Cache, thereby adding one more layer of load balancing for Forms requests. Or, they can also replace OracleAS Web Cache and load balance requests directly to Oracle HTTP Servers.

For the OracleAS Infrastructure that is used by Forms Services installations, inclusive of Oracle Identity Management, use the Oracle Application Server Cold Failover Clusters or Oracle Application Server Active Failover Cluster high availability solutions. These are described in Chapter 3, "Infrastructure High Availability".

For more information about Forms Services architecture and setup, refer to *Oracle Application Server Forms Services Deployment Guide*.

Oracle Application Server Integration High Availability

High availability for the Oracle Application Server 10g e-business integration products, Oracle Application Server 10g InterConnect and Oracle Application Server 10g ProcessConnect, are dependent on the various high availability solutions for the Infrastructure (see the section "High Availability Configurations for Infrastructure" on page 3-5). This is because InterConnect and ProcessConnect utilize the database in the Infrastructure to store and queue information. However, not all high availability solutions for the Infrastructure can be used by InterConnect or ProcessConnect. The following points elaborate further:

- Oracle Application Server 10g InterConnect

High availability is supported by the Oracle Application Server Cold Failover Clusters and Oracle Application Server Active Failover Cluster solutions. The `adapter.ini` and `hub.ini` files are populated with the host, port, and instance information of all the nodes in the cluster. When a node or database instance failure occurs, the InterConnect adapters are able to reconnect and continue message delivery without the need to republish the message and without losing the message at any stage of processing and delivery.

Detailed information on the `adapter.ini` and `hub.ini` contents can be found in chapter 8, under the section "RAC Support," in the *Oracle Application Server InterConnect User's Guide*.

- Oracle Application Server 10g ProcessConnect

ProcessConnect uses the database in the Infrastructure to store connection information instead of storing the information in files as for InterConnect. Hence, both the Oracle Application Server Cold Failover Clusters and Oracle Application Server Active Failover Cluster solutions for the Infrastructure enable high availability for ProcessConnect.

Middle Tier Recovery Solutions

Once a failure has occurred in your system, it is important to recover from that failure as quickly as possible. There are four main types of recovery solutions that you can use, depending on the type and severity of the failure.

Restarting Processes

Recovering from almost all types of failures requires restarting one or more failed processes in your system. There are three process restart scenarios:

- Automatic restart of processes: The failed processes are automatically restarted by OPMN upon detected failure. No manual intervention is required.
- Manually restart an individual process: This implies that the process failure does not affect any other middle tier or Infrastructure processes, and can be restarted individually.
- Manually restart all processes.

Most types of failures in both the middle tier and Infrastructure only require a process restart solution. Such failures include the death of OPMN, an Oracle Application Server Metadata Repository failure, or an Application Server Console crash.

Restoring from Cold Backup

Some failures require more involved recovery scenarios than simply restarting processes. In some cases, you will have to perform restoration operations based on cold backup procedures that you had previously implemented. These cold backups include installed OracleAS binaries. Cold backup restoration operations can be done for both the middle tier and the Infrastructure.

- Middle tier restoration from cold backup - Restoration of the entire Oracle Application Server middle tier, including the Oracle Home, configuration files, and database files, which were backed up after completing a clean and normal shutdown of all Oracle Application Server Infrastructure processes and the Oracle Application Server Metadata Repository.
- Infrastructure restoration from cold backup - Restoration of the entire Oracle Application Server Infrastructure instance, including the Oracle Home, configuration files, and data base files, which were backed up after completing a clean and normal shutdown of all Oracle Application Server Infrastructure processes and the Oracle Application Server Metadata Repository.

Failures that require the restoring from cold backup solution for recovery include node failure where the node needs to be completely replaced, and the deletion or corruption of Oracle software or binary files. Failures that require this type of recovery solution also then require the manual restart of all processes. For details about specific failure types and how to recover, see the *Oracle Application Server 10g Administrator's Guide*

Restoring from Online Backup

Depending on the type of failure your system is experiencing, you may need to restore your system from an online backup. There are four types of online backup restoration scenarios:

- Middle tier restoration from online backup - Restoration of the Oracle Application Server configuration files, which were backed up while processes were up and running on the middle tier. This also includes restoring a stamped image, which may require additional steps to complete the restoration.

- Infrastructure restoration from online backup - Restoration of the Oracle Application Server Infrastructure configuration files, which were backed up after completing a proper online backup of the Oracle Application Server instance and Oracle Application Server Metadata Repository.
- Oracle Application Server Metadata Repository restoration from online backup - Restoration of the Oracle Application Server Metadata Repository taken from a proper online backup. Complete recovery is required of the database component.
- Infrastructure configuration files restoration from online backup - Restoration of the Infrastructure configuration files taken from an online backup.

Failures that require restoration from online backup solutions for recovery include data failure in the Oracle Application Server Metadata Repository, and deletion or corruption of Oracle Application Server component runtime configuration files. Failures that require this type of solutions also then require one or more processes to be restarted. For details about specific failure types and how to recover, see the *Oracle Application Server 10g Administrator's Guide*.

Disaster Recovery

Disaster recovery (DR) refers to how a system recovers after a catastrophic site failure. Catastrophic failures include earthquakes, tornadoes, floods, and fires. On the most basic level, DR involves replicating an entire site, not just pieces of hardware or subcomponents. The service-level requirements for DR depend on the business applications. Some applications may not have any disaster recovery requirements. Others may simply have backup data tapes from which they would rebuild a new working site over a period of time. Still others may have requirements to begin operations with a few days or hours after the disaster. The most stringent requirement is to keep the services running despite the disaster.

The Oracle Application Server disaster recovery solution consists of two identically configured sites. Both sites may be dispersed geographically, and if so, they are connected via a network. When the primary site becomes unavailable due to disaster, the secondary site can become operational within a reasonable amount of time. Client requests are always routed to the site in the production role. After a failover or switchover operation occurs due to an outage, client requests are routed to another site that assumes the production role. Each site contains identical middle tier servers, which are also identical between the two sites. The site that is in the production role contains a production backend customer database and production Oracle Application Server Metadata Repository configured using the cold failover cluster Infrastructure high availability solution to protect from host failure. The site in the standby role contains a physical standby of the Oracle Application Server Metadata Repository. Database switchover and failover functions allow the roles to be traded between sites.

See Also: Chapter 6, "Oracle Application Server Disaster Recovery"

DCM Archive/Recover

The DCM archive and recovery feature allows you to take a snapshot of your system configuration. Taken at a time when everything is working properly and optimally, you can restore the system to this previous configuration in the event of a failure. In response to a catastrophic failure, the snapshot can be restored to a system in a remote location.

See Also: *Distributed Configuration Management Reference Guide* for instructions on how to perform archive and recover operations.

Configuration Cloning

Using the DCM tool, `dcmtcl`, you can clone the configuration of an existing Oracle Application Server instance to a new instance. The new instance will then have the exact same configuration as the first instance, thereby reducing the possibility of introducing configuration errors in the setup of the new instance.

The cloning process involves creating a new DCM archive and applying it to a new instance. This is different from restoring a DCM archive to the same instance the archive was created from.

Note: Only configurations managed by DCM can be cloned using the `dcmtcl` archive commands. DCM currently manages configuration data for the OC4J, OHS, OPMN, and JAZN components.

See Also: The chapter on archiving configurations in the *Distributed Configuration Management Reference Guide*.

Infrastructure High Availability

This chapter focuses on the high availability aspects of the Oracle Application Server 10g Infrastructure. It discusses the features and architectural solutions for high availability of the Infrastructure and is divided into the following sections:

- Oracle Application Server 10g Infrastructure Overview
- Oracle Application Server 10g Infrastructure Components
- High Availability Configurations for Infrastructure

Oracle Application Server 10g Infrastructure Overview

Oracle Application Server 10g provides a completely integrated infrastructure and framework for development and deployment of enterprise applications. An Oracle Application Server 10g Infrastructure installation type provides centralized product metadata, security and management services, and configuration information and data repositories for the Oracle Application Server 10g middle tier. By integrating the Infrastructure services required by the middle tier, time and effort required to develop enterprise applications are reduced. In turn, the total cost of developing and deploying these applications is reduced, and the deployed applications are more reliable.

The Oracle Application Server 10g Infrastructure provides the following overall services:

- **Product Metadata Service**

Oracle Application Server 10g Infrastructure stores all application server metadata required by Oracle Application Server 10g middle tier instances. This data is stored in an Oracle9i database, thereby leveraging the robustness of the database to provide a reliable, scalable, and easy-to-manage metadata repository.

- **Security Service**

The security service provides a consistent security model and identity management for all applications deployed on Oracle Application Server 10g. The service enables centralized authentication using single sign-on, Web-based administration through the Oracle Delegated Administration Services, and centralized storage of user authentication credentials. The Oracle Internet Directory is used as the underlying repository for this service.

- **Management Service**

This service is used by Distributed Configuration Management to manage and administer Oracle Application Server 10g middle tier instances and the Oracle Application Server 10g Infrastructure. It is also used to administer clustering services for the middle tier. Application Server Console reduces the total

administrative cost by centralizing the management of deployed J2EE applications.

Oracle Application Server 10g Infrastructure Components

The Oracle Application Server 10g Infrastructure consists of several components that contribute to its role and function. These components work with each other to provide the Infrastructure's product metadata, security, and management services. This section describes these Infrastructure components, which are:

- Oracle Application Server Metadata Repository
- Oracle Identity Management
- Oracle HTTP Server
- Oracle Application Server Containers for J2EE (OC4J)
- Oracle Enterprise Manager - Application Server Console

Oracle Application Server Metadata Repository

Oracle Application Server Metadata Repository is an Oracle9i Enterprise Edition database server and stores component-specific information that is accessed by the Oracle Application Server middle tier or Infrastructure components as part of their application deployment. The end user or the client application does not access this data directly. For example, a Portal application on the middle tier accesses the Portal metadata as part of the Portal page assembly aggregation. Metadata also includes demo data for many Oracle Application Server components, such as data used by the Order Management Demo for BC4J.

Oracle Application Server metadata and customer or application data can co-exist in the Oracle Application Server Metadata Repository, the difference is in which applications are allowed to access them.

The Oracle Application Server Metadata Repository stores three main types of metadata corresponding to the three main Infrastructure services described in the section "Oracle Application Server 10g Infrastructure Overview". These types of metadata are:

- product metadata
- identity management metadata
- management metadata

Table 3–1 shows the Oracle Application Server components that store and use these types of metadata during application deployment.

Table 3–1 Metadata and Infrastructure Components

Type of Metadata	Infrastructure Components Involved
Product metadata (includes demo data)	Oracle Application Server Metadata Repository
Identity Management metadata	OracleAS Single Sign-On, Oracle Internet Directory, Oracle Application Server Certificate Authority
Management metadata	Distributed Configuration Management, Oracle Enterprise Manager

When to Use Oracle Application Server Metadata Repository

Oracle Application Server Metadata Repository (OracleAS Metadata Repository) is needed for all application deployments except for those using the J2EE and Web Cache installation type. Oracle Application Server provides three middle tier installation options:

- **J2EE and Web Cache:** Installs Oracle HTTP Server, Oracle Application Server Containers for J2EE (OC4J), Oracle Application Server Web Cache (OracleAS Web Cache), Web Services, Oracle Business Components for Java (BC4J), and Application Server Console.
- **Portal and Wireless:** Installs all components of J2EE and OracleAS Web Cache, plus UDDI, Oracle Application Server Portal (OracleAS Portal), Oracle Application Server Syndication Services (OracleAS Syndication Services), Oracle Ultra Search, and Oracle Application Server Wireless (OracleAS Wireless).
- **Business Intelligence and Forms:** Installs all components of J2EE and OracleAS Web Cache, OracleAS Portal and Oracle Application Server 10g Wireless, plus Oracle Application Server Forms Services, Oracle Application Server Reports Services, Oracle Application Server Discoverer, and Oracle Application Server Personalization.

Integration components, such as Oracle Application Server ProcessConnect, Oracle Application Server InterConnect, and Oracle Workflow are installed on top of any of these middle tier install options.

The Distributed Configuration Management (DCM) component enables middle tier management, and stores its metadata in the OracleAS Metadata Repository for both the Portal and Wireless, and the Business Intelligence and Forms install options. For the J2EE and Web Cache installation type, by default, DCM uses a file-based repository. If you choose to associate the J2EE and Web Cache installation type with an Infrastructure, the file-based repository is moved into the OracleAS Metadata Repository.

Note: The OracleAS Metadata Repository can be installed in an existing Real Application Clusters database (without the Oracle Identity Management components of the Infrastructure). The *Oracle Application Server 10g Installation Guide* provides information on this installation scenario.

See Also: *Oracle Application Server 10g Installation Guide* for information on the Oracle Application Server 10g installation details.

Oracle Identity Management

The Oracle Identity Management framework in the Infrastructure includes the following components:

- Oracle Internet Directory
- Oracle Application Server Single Sign-On

See Also: *Oracle Identity Management Concepts and Deployment Planning Guide*

Oracle Internet Directory

Oracle Internet Directory is Oracle's implementation of a directory service using the Lightweight Directory Access Protocol (LDAP) version 3. It runs as an application on the Oracle9i database and utilizes the database's high performance, scalability, and high availability.

Oracle Internet Directory provides a centralized repository for creating and managing users for the rest of the Oracle Application Server 10g components such as OC4J, Oracle Application Server 10g Portal, or Oracle Application Server 10g Wireless. Central management of user authorization and authentication enables users to be defined centrally in Oracle Internet Directory and shared across all Oracle Application Server 10g components.

Oracle Internet Directory is provided with a Java-based management tool (Oracle Directory Manager), a Web-based administration tool (Oracle Delegated Administration Services) for trusted proxy-based administration, and several command-line tools. Oracle Delegated Administration Services provide a means of provisioning end users in the Oracle Application Server 10g environment by delegated administrators who are not the Oracle Internet Directory administrator. It also allows end users to modify their own attributes.

Oracle Internet Directory also enables Oracle Application Server 10g components to synchronize data about users and group events, so that those components can update any user information stored in their local application instances.

See Also: *Oracle Internet Directory Administrator's Guide* for more information.

Oracle Application Server Single Sign-On

OracleAS Single Sign-On is a multi-part environment which is made up of both middle tier and database functions allowing for a single user authentication across partner applications. A partner application can be achieved either by using the SSOSDK or via the Apache `mod_ossso` module. This module allows Apache (and subsequently URLs) to be made partner applications.

OracleAS Single Sign-On is fully integrated with Oracle Internet Directory, which stores user information. It supports LDAP-based user and password management through Oracle Internet Directory.

OracleAS Single Sign-On supports Public Key Infrastructure (PKI) client authentication, which enables PKI authentication to a wide range of Web applications. Additionally, it supports the use of X.509 digital client certificates and Kerberos Security Tickets for user authentication.

By means of an API, OracleAS Single Sign-On can integrate with third-party authentication mechanisms such as Netegrity Site Minder.

See Also: *Oracle Application Server Single Sign-On Administrator's Guide*. (This guide also includes Identity Management replication instructions.)

Oracle HTTP Server

The Infrastructure installation type installs Oracle HTTP Server for the Infrastructure. This is used to service requests from other distributed components of the Infrastructure and middle tier instances. In the Infrastructure, Oracle HTTP Server services requests for OracleAS Single Sign-On and Oracle Delegated Administration

Services. The latter is implemented as a servlet in an OC4J process in the Infrastructure.

See Also: *Oracle HTTP Server Administrator's Guide*

Oracle Application Server Containers for J2EE (OC4J)

In the Infrastructure, OC4J is installed in the Infrastructure to run Oracle Delegated Administration Services and OracleAS Single Sign-On. The former runs as a servlet in OC4J.

Oracle Delegated Administration Services provide a self-service console (for end users and application administrators) that can be customized to support third-party applications. In addition, it provides a number of services for building customized administration interfaces that manipulate Oracle Internet Directory data. Oracle Delegated Administration Services are a component of Oracle Identity Management.

See Also: *Oracle Internet Directory Administrator's Guide* for more information about Oracle Delegated Administration Services.

Oracle Enterprise Manager - Application Server Console

Oracle Enterprise Manager - Application Server Console (Application Server Console) provides a Web-based interface for managing Oracle Application Server components and applications. Using the Oracle Application Server Console, you can do the following:

- monitor Oracle Application Server components, Oracle Application Server middle tier and Infrastructure instances, OracleAS Clusters, and deployed J2EE applications and their components
- configure Oracle Application Server components, instances, OracleAS Clusters, and deployed applications
- operate OracleAS components, instances, OracleAS Clusters, and deployed applications
- manage security for OracleAS components and deployed applications

For more information on Oracle Enterprise Manager and its two frameworks, see *Oracle Enterprise Manager Concepts*.

See Also: *Oracle Application Server 10g Administrator's Guide* - provides descriptions on Application Server Console and instructions on how to use it.

High Availability Configurations for Infrastructure

As described earlier the Oracle Application Server 10g Infrastructure provides the following services

- product metadata
- security service
- management service

From an availability standpoint, these services are provided by the following components, which must all be available to guarantee availability of the Infrastructure:

- OracleAS Metadata Repository

- Oracle Net listener
- Oracle HTTP Server
- For Oracle Identity Management:
 - Oracle Internet Directory and Oracle Internet Directory monitor
 - OC4J Oracle Delegated Administration Services instance
 - OracleAS Single Sign-On
- For Oracle Management Services:
 - Distributed Configuration Management

For the Infrastructure to provide all essential services, all of the above components must be available. On UNIX platforms, this means that the processes associated with these components must be up and active. Any high availability solution must be able to detect and recover from any software failures of any of the processes associated with the Infrastructure components. It must also be able to detect and recover from any hardware failures on the hosts that are running the Infrastructure.

In Oracle Application Server 10g, all of the Infrastructure processes, except the database, its listener, and Application Server Console, are started, managed, and restarted by the Oracle Process Management and Notification (OPMN) framework. This means any failure of an OPMN-managed process is handled internally by OPMN. OPMN is automatically installed and configured at install time. However, any database process failure or database listener failure is not handled by OPMN. Also, failure of any OPMN processes leaves the Infrastructure in a non-resilient mode if the failure is not detected and appropriate recovery steps taken.

OracleAS provides two solutions to provide intrasite high availability for the Infrastructure. These are:

- Oracle Application Server Cold Failover Clusters
- Oracle Application Server Active Failover Cluster (UNIX)

These intrasite high availability solutions provide protection from local hardware and software failures that cannot be detected and recovered by OPMN. Examples of such failures are a system panic or node crash. These solutions, however, cannot protect the Infrastructure from site failures or media failures, which result in damage to or loss of data.

Oracle Application Server 10g provides a disaster recovery solution to protect against disasters and site failures. This solution is described in Chapter 6, "Oracle Application Server Disaster Recovery".

A site failure or disaster will most likely affect all the systems including middle tiers, Infrastructure, and backend databases. Hence, the disaster recovery solution also provides mechanisms to protect the middle tier and the Infrastructure database.

In short, the intrasite high availability solutions, OracleAS Cold Failover Cluster and OracleAS Active Failover Cluster, provide resilience for only the OracleAS Infrastructure from local hardware and software failures. The middle tier can continue to function with a resilient Infrastructure. The disaster recovery solution, on the other hand, deals with a complete site failure, which requires failing over not only the Infrastructure but also the middle tier. The intrasite high availability solutions for the Infrastructure are discussed in the following sections.

Oracle Application Server Cold Failover Clusters

The Oracle Application Server Cold Failover Clusters (OracleAS Cold Failover Cluster) solution for the Infrastructure uses a two node hardware cluster as depicted in Figure 3-1, "Normal operation of OracleAS Cold Failover Cluster solution" below.

Terminology

For the purpose of describing the solution, it is important to clarify the following terminology within the context of the OracleAS Cold Failover Cluster solution.

Hardware Cluster A cluster, in generic definition, is a collection of loosely coupled computers (called nodes) that provides a single view of network services (for example: an IP address) or application services (for example: databases, web servers) to clients of these services. Each node in a cluster is a standalone server that runs its own processes. These processes can communicate with one another to form what looks like a single system that cooperatively provides applications, system resources, and data to users. This type of clustering offers several advantages over traditional single server systems for highly available and scalable applications.

Hardware clusters are clusters that achieve high availability and scalability through the use of additional hardware (cluster interconnect, shared storage) and software (health monitors, resource monitors). (The cluster interconnect is a private link used by the hardware cluster for heartbeat information to detect node death.) Due to the need for additional hardware and software, hardware clusters are commonly provided by hardware vendors such as SUN, HP, IBM, and Dell. While the number of nodes that can be configured in a hardware cluster is vendor dependent, for the purpose of Oracle Application Server 10g Infrastructure High Availability using the Oracle Application Server Cold Failover Clusters solution, only two nodes are required. Hence, this document assumes a two-node hardware cluster for that solution.

Failover Failover is the process by which the hardware cluster automatically relocates the execution of an application from a failed node to a designated standby node. When a failover occurs, clients may see a brief interruption in service and may need to reconnect after the failover operation has completed. However, clients are not aware of the physical server from which they are provided the application and data. The hardware cluster's software provides the APIs to automatically start, stop, monitor, and failover applications between the two nodes of the hardware cluster.

Primary Node The node that is actively executing one or more Infrastructure installations at any given time. If this node fails, the hardware cluster automatically fails the Infrastructure over to the secondary node. Since the primary node runs the active Infrastructure installation(s), it is considered the "hot" node.

Secondary Node This is the node that takes over the execution of the Infrastructure if the primary node fails. Since the secondary node does not originally run the Infrastructure, it is considered the "cold" node. And, because the application fails from a hot node (primary) to a cold node (secondary), this type of failover is called cold failover.

Logical or Virtual IP To present a single system view of the cluster to network clients, hardware clusters use what is called a logical or virtual IP address. This is a dynamic IP address that is presented to the outside world as the entry point into the cluster. The hardware cluster's software manages the movement of this IP address between the two physical nodes of the cluster while the external clients connect to this IP address without the need to know which physical node this IP address is currently active on. In a typical two-node cluster configuration, each physical node has its own physical IP

address and hostname, while there could be several logical IP addresses, which float or migrate between the two nodes. For a given OracleAS Infrastructure installation, the logical IP/virtual name associated with that installation is the IP/name that is used by the clients to connect to the Infrastructure. Refer to the *Oracle Application Server 10g Installation Guide* for more information on the installation process.

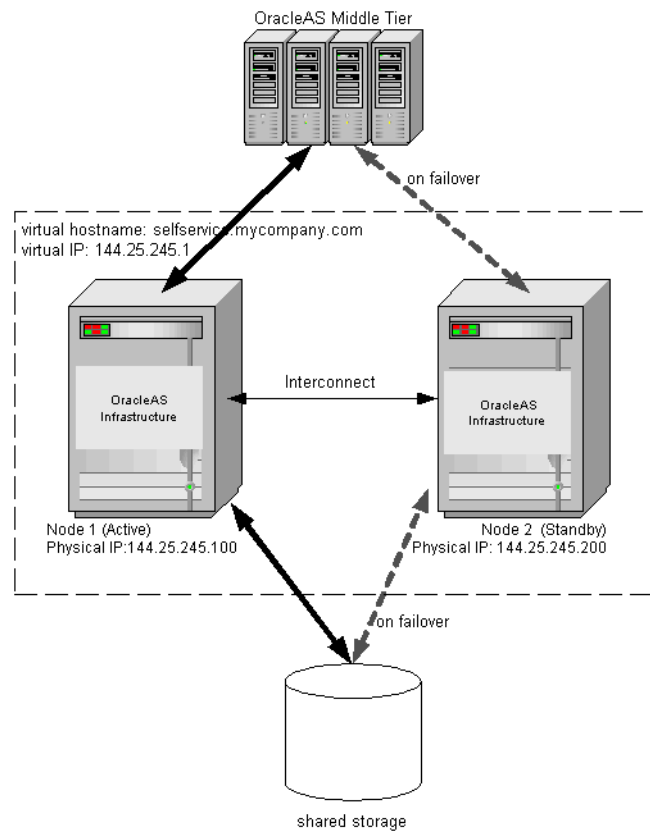
Virtual Hostname The virtual hostname is the hostname associated with the logical or virtual IP. This is the name that is chosen to give the OracleAS middle tier a single system view of the hardware cluster. This name-IP entry must be added to the DNS that the site uses, so that the middle tier nodes can associate with the Infrastructure without having to add this entry into their local `/etc/hosts` (or equivalent) file. For example, if the two physical hostnames of the hardware cluster are `node1.mycompany.com` and `node2.mycompany.com`, the single view of this cluster can be provided by the name `selfservice.mycompany.com`. In the DNS, `selfservice` maps to the logical IP address of the Infrastructure, which floats between `node1` and `node2` without the middle tier knowing which physical node is active and servicing the requests.

Note: Whenever the phrase "virtual name" is used in this document, it is assumed to be associated with the logical IP address. In cases where just the IP address is needed or used, it will be explicitly stated so.

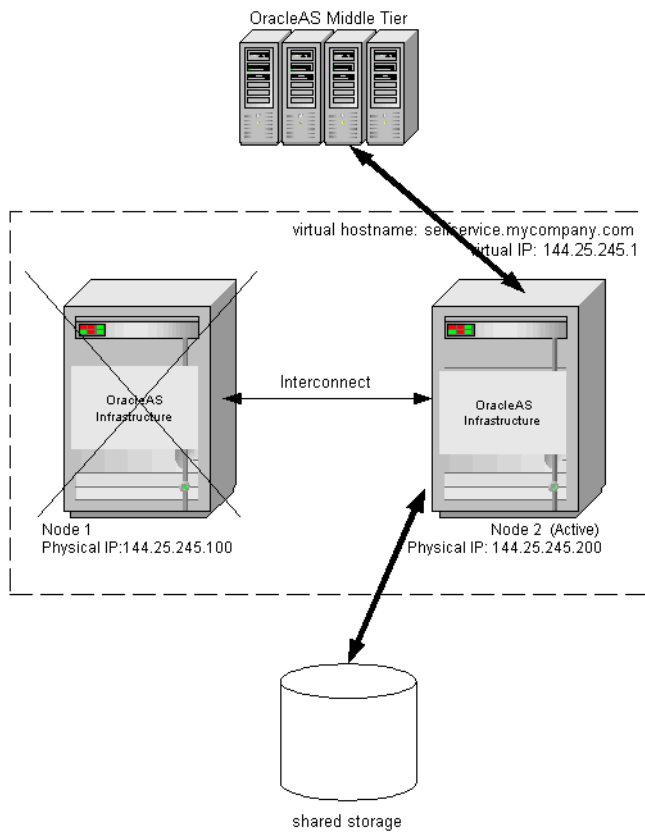
Shared Storage Even though each hardware cluster node is a standalone server that runs its own set of processes, the storage subsystem required for any cluster-aware purpose is usually shared. Shared storage refers to the ability of the cluster to be able to access the same storage, usually disks, from both the nodes. While the nodes have equal access to the storage, only one node, the primary node, has active access to the storage at any given time. The hardware cluster's software grants the secondary node access to this storage if the primary node fails. For the OracleAS Infrastructure, its `ORACLE_HOME` is on such a shared storage file system. This file system is mounted by the primary node; if that node fails, the secondary node takes over and mounts the file system. In some cases, the primary node may relinquish control of the shared storage, such as when the hardware cluster's software deems the Infrastructure as unusable from the primary node and decides to move it to the secondary.

Architecture (UNIX)

Figure 3-1 shows the layout of the two-node cluster for the OracleAS Cold Failover Cluster high availability solution. The two nodes are attached to shared storage. For illustration purposes, a virtual/logical IP address of 144.25.245.1 is active on physical Node 1. Hence, Node 1 is the primary or active node. The virtual name `selfservice.mycompany.com` is mapped to this virtual IP address, and the middle tier associates the Infrastructure with `selfservice.mycompany.com`.

Figure 3–1 Normal operation of OracleAS Cold Failover Cluster solution

In normal operating mode, the hardware cluster's software enables the virtual IP 144.25.245.1 on physical Node 1 and starts all Infrastructure processes (database, database listener, Oracle Enterprise Manager process, and OPMN) on that node. OPMN then starts, monitors, and restarts, if necessary, any of the following failed Infrastructure processes: Oracle Internet Directory, OC4J instances, and Oracle HTTP Server.

Figure 3–2 Infrastructure after primary node failover

If the primary node fails, the virtual IP address 144.25.245.1 is manually enabled on the secondary node (Figure 3–2). All the Infrastructure processes are then started on the secondary node. The middle tier processes accessing the Infrastructure will see a temporary loss of service as the virtual IP and the shared storage are moved over and the database, database listener, and all other Infrastructure processes are started. Once the processes are up, middle tier processes that were retrying during this time are reconnected. New connections are not aware that a failover has occurred.

While the hardware cluster framework can start, monitor, detect, restart, or failover Infrastructure processes, these actions are not automatic and involve some scripting or simple programming. Required scripts are described in Chapter 5, "Managing Infrastructure High Availability".

For information on setting up and operating the OracleAS Cold Failover Cluster solution for the Infrastructure, see *Oracle Application Server 10g Installation Guide*. This guide covers pre-installation and installation tasks.

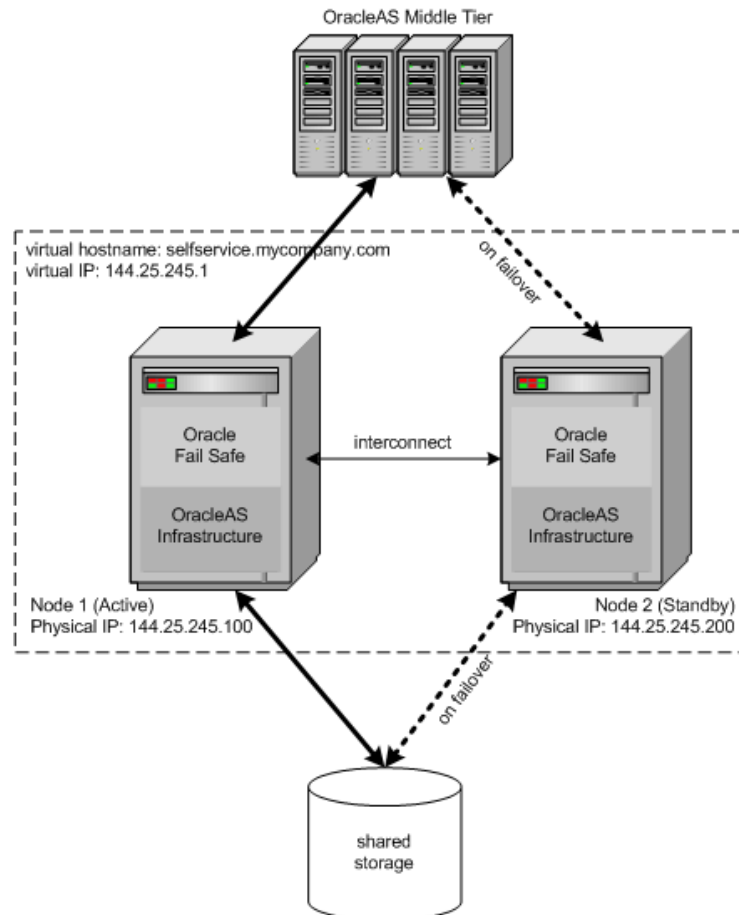
Architecture (Windows)

The OracleAS Cold Failover Cluster solution consists of a two-node cluster accessing a shared disk (see Figure 3–3) that contains the Infrastructure's data files. At any point in time, only one node is active. During normal operation, the second node is on standby. OracleAS middle tier components access the cluster through a virtual hostname that is mapped to a virtual IP in the subnet. In the example in Figure 3–3, the virtual hostname `selfservice.mycompany.com` and virtual IP 144.25.245.1 are used. When a failover occurs from node 1 to node 2, these virtual hostname and IP are

moved to the standby node, which now becomes the active node. The failure of the active node is transparent to the OracleAS middle tier components.

Note: Only static IP addresses can be used in the OracleAS Cold Failover Cluster solution for Windows.

Figure 3–3 Oracle Application Server Cold Failover Clusters solution for Windows



The concepts explained in the previous section (OracleAS Cold Failover Cluster for UNIX) are also applicable for the Windows OracleAS Cold Failover Cluster solution, which uses Microsoft Cluster Server software for managing high availability for the hardware cluster. Additionally, Oracle Fail Safe is used in conjunction with Microsoft Cluster Server to configure the following components:

- virtual hostname and IP
- OracleAS Infrastructure database
- Oracle Process Management and Notification service
- Application Server Console

Central to the Windows OracleAS Cold Failover Cluster solution is the concept of resource groups. A group is a collection of resources defined through Oracle Fail Safe. During failover from the active node to the standby node, the group, and hence, the resources in it, failover as a unit. During installation and configuration of the OracleAS

Cold Failover Cluster, a single group is defined for the solution. This group consists of the following:

- virtual IP for the cluster
- virtual hostname for the cluster
- shared disk
- Infrastructure database
- TNS listener for the database
- OPMN
- Application Server Console

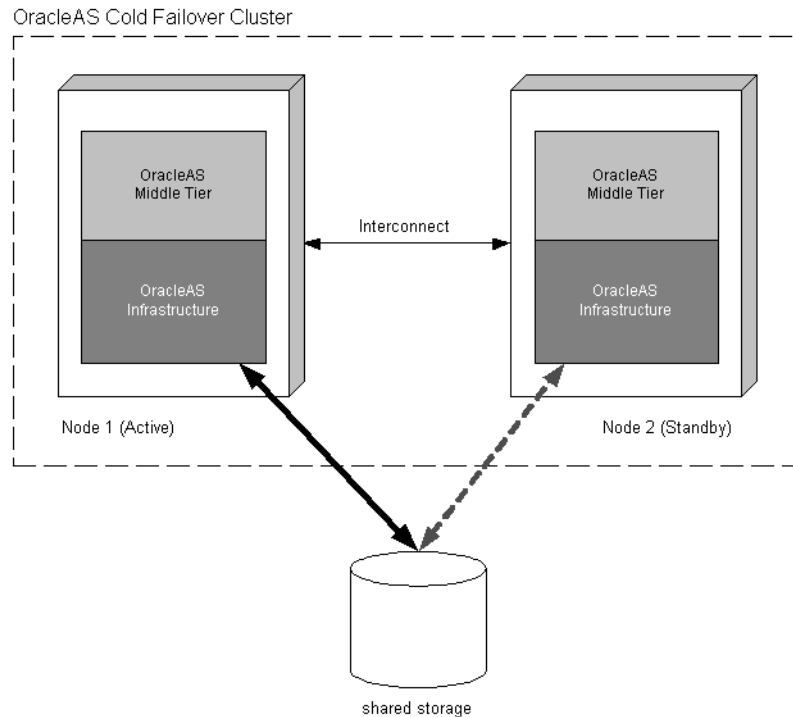
The integration of Oracle Fail Safe and Microsoft Cluster Server provide an easy to manage environment and automatic failover functionality in the OracleAS Cold Failover Cluster solution. The Infrastructure database, its TNS listener, and OPMN are installed as Windows services and are monitored by Oracle Fail Safe and Microsoft Cluster Server. Upon failure of any of these Windows services, Microsoft Cluster Server will try to restart the service three times (default setting) before failing the group to the standby. Additionally, OPMN monitors, starts, and restarts the Oracle Internet Directory, OC4J, and Oracle HTTP Server processes.

See Also: *Oracle Application Server 10g Installation Guide* for details on the installation process and requirements for installation.

Middle Tier on OracleAS Cold Failover Cluster Nodes

OracleAS middle tier can also be installed on the same node(s) as the OracleAS Cold Failover Cluster solution (see Figure 3–4). If the OracleAS middle tier is installed on both nodes of the OracleAS Cold Failover Cluster, both middle tier installations are concurrently active and servicing requests while the Infrastructure is active only on one of the nodes. Figure 3–4 provides a graphical depiction of this discussion.

Figure 3–4 OracleAS Middle Tier on same nodes as OracleAS Cold Failover Cluster solution



This set up has the following characteristics:

- The middle tiers are installed on local storage, and a load balancer should be available in front of them to load balance between them.
- The middle tiers do not benefit from the failover capabilities of the hardware cluster system or OracleAS Cold Failover Cluster solution. They have their own ways of achieving high availability, as discussed in Chapter 2 and Chapter 4 of this guide.
- The middle tier instances (J2EE and Web Cache installation type) can be grouped together to form an Oracle Application Server 10g Cluster, benefiting from the high availability attributes of Oracle Application Server 10g Clusters as described in Chapter 2.
- On each node, port conflicts between the middle tier and the Infrastructure must be avoided. Port numbers used by the middle tier must be different from those used by the Infrastructure. Any conflict can be avoided at installation time using the `staticports.ini` file. See *Oracle Application Server 10g Installation Guide*.
- If the Infrastructure on a node experiences a software failure, the middle tier on the same node may still be serviceable. This can be true even after the Infrastructure fails over to the standby node in the OracleAS Cold Failover Cluster.

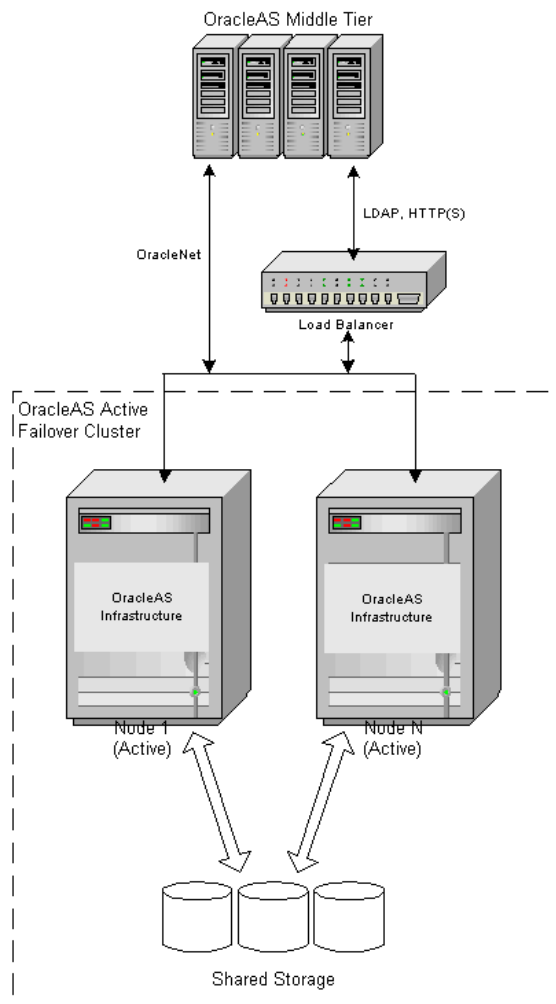
See Also: *Oracle Application Server 10g Installation Guide*

Oracle Application Server Active Failover Cluster (UNIX)

Note: Check OracleMetalink (<http://metalink.oracle.com>) for the most current certification status of this feature or consult your Oracle sales representative before deploying this feature in a production environment.

Oracle Application Server Active Failover Cluster (OracleAS Active Failover Cluster) provides a robust cluster architecture for the Infrastructure. It provides a more transparent high availability solution than the OracleAS Cold Failover Cluster solution. Because the nodes in the OracleAS Active Failover Cluster solution are all active, failover from one node to another is quick and requires no manual intervention. The active-active set up also provides scalability to the Infrastructure deployed on it. Figure 3-5 depicts the overall architecture of the solution.

Figure 3-5 OracleAS Active Failover Cluster high availability solution



In this solution, the Infrastructure software is installed identically on each node of a hardware cluster that is running OracleAS Active Failover Cluster technology. Each node has a local copy of the Infrastructure software (including Oracle Identity Management software) and an instance of the database. The database files are installed

in shared storage accessible by all nodes. The database instances open the database concurrently for read/write operations. The Infrastructure configuration files that are not in the database but in the file system are local to each node. These files contain node-specific configuration information.

The cluster is front-ended by a load balancer appliance. Oracle recommends that this load balancer be deployed in a fault-tolerant mode to maintain availability in case of load balancer failure. The load balancer appliance is used to direct non Oracle Net traffic from the middle tier to the Infrastructure. This traffic includes HTTP, HTTPS, and LDAP requests. The configuration of the load balancer is set to direct requests from the middle tier to any of the active Infrastructure nodes.

Note: Check <http://metalink.oracle.com> for information on supported external load balancers.

Oracle Net traffic from the middle tier does not go through the load balancer. This traffic is directed to the Infrastructure nodes via connect descriptors with multiple addresses in the address list. The address list is used to load balance certain Oracle Net traffic across the Infrastructure nodes. Oracle Net traffic include those initiated by:

- JDBC OCI
- JDBC thin
- dblinks using connect strings or tnsAlias
- tnsAlias-based access such as DADs (Database Access Descriptors)
- connect descriptor-based access

The OracleAS Active Failover Cluster high availability solution enables failover for failure of a whole node as well as failure of individual components of the node such as the database instance and Oracle Internet Directory.

The following considerations apply to this solution:

1. The OracleAS Active Failover Cluster is deployed on a hardware cluster.
2. All nodes in the cluster are peers in the following ways:
 - a. They run the same version of operating system.
 - b. They have the same version or compatible versions of all software such as the Java runtime.
 - c. ORACLE_HOME path and structure is the same on all nodes of the cluster.
 - d. ORACLE_SID has to be unique for each database instance on each node.
 - e. Service name has to be common for all database instances.
 - f. Identical port numbers for Infrastructure components.
3. Infrastructure components are in one cluster (not asymmetrically distributed) and in a single OracleAS Active Failover Cluster database. All nodes have identical configuration of Infrastructure components (OPMN, Oracle HTTP Server, OracleAS Single Sign-On, Oracle Enterprise Manager process, Oracle Internet Directory LDAP server, and Oracle Delegated Administration Services).

For information on setting up and operating the OracleAS Active Failover Cluster high availability solution for the Infrastructure, see Chapter 5, "Managing Infrastructure High Availability". The pre-installation and installation tasks for this high availability solution are provided in detail in *Oracle Application Server 10g Installation Guide*.

Load Balancer Configuration

In order for an OracleAS Active Failover Cluster to service Oracle Internet Directory LDAP and HTTP (for OracleAS Single Sign-On and Oracle Delegated Administration Services) requests, a load balancer is required for the OracleAS Active Failover Cluster configuration. The hostname of the load balancer virtual server is exposed as the hostname of the Infrastructure for these requests. This section describes the configuration requirements for the load balancer for the default installation of OracleAS Active Failover Cluster.

For high availability, the following is recommended:

- The load balancer should be deployed in a fault tolerant configuration. Two load balancers should be used. These fault tolerant load balancers should be identical in terms of their configuration and capacity. Their failover should be automatic and seamless from the middle tier's standpoint.
- The load balancer type used should be able to handle both HTTP and LDAP traffic in the default OracleAS Active Failover Cluster configuration described in this chapter. Any load balancing mechanism that supports only one of the protocols (for example, OracleAS Web Cache for HTTP) cannot be used in the default configuration.
- The load balancer should be accessible from all nodes of the OracleAS Active Failover Cluster deployment.
- The load balancer should be accessible from all machines that need to access the Infrastructure.
- The load balancer should not drop idle connections. Any timeouts associated with dropping of connection should be eliminated.

Two load balancer parameters are of primary importance for the OracleAS Active Failover Cluster configuration:

- The nodes to which the load balancer directs traffic.
- The persistence setting of the load balancer.

The recommended setting for the load balancer for the above two parameters are provided below in Table 3–2. Load balancers come in many flavors and each may have its own configuration mechanism. Consult your load balancer's documentation for the specific instructions to achieve these configurations.

Table 3–2 Recommended settings for load balancer

Deployment Phase	Traffic redirection Setting	Persistence Setting
OracleAS Active Failover Cluster installation	<ul style="list-style-type: none"> ■ Load balancer directs traffic to the node being installed and only to that node. 	NA

Table 3–2 (Cont.) Recommended settings for load balancer

Deployment Phase	Traffic redirection Setting	Persistence Setting
OracleAS Active Failover Cluster normal operations	<ul style="list-style-type: none"> ▪ Load Balancer directs traffic to all nodes of the OracleAS Active Failover Cluster configuration that are up and is configured to load balance this traffic. ▪ If not all hardware cluster nodes are used for the OracleAS Active Failover Cluster, the number of nodes to load balance traffic to may be less than the nodes in the hardware cluster . ▪ If any node of the OracleAS Active Failover Cluster is not available or if any of the Oracle Internet Directory, HTTP, or SSO processes is down, the load balancer should not direct traffic to these nodes. 	Session level persistence should be configured for LDAP and HTTP traffic.
OracleAS Active Failover Cluster node or process is brought down	<ul style="list-style-type: none"> ▪ If a node or an OracleAS Active Failover Cluster process on a node is being brought down, the current node should be disabled from the list of nodes the load balancer directs traffic to. Refer to your load balancer's documentation for the best way to do this. 	Session level persistence should be configured for LDAP and HTTP traffic.
middle tier association	<ul style="list-style-type: none"> ▪ When a new middle tier is being associated (such as during middle tier installation), the load balancer is required to direct traffic to just one node of the OracleAS Active Failover Cluster. This could be any node in the cluster. This is required only during the duration of the middle tier installation. Once middle tier association is done, the load balancer can be configured back to its previous state. Check with your load balancer documentation on accomplishing this without disrupting existing connections (primarily LDAP) to the OracleAS Active Failover Cluster. 	Session level persistence should be configured for LDAP and HTTP requests.

The persistence mechanism used should provide session level stickiness. By default, HTTP and Oracle Internet Directory requests both use the same virtual host address configured for the load balancer. Hence, the persistence mechanism used is available for both kinds of requests.

If the load balancer allows for the configuration of different persistence mechanisms for different server ports (LDAP and HTTP) for the same virtual server, then this is recommended strategy. In this case, a cookie-based persistence with session-level timeout is more suitable for the HTTP traffic. No persistence setting is required for the LDAP traffic.

If the load balancer does not allow specification of different persistence mechanisms for LDAP and HTTP, then the timeout value for session level stickiness should be configured based on the requirements of the deployed application. The timeout value should not be too high as chances of traffic from a given middle tier instance always being directed to the same node of the OracleAS Active Failover Cluster are higher. Alternatively, if the timeout is too low, the chances of a session timeout occurring for longer running operations that access the Infrastructure are higher.

The recommended default stickiness timeout is 60 seconds. This should be adjusted based on the nature of the deployment and the load balancing achieved across the OracleAS Active Failover Cluster nodes. It should be increased if session timeouts are experienced by Delegated Administration Services users. It should be decreased if even load balancing is not achieved.

Both the LDAP & HTTP traffic should be tested after configuration of the load balancer. This should be done from any machine outside the OracleAS Active Failover Cluster. The tests should have the following coverage:

- Access and test the Oracle Delegated Administration Services URL to test HTTP requests.
- Access and test the OracleAS Single Sign-On URL to test HTTP requests.
- Access and test the Oracle Internet Directory by running a few `ldapsearch` commands for LDAP requests.

The requests types above should be directed to different nodes of the OracleAS Active Failover Cluster. The desired operation(s) should complete successfully for the tests to be successful.

Managing and Operating Middle Tier High Availability

This chapter describes how to perform configuration changes and on-going maintenance of OracleAS Clusters. Because managing individual nodes involves some complexity, an OracleAS Cluster provides the ability to manage the nodes as a single entity, thereby reducing the management complexity. Instructions are provided for managing and configuring OracleAS Clusters using Oracle Enterprise Manager - Application Server Console (Application Server Console) and where required, using the `dcmctl` command line utility.

This chapter covers the following topics:

- Middle Tier High Availability Configuration Overview
- Managing and Configuring OracleAS Clusters
- Using a File-Based Repository with OracleAS Clusters
- OC4J Configuration with an OracleAS Cluster
- Oracle HTTP Server Configuration with OracleAS Clusters
- Security – Configuring Single Sign-On
- Advanced Clustering Configuration

Middle Tier High Availability Configuration Overview

Oracle Application Server supports different clustering configuration options to support high availability in the Oracle Application Server middle tier. OracleAS Clusters provide distributed configuration information and let multiple Oracle Application Server instances work together and behave as a single system to external clients. When configured to use redundant components throughout, OracleAS Clusters support a highly available system in which to deploy and run applications with no single point of failure.

Note: Only OracleAS instances of the J2EE and Web Cache installation type can be clustered as an OracleAS Cluster.

This section covers the following topics:

- Configuration Overview OracleAS Clusters Managed Using a Repository
- Manually Configured OracleAS Clusters Configuration Overview

- OracleAS Web Cache Cluster Overview

See Also: Chapter 2, "Middle Tier High Availability"

Configuration Overview OracleAS Clusters Managed Using a Repository

When administering an **OracleAS Cluster** that is managed using a repository, an administrator uses either Application Server Console or `dcmctl` commands to manage and configure common configuration information. The Oracle Application Server manageability components then replicate the common configuration information across all Oracle Application Server instances within the cluster. Using OracleAS Clusters, the common configuration information for the cluster is called the cluster-wide configuration.

Note: There is configuration information that can be configured individually, per Oracle Application Server instance within a cluster (these configuration options are also called **instance-specific parameters**).

Each application server instance in a OracleAS Cluster has the same base configuration. The base configuration contains the cluster-wide configuration and excludes instance-specific parameters.

This section covers the following:

- Oracle Application Server Clusters Managed Using Database Repository
- Oracle Application Server Clusters Managed Using File-Based Repository
- Common Tasks for OracleAS Cluster Configuration

Oracle Application Server Clusters Managed Using Database Repository

Oracle Application Server Clusters managed using a database repository utilize Oracle9i database to store configuration information and metadata, including both cluster-wide configuration information and instance-specific parameters.

Using a database repository protects configuration information by storing the information in the database. Using the database, combined with Oracle Application Server high availability solutions both protects configuration information and allows you to continue operations after system failures.

Oracle Application Server Clusters Managed Using File-Based Repository

Oracle Application Server Clusters managed using a file-based repository use the file system to store configuration information, including both cluster-wide configuration information and instance-specific parameters. Using Oracle Application Server Clusters managed using a file-based repository does not present a single point of failure; remaining Oracle Application Server instances within a cluster are available to service client requests when one or more Oracle Application Server instances is down.

Configuring and managing Oracle Application Server Clusters managed using a file-based repository requires that the administrator set up a farm and perform certain configuration tasks using the `dcmctl` command line utility.

See Also: "Using a File-Based Repository with OracleAS Clusters" on page 4-9

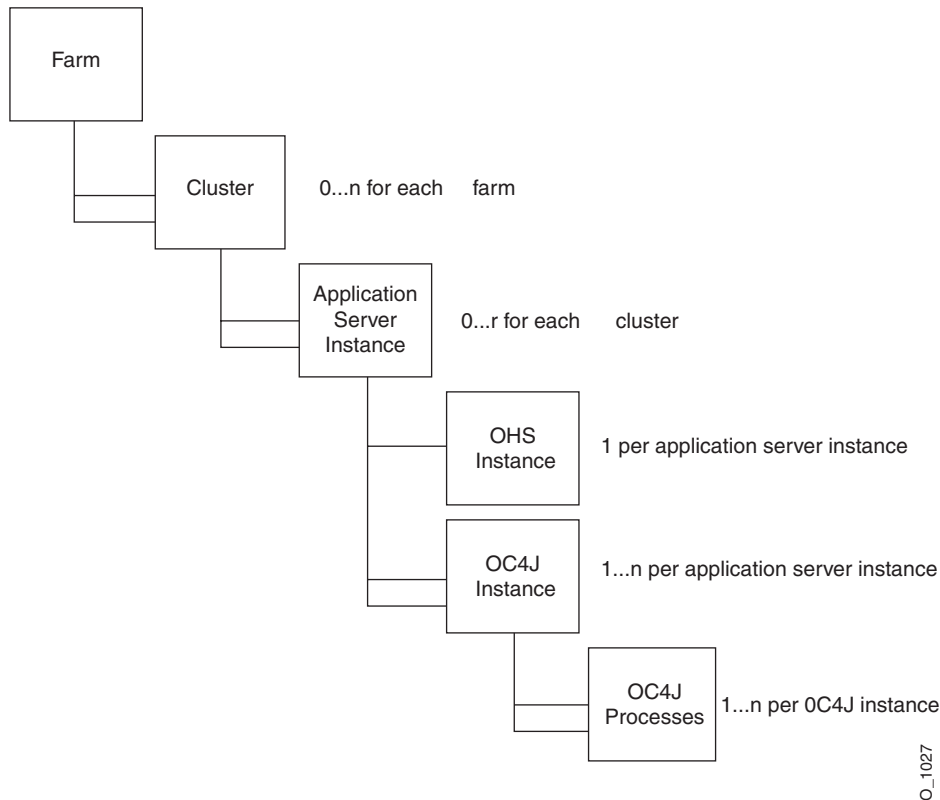
Common Tasks for OracleAS Cluster Configuration

Figure 4–1 shows the cluster configuration hierarchy, starting with an Oracle Application Server top-level farm for an OracleAS Cluster. This figure applies to both types of OracleAS Clusters: those managed using a file-based repository and those managed using a database repository.

Figure 4–1 shows the OracleAS Clusters configuration hierarchy, including the following:

- Clusters that contain Oracle Application Server instances
- Oracle Application Server instances containing a single Oracle HTTP Server and one or more OC4J instances
- OC4J instances containing the following:
 - One or more OC4J islands
 - One or more OC4J processes within OC4J islands
 - Deployed applications

Figure 4–1 Application Server Console Cluster Configuration Tree



Manually Configured OracleAS Clusters Configuration Overview

Manually configured OracleAS Clusters store configuration information in local configuration files and do not use either a database repository or a file-based repository. In a manually configured cluster, it is the administrator's responsibility to

synchronize the configuration of the Oracle Application Server instances that are part of the cluster.

OracleAS Web Cache Cluster Overview

In an OracleAS Web Cache cluster, multiple instances of OracleAS Web Cache operate as one logical cache to provide high availability. Each OracleAS Web Cache in the cluster is called a cache cluster member. A cache cluster can consist of two or more members. The cache cluster members communicate with one another to request cacheable content that is cached by another cache cluster member and to detect when a cache cluster member fails. When a cache cluster member detects the failure of another cluster member, the remaining cache cluster members automatically take over ownership of the content of the failing member. When the cache cluster member can be reached again, OracleAS Web Cache again reassigns the ownership of the content.

See Also: *Oracle Application Server Web Cache Administrator's Guide* for information on OracleAS Web Cache clustering and configuring a OracleAS Web Cache cluster.

Managing and Configuring OracleAS Clusters

This section describes how to create and use an OracleAS Cluster. The information in this section applies both to Oracle Application Server Clusters managed using a database repository and to those managed using a file-based repository.

This section covers the following topics:

- Creating and Managing OracleAS Clusters
- Managing Application Server Instances in an OracleAS Cluster

Note: As an alternative to using Application Server Console, you can create an OracleAS Cluster, add application server instances to the cluster, and manage the cluster using `dcmctl` commands.

See Also: *Distributed Configuration Management Reference Guide* for information on `dcmctl` commands

Creating and Managing OracleAS Clusters

The collection of Oracle Application Server instances within a single repository, either a database repository or a file-based repository is known as a farm. When an Oracle Application Server instance is part of a farm, you can view a list of all application server instances that are part of the farm when you start Application Server Console. The application server instances shown in the Standalone Instances area on the Application Server Console Farm Home Page are available to be added to a cluster.

This section covers the following:

- Associating an Instance with a Farm
- Creating OracleAS Clusters Using Application Server Console
- Managing OracleAS Clusters Using Application Server Console

Associating an Instance with a Farm

If you have not already done so during the Oracle Application Server installation process, you can associate an application server instance with a farm using one of the following techniques:

- Associating an Instance to be Managed Using a Database Repository
- Associating an Instance to be Managed Using a File-Based Repository

Associating an Instance to be Managed Using a Database Repository For a farm that uses a database repository, do the following to add an application server instance to the farm:

1. Navigate to the Oracle Application Server Instance Home Page.
2. In the **Home** area, select the **Infrastructure** link and follow the instructions for associating an application server instance with an Oracle Application Server Infrastructure.

See Also: *Oracle Application Server 10g Administrator's Guide*

Associating an Instance to be Managed Using a File-Based Repository For a farm that is managed using a file-based repository, you need to use the `dcmtl joinFarm` command to add a standalone application server instance to the farm.

See Also: "Joining a Farm Managed Using a File-Based Repository" on page 4-11

Creating OracleAS Clusters Using Application Server Console

You create a new OracleAS Cluster using the Application Server Console Farm Home Page. Application Server Console only shows the Farm Home Page when an Oracle Application Server instance is part of a farm.

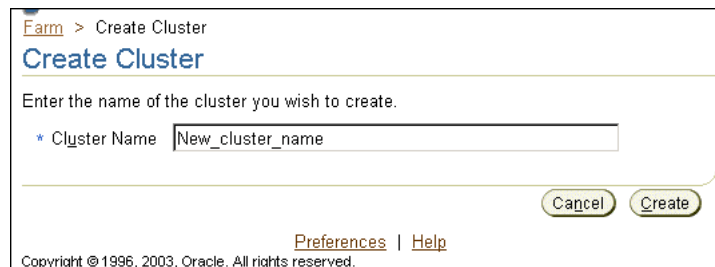
From the Farm Home page, create a new OracleAS Cluster as follows:

1. Navigate to the Farm Home Page.
2. Select the **Create Cluster** button.

Application Server Console displays the Create Cluster page.

Figure 4–2 shows the Create Cluster page.

Figure 4–2 Create Cluster Page



3. Enter a name for the new cluster and click **Create**. Each new cluster name within the farm must be unique.

A confirmation page appears.

4. Click **OK** to return to the Farm Home Page.

After creating a new OracleAS Cluster, the Farm Home page shows the cluster in the Clusters area. After creating a new cluster, the cluster is empty and does not include any application server instances. Use the **Join Cluster** button on the Farm Home page to add application server instances to the cluster.

Note: For Oracle Application Server Clusters using a file-based repository, Oracle recommends a size of four or less OracleAS instances per cluster.

See Also: Managing Application Server Instances in an OracleAS Cluster on page 4-7

Managing OracleAS Clusters Using Application Server Console

Figure 4–3 shows the Application Server Console Farm Home Page, including two clusters, cluster1 and cluster2.

Figure 4–3 Oracle Application Server 10g Farm Page



Table 4–1 lists the cluster control options available on the Farm Home Page.

Table 4–1 Oracle Application Server Farm Page Options

If you want to...	Then...
Start all application server instances in an OracleAS Cluster	Select the radio button next to the cluster and click Start
Restart all application server instances in an OracleAS Cluster	Select the radio button next to the cluster and click Restart
Stop all application server instances in an OracleAS Cluster	Select the radio button next to the cluster and click Stop
Delete an OracleAS Cluster, including any application server instances still included in the cluster.	Select the radio button next to the cluster and click Delete

Managing Application Server Instances in an OracleAS Cluster

Oracle Application Server replicates cluster-wide configuration within an OracleAS Cluster. This applies whether the cluster contains only one application server instance or many application server instances. To provide high availability for the Oracle Application Server middle tier using an OracleAS Cluster, a cluster needs to contain at least two application server instances.

This section covers the following topics:

- Adding an Application Server Instance to an OracleAS Cluster
- Removing an Application Server Instance from an OracleAS Cluster

See Also:

- "Cluster-Wide Configuration Changes and Modifying OC4J Instances" on page 4-19
- *Distributed Configuration Management Reference Guide* for information on `demctl` commands

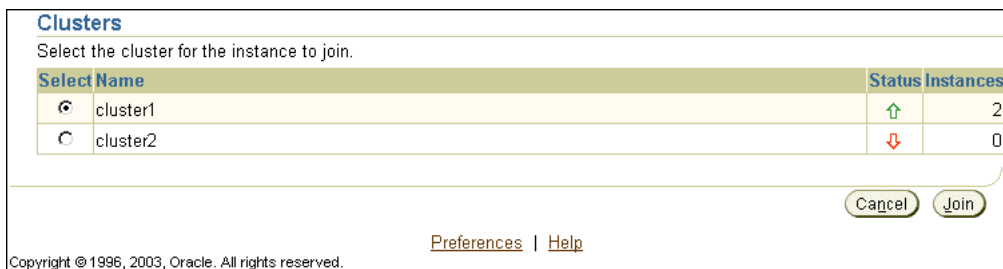
Adding an Application Server Instance to an OracleAS Cluster

To add an application server instance to a cluster:

1. Navigate to the Farm Home Page.
2. Select the radio button for the application server instance that you want to add to a cluster from the Standalone Instances section.
3. Click **Join Cluster**.

Figure 4-4 shows the Join Cluster page.

Figure 4-4 Join Cluster Page



4. Select the radio button of the cluster that you want the application server instance to join.
5. Click **Join**. OracleAS adds the application server instance to the selected cluster and then displays a confirmation page.
6. Click **OK** to return to the Farm Home Page.

Repeat these steps for each additional standalone application server instance you want to join the cluster.

Note the following when adding application server instances to an OracleAS Cluster:

- When adding application server instances to an OracleAS Cluster, the order that you add instances to the cluster is significant. The first application server instance

that joins the cluster is used as the base configuration for all additional application server instances that join the cluster. The base configuration includes all cluster-wide configuration information. It does not include instance-specific parameters.

- After the first application server instance joins the cluster, the base configuration overwrites existing cluster-wide configuration information for subsequent application server instances that join the cluster. Each additional application server instance, after the first, that joins the cluster inherits the base configuration specified for the first application server instance that joins the cluster.
- Before an application server instance joins a cluster, Application Server Console stops the instance. You can restart the application server instance by selecting the cluster link, selecting the appropriate instance from within the cluster, and then selecting the **Start** button.
- An application server instance is removed from the Standalone Instances area when the instance joins a cluster.
- To add multiple standalone application server instances to a cluster in a single operation, use the `dcmctl joinCluster` command.
- When an application server instance contains certain Oracle Application Server components, it is not clusterable. Use the `dcmctl isClusterable` command to test if an application server instance is clusterable. If the application server instance is not clusterable, then Application Server Console returns an error when you attempt to add the instance to a cluster.
- To be clusterable, all application server instances that are to be members of an OracleAS Cluster must be installed on the same operating system (this includes the same variant of UNIX).

Removing an Application Server Instance from an OracleAS Cluster

To remove an application server instance from a cluster, do the following:

1. Select the cluster in which you are interested on the Farm Home Page. This brings you to the cluster page.
2. Select the radio button of the application server instance to remove from the cluster and click **Remove**.

To remove multiple standalone application server instances, you need to repeat these steps multiple times.

Note the following when removing application server instances from an OracleAS Cluster:

- Before an application server instance leaves a cluster, Application Server Console stops the instance. After the operation completes, you restart the application server instance from the Standalone Instances area of the Farm Home Page.
- The `dcmctl leaveCluster` command removes one application server instance from the cluster at each invocation.
- When the last application server instance leaves a cluster, cluster-wide configuration information associated with the cluster is removed. The cluster is now empty and the base configuration is not set. Subsequently, Oracle Application Server uses the first application server instance that joins the cluster as the base configuration for all additional application server instances that join the cluster.
- You can remove an application server instance from the cluster at any time. The first instance to join a cluster does not have special properties. The base

configuration is created from the first instance to join the cluster, but this instance can be removed from the cluster in the same manner as the other instances.

Using a File-Based Repository with OracleAS Clusters

You can create OracleAS Clusters that do not depend on the database to store cluster-wide configuration and management information. Using a **file-based repository**, cluster-wide configuration information and related metadata is stored on the file system of an Oracle Application Server instance that is the **repository host** (host). Oracle Application Server instances that are part of a farm that uses a file-based repository depend on the repository host to store cluster-wide configuration information. After creating a farm that includes Oracle Application Server instances managed using a file-based repository, you can create OracleAS Clusters.

This section covers the following topics:

- Initializing File-Based Repository Host and Adding Instances to a Farm
- Managing Instances in a Farm That Uses a File-Based Repository

Initializing File-Based Repository Host and Adding Instances to a Farm

This section describes how to create a farm that uses a file-based repository and covers the following:

- Testing an Instance With `whichFarm` and Leaving a Farm
- Initializing the Repository Host Instance for a File-Based Repository
- Joining a Farm Managed Using a File-Based Repository

After a farm is created that includes Oracle Application Server instances managed using a file-based repository, you can create OracleAS Clusters using either Application Server Console or `dcmctl` commands.

Testing an Instance With `whichFarm` and Leaving a Farm

To create a file-based repository you need to start with a standalone application server instance. A standalone application server instance is an instance that is not associated with a farm. To verify that the Oracle Application Server instance that you want to use as the repository host for a file-based repository is a Standalone Instance, issue the following command:

```
% dcmctl whichFarm
```

This command returns the following when an instance is not associated with any farm:

```
Standalone OracleAS instance
```

Note: The `whichFarm` command returns detailed output when `dcmctl` runs with the verbose setting on. When verbose is off, `whichFarm` returns less output. Use the `dcmctl set -v off` to set the verbose mode off. Likewise, use `dcmctl set -v on` to set the verbose mode on.

Table 4–2 shows sample output from `whichFarm`. When an instance is not a standalone instance `whichFarm` returns information showing that the instance is part of a farm.

Table 4–2 Dcmctl whichFarm Command Verbose Output

Executing whichFarm on instance of type	Generates the following output...
Standalone Instance	% dcmctl whichFarm Standalone OracleAS instance
File-Based Repository – repository host instance (host)	% dcmctl whichFarm Farm Name: .private.904M13.dcm.repository Host Instance: 904M13.sc-sun.us.oracle.com Host Name: sc-sun.us.oracle.com Repository Type: Distributed File Based (host) SSL In Use: false
File-Based Repository – not the repository host instance	% dcmctl whichFarm Farm Name: .private.904M13.dcm.repository Host Instance: 904M13.sc-sun.us.oracle.com Host Name: sc-sun.us.oracle.com Repository Type: Distributed File Based SSL In Use: false
Database Repository – Infrastructure Instance	% dcmctl whichFarm Host Instance: 904M12infra.sc-sun.us.oracle.com Host Name: sc-sun.us.oracle.com Repository Type: Database (host)
Database Repository – not on the Infrastructure Instance	% dcmctl whichFarm Farm Name: tv1.us.oracle.com Host Instance: 904M12infra.sc-sun.us.oracle.com Host Name: sc-sun.us.oracle.com Repository Type: Database

When the instance that you want to use with a file-based repository is part of an existing farm, you need to first leave the farm before you can initialize a file-based repository.

Note: Using the `leaveFarm` command on an instance stops all the Oracle Application Server components running on the instance.

Use the `leavefarm` command to leave the farm as follows:

```
% dcmctl leaveFarm
```

After you leave the farm, `whichFarm` returns the following:

```
% dcmctl whichFarm
Standalone OracleAS instance
```

There are restrictions on leaving a farm using `dcmctl leaveFarm`, including the following:

- If you attempt to use `dcmctl leaveFarm` on an Oracle Application Server Infrastructure system, `dcmctl` reports an error unless the Infrastructure system is the only Oracle Application Server instance that is part of the farm.
- Running the `dcmctl leaveFarm` command stops all the Oracle Application Server components running on the Oracle Application Server instance.
- You cannot use `leaveFarm` on an Oracle Application Server Infrastructure system that serves as the repository for any Oracle Application Server instances other

than itself. To run `leavefarm` on the Oracle Application Server Infrastructure, you must first go the Oracle Application Server instances and run `leaveFarm` on those instances.

Initializing the Repository Host Instance for a File-Based Repository

After selecting the Oracle Application Server instance to be the repository host for the file-based repository, do the following to create a farm and initialize the file-based repository on the repository host instance:

1. Issue the following command on the Oracle Application Server instance that is to be the repository host instance for the file-based repository:

```
dcmctl getRepositoryid
```

2. Using the repository ID that you obtain, issue the command:

```
dcmctl joinFarm -r <repositoryID>
```

Where *repositoryID* is the value returned from the previous step. The `dcmctl joinFarm` command sets up the repository host instance and initializes the farm managed using a file-based repository; Oracle Application Server stores the farm's configuration information in a file-based repository on the repository host instance.

Note: If you create a farm using the `dcmctl` command and you are using Application Server Console to view the changes, you need to restart Application Server Console for the changes to be shown. Use the `emctl start` and `stop` commands to restart Application Server Console.

Joining a Farm Managed Using a File-Based Repository

After selecting the repository host instance for the file-based repository and initializing the file-based repository, do the following to add additional application server instances to the farm:

1. Obtain the repository ID on the repository host instance. To do this, issue the following command:

```
dcmctl getRepositoryId
```

To obtain the repository ID for the repository host instance, you can issue the `getRepositoryid` command on any system which is part of the farm you want to join (that is, if another instance uses the same repository host instance, you can use the `dcmctl getRepositoryid` command on that system).

2. On the application server instance that you want to add to the farm, do the following:

```
dcmctl joinFarm -r <repositoryID>
```

Where the *repositoryID* you specify is the value returned in step 1.

Note: In this step, do not specify the repository ID of the instance you are on. Rather, use the repository ID for the farm's repository host instance, or use the repository ID of any instance that is already part of the farm you want to join (this value is the same for all Oracle Application Server instances that use the same file-based repository).

Also, if you are using Application Server Console to view the changes after executing the `dcmctl joinFarm` command, you need to restart Application Server Console for the changes to appear. Use the `emctl start` and `stop` commands to restart Application Server Console.

See Also: *Distributed Configuration Management Reference Guide* for information on `dcmctl` commands

Managing Instances in a Farm That Uses a File-Based Repository

This section covers the following topics:

- Managing Oracle Application Server Instances and Clusters With a File-Based Repository
- Availability Issues for OracleAS Clusters With a File-Based Repository
- Exporting and Importing Configuration Information With a File-Based Repository
- Moving an Instance Between Repositories
- Enabling SSL For Communication Between Instances That are Using a File-Based Repository

Managing Oracle Application Server Instances and Clusters With a File-Based Repository

Once a farm is set up that is managed using a file-based repository, you can use the Application Server Console or `dcmctl` commands to create and manage OracleAS Cluster within the farm, and you can configure standalone instances within the farm to join a cluster.

Note: For Oracle Application Server Clusters using a file-based repository, Oracle recommends a size of four or less OracleAS instances per cluster.

See Also:

- "Managing and Configuring OracleAS Clusters" on page 4-4
- *Distributed Configuration Management Reference Guide* for information on `dcmctl` commands

Availability Issues for OracleAS Clusters With a File-Based Repository

An important consideration for using OracleAS Clusters with a file-based repository is determining which Oracle Application Server instance is the repository host.

Consider the following when selecting the repository host for a file-based repository:

- When the repository host instance is temporarily unavailable, an OracleAS Cluster that uses a file-based repository is still able to run normally, but it cannot update any configuration information.
- Since the Oracle Application Server instance that is the repository host instance stores and manages the cluster related configuration information in its file system, the repository host instance should use mirrored or RAID disks. If the repository host instance uses disk mirroring, this improves the availability of the OracleAS Clusters.
- When the repository host instance is not available, read-only configuration operations are not affected on any Oracle Application Server instances that are running (the farm's cluster-wide configuration information is distributed and managed through local Oracle Application Server Java Object Cache),
- When the repository host instance is not available, operations that attempt to change configuration information in the file-based repository will generate an error. These operations must be delayed until the repository host instance is available, or until the repository host instance is relocated to another application server instance within the farm.

Exporting and Importing Configuration Information With a File-Based Repository

Oracle Application Server provides commands to save a file-based repository and prevent it from being permanently lost, when the repository host instance goes down or its file system is damaged. Using the `exportRepository` command you can save the entire file-based repository. After saving the configuration information using the `exportRepository` command, using `importRepository`, you can restore the saved configuration information to the repository host instance, or to a different instance in the farm.

To export the repository from the repository host instance, do the following:

```
dcmctl exportRepository -file <file_name>
```

To import a saved file-based repository, on the system that is to be the repository host instance for the farm, do the following:

```
dcmctl importRepository -file <file_name>
```

Note: Before running `dcmctl` with the `importRepository` option, stop all other DCM daemons in the instances of the same farm except for the current instance where you are running `dcmctl`. Use the following command at each instance:

```
ORACLE_HOME/opmn/bin/opmnctl stopproc ias-component=dcm-daemon
```

The `file_name` is a previously saved file that was created using the `dcmctl exportRepository` command. When the file-based repository is restored to a different Oracle Application Server instance, the instance where the `importRepository` command runs becomes the new repository host instance.

To specify that the Oracle Application Server instance that was the old repository host instance for a file-based repository is no longer the repository host instance, issue the following command:

```
dcmctl repositoryRelocated
```

Note: If you want the instance that was the old repository host instance to join the farm, be sure to run `repositoryRelocated` on the old repository host instance.

Moving an Instance Between Repositories

If you have an Oracle Application Server instance joined to a farm that uses a file-based repository, you can move that instance to another repository whether it is a file-based or database-based repository. The steps to move to another repository type involve the steps to leave a farm and join another farm.

When an OracleAS instance leaves a farm, it essentially becomes a standalone instance. The instance's DCM-managed configuration metadata in the repository is moved to the instance. Any archives for the instance are also deleted. However, connections to the Infrastructure database that may exist for other components (Oracle Application Server Single Sign-On, JAAS, and Oracle Internet Directory) are not removed.

To leave a farm, execute the following command at the OracleAS instance:

```
dcmctl leaveFarm
```

Note: After executing the `dcmctl leaveFarm` command, it is recommended that you create a new baseline archive for the instance that just left the farm. Refer to the *Distributed Configuration Management Reference Guide* for archiving instructions.

The following sections provide instructions to move an OracleAS instance from a file-based repository to other repositories:

- Moving to a Database-Based Repository
- Moving to Another File-Based Repository

Moving to a Database-Based Repository When moving an OracleAS instance from a file-based repository to a database-based repository, you must first disassociate the instance from its current repository by leaving the repository's farm. The instance then becomes a standalone instance at which point you can join it to the farm of a database-based repository. The following instructions tell you how to perform these tasks:

1. Determine if the instance is still part of a farm using the following command:

```
dcmctl whichFarm
```

2. If the command returns a farm name, the OracleAS instance is still part of a farm, and hence, still associated with an existing repository. Use the `dcmctl leaveFarm` command to bring the instance to a standalone state.
3. If the instance is joining the farm of the database-based repository for the first time, its configuration metadata is not in the repository. Use the Application Server Console to join the farm of the repository. Instructions to do this are in the *Oracle Application Server 10g Administrator's Guide*.
4. If the instance is joining a farm of the database-based repository, and the instance was a member of that farm earlier, use the following command to rejoin the farm:

```
dcmctl joinFarm
```

Moving to Another File-Based Repository To join the instance to another farm that is using a file-based repository, use the `dcmctl` command together with the file-based repository's ID. At the command line of the OracleAS instance:

1. Run the following command:

```
dcmctl whichFarm
```

2. If the command returns a farm name, the OracleAS instance is still part of a farm, and hence, still associated with an existing repository. Use the `dcmctl leaveFarm` command to bring the instance to a standalone state.
3. After ensuring that the instance is not part of a farm, run the following command at one of the instances that is joined to the farm of the repository that you want to join. This command gets the repository ID of the file-based repository. If you want to establish and join a new file-based repository using the host where the standalone instance is as the repository host, run the following command at the standalone instance.

```
dcmctl getRepositoryId
```

A repository identifier in the format "hostname:port" is returned.

4. Join the farm of the desired repository using the following command:

```
dcmctl joinFarm -r <repository_ID>
```

Note: if you are using Application Server Console to view the changes after executing the `dcmctl joinFarm` command, you need to restart Application Server Console for the changes to appear. Use the `emctl start` and `stop` commands to restart Application Server Console.

See Also: *Distributed Configuration Management Reference Guide*

Enabling SSL For Communication Between Instances That are Using a File-Based Repository

When instances in a farm use a file-based repository, you can configure DCM so that configuration information that is sent between instances uses SSL. This feature provides for the security of messages sent between all instances in the farm and prevents unauthorized instances from joining the farm.

This section describes the steps required to setup SSL and certificate-based security for instances that use a file-based repository. The overall steps are:

- Generating the Keystore
- Shutdown Oracle Application Server Processes on Each Instance
- Set Up the Keystore Information File on Each Instance in the Farm
- Enable SSL By Configuring `dcmCache.xml`
- Verify that Configuration Changes are Effected
- Start Each Instance in the Farm
- Adding a New Instance to a SSL-Enabled Farm

Generating the Keystore Use the JDK `keytool` command to generate a certificate and set up the keystore, as documented in:

<http://java.sun.com/j2se/1.4.1/docs/tooldocs/solaris/keytool.html>

If you have already generated the key pair and obtained the certificate for OC4J, then you can use the same keystore you previously obtained.

To use SSL certificate-based security, a Java keystore must be setup on each instance in the farm. This keystore may be the same as that used by other Java applications or it can be unique for DCM file-based repository configuration. Note the path to each keystore location for each instance in the farm.

Shutdown Oracle Application Server Processes on Each Instance At each instance of the farm, execute the following commands to shut down Oracle Application Server processes:

in UNIX:

```
$ORACLE_HOME/bin/emctl stop iasconsole
$ORACLE_HOME/dcm/bin/dcmctl stopproc
```

in Windows:

```
%ORACLE_HOME%\bin\emctl stop iasconsole
%ORACLE_HOME%\dcm\bin\dcmctl shutdown
```

Set Up the Keystore Information File on Each Instance in the Farm After obtaining the keystore and certificate information, on each Oracle Application Server instance in the farm, you need to use the `dcmctl configRepositorySSL` command to create the file that holds keystore information.

Important: The keystore information file must be set up for the repository host instance of the file-based repository before any other instance in the farm. To find the repository host and host instance, execute the following:

```
dcmctl getrepositoryid
```

To set up the keystore information file, execute the following instructions beginning with the repository host instance of the file-based repository (after that, the instructions can be performed in no particular sequence for the remaining instances):

1. Copy the keystore file that you generated in the first step, "Generating the Keystore," to a location in the local host.
2. Use the `configRepositorySSL` as follows on each instance to create the keystore information file:

```
dcmctl configRepositorySSL -keystore <path_to_keystore> -storepass <password>
```

The generated file, `.ssl.conf`, is stored in `<ORACLE_HOME>/dcm/config`.

Enable SSL By Configuring `dcmCache.xml` Modify the `dcmCache.xml` cache configuration `<useSSL>` attribute as shown in Table 4-3 to enable or disable the use of SSL.

Optionally, you can specify the location of the file that was generated using `configRepositorySSL` by modifying the value of the `<sslConfigFile>` element. If you modify this value, you need to copy the `.ssl.conf` file that

configRepositorySSL generated to the new file that you specify using <sslConfigFile>.

The dcmCache.xml file is in \$ORACLE_HOME/dcm/config directory in Unix, and in %ORACLE_HOME%\dcm\config directory in Windows.

Table 4–3 Elements for Enabling SSL in a Farm Using a File-Based Repository

Element	Description
<useSSL> true false </useSSL>	Set to true to use SSL or to false to disable the use of SSL by the DCM file-based repository communications mechanism. The default value is false. Valid values: true, false
<sslConfigFile> sslfile </sslConfigFile>	Specifies the name, sslfile for the SSL configuration file. The default value is .ssl.conf. For most installations, there should be no need to change the default value for this element.

Verify that Configuration Changes are Effected Ensure that the configuration changes are effected by executing the following command on each instance in the farm beginning with the repository host instance:

```
dcmctl getstate
```

The synchronization state of the local instance with the file-based repository is shown.

Start Each Instance in the Farm After the security configuration is consistent across all the instances in the farm, restart each instance, beginning with the repository host instance, using the following command:

in UNIX:

```
$ORACLE_HOME/opmn/bin/opmnctl startall  
$ORACLE_HOME/bin/emctl start iasconsole
```

in Windows:

```
%ORACLE_HOME%\opmn\bin\opmnctl startall  
%ORACLE_HOME%\bin\emctl start iasconsole
```

Adding a New Instance to a SSL-Enabled Farm You can add a standalone instance to a farm that is using SSL. On the standalone machine:

1. Copy the keystore file that you generated in the first step, "Generating the Keystore," to a location in the local host.
2. Use the configRepositorySSL as follows on each instance to create the keystore information file:

```
dcmctl configRepositorySSL -keystore <path_to_keystore> -storepass <password>
```

The generated file, .ssl.conf, is stored in <ORACLE_HOME>/dcm/config.

3. Follow the instructions in the section Joining a Farm Managed Using a File-Based Repository on page 4-11 to join the instance to the farm.

OC4J Configuration with an OracleAS Cluster

This section describes OC4J configuration for OC4J Instances and processes that are part of OracleAS Clusters that are managed using repositories.

This section covers the following:

- Overview of OracleAS Cluster Configuration for OC4J Instances
- Cluster-Wide Configuration Changes and Modifying OC4J Instances
- Configuring OC4J Instance-Specific Parameters

See Also: *Oracle Application Server Containers for J2EE User's Guide* for detailed information on configuring OC4J Instances

Overview of OracleAS Cluster Configuration for OC4J Instances

After application server instances join OracleAS Clusters, the application server instances, and the OC4J instances that run on the application server instances have the following properties:

- Each application server instance has the same cluster-wide configuration. When you use Application Server Console or `dcmctl` to modify any cluster-wide OC4J parameters, the modifications are propagated to all application server instances in the cluster. To make cluster-wide OC4J configuration changes you need to change the configuration parameters on a single application server instance; the Oracle Application Server distributed configuration management system then propagates the modifications to all the other application server instances within the cluster.
- When you modify any instance-specific parameters, on an OC4J instance that is part of a cluster, the change is not propagated across the cluster. Changes to instance-specific parameters are only applicable to the specific application server instance where the change is made. Since different hosts running application server instances in the cluster could each have different capabilities, such as total system memory, it may be appropriate for the OC4J processes within an OC4J instance to run with different configuration options.

Table 4–4 provides a summary of the OC4J instance-specific parameters. Other OC4J parameters are cluster-wide parameters and are replicated across OracleAS Clusters.

Table 4–4 OC4J Instance-Specific Parameters Summary for OracleAS Clusters that are managed using repositories

OC4J Parameter	Description
islands definitions	<p>While you want to keep the names of islands consistent across the application server instances, the definition of the islands and the number of OC4J processes associated with each island is configured on each instance, and the Oracle Application Server configuration management system does not replicate the configuration across the cluster.</p> <p>Note: state is replicated in OC4J islands with the same name across application boundaries and across the cluster. So to assure high availability, with stateful applications, the OC4J island names must be the same in each OC4J instance across the cluster.</p>
number of OC4J processes	<p>While you want to keep the names of islands consistent across the application server instances, the definition of the islands and the number of OC4J processes associated with each island is configured on each instance, and the Oracle Application Server configuration management system does not replicate the configuration across the cluster.</p> <p>On different hosts you can tune the number of OC4J processes specified to run per island to match the host capabilities.</p>
port numbers	The RMI, JMS, and AJP port numbers can be different for each host.
command line options	The command line options you use can be different for each host.

Cluster-Wide Configuration Changes and Modifying OC4J Instances

This section covers the following topics:

- Creating or Deleting OC4J Instances on OracleAS Clusters
- Deploying Applications on OracleAS Clusters
- Configuring Web Application State Replication for OracleAS Clusters
- Configuring EJB Application State Replication for OracleAS Clusters
- Configuring Stateful Session Bean Replication for OracleAS Clusters

See Also: *Oracle Application Server Containers for J2EE User's Guide* for complete information OC4J configuration and application deployment

Creating or Deleting OC4J Instances on OracleAS Clusters

You can create a new OC4J instance on any application server instance within managed OracleAS Clusters and the OC4J instance will be propagated to all application server instances across the cluster.

To create an OC4J instance, do the following:

1. Navigate to any application server instance within the cluster.
2. Select the **Create OC4J Instance** button. This brings up the page that requests a name for the new instance. Provide a name in the field.
3. Click create.
4. The Oracle Application Server distributed configuration management system propagates the new OC4J instance across the cluster.

A new OC4J instance is created with the name you provided. This OC4J instance shows up on each application server instance page across the cluster, in the System Components section.

To delete an OC4J instance, select the radio button next to the OC4J instance you wish to delete, then click Delete. The Oracle Application Server Distributed Configuration Management system propagates the OC4J removal across the cluster.

Deploying Applications on OracleAS Clusters

Using OracleAS Clusters, when you deploy an application to one application server instance, the application is propagated to all application server instances across the cluster.

To deploy an application across a cluster, do the following:

1. Select the cluster you want to deploy the application to.
2. Select any application server instance from within the cluster.
3. Select an OC4J instance on the application server instance where you want to deploy the application.
4. Deploy the application to the OC4J instance using either Application Server Console or `dcmctl` commands.
5. The Oracle Application Server Distributed Configuration Management system then propagates the application across the cluster.

See Also: *Oracle Application Server Containers for J2EE User's Guide* for complete information on deploying applications to an OC4J instance.

Configuring Web Application State Replication for OracleAS Clusters

To assure that Oracle Application Server maintains, across OracleAS Clusters, the state of stateful Web applications you need to configure state replication for the Web applications.

To configure state replication for stateful Web applications, do the following:

1. Select the Administration link on the OC4J Home Page.
2. Select **Replication Properties** in the Instance Properties column.
3. Scroll down to the Web Applications section. Figure 4–5 shows this section.
4. Select the **Replicate session state** checkbox.

Optionally, you can provide the multicast host IP address and port number. If you do not provide the host and port for the multicast address, it defaults to host IP address 230.0.0.1 and port number 9127. The host IP address must be between 224.0.0.2 through 239.255.255.255. Do not use the same multicast address for both HTTP and EJB multicast addresses.

Note: When choosing a multicast address, ensure that the address does not collide with the addresses listed in

<http://www.iana.org/assignments/multicast-addresses>

Also, if the low order 23 bits of an address is the same as the local network control block, 224.0.0.0 – 224.0.0.255, then a collision may occur. To avoid this provide an address that does not have the same bits in the lower 23 bits of the address as the addresses in this range.

5. Add the `<distributed/>` tag to all `web.xml` files in all Web applications. If the Web application is serializable, you must add this tag to the `web.xml` file.

The following shows an example of this tag added to `web.xml`:

```
<web-app>
  <distributed/>
  <servlet>
    ...
  </servlet>
</web-app>
```

Figure 4–5 Web State Replication Configuration

See Also: *Oracle Application Server Containers for J2EE User's Guide*

Configuring EJB Application State Replication for OracleAS Clusters

To create an EJB cluster, you specify the OC4J instances that are to be involved in the cluster, configure each of them with the same multicast address, username, and password, and deploy the EJB, which is to be clustered, to each of the nodes in the cluster.

Unlike HTTP clustering, EJBs involved in a cluster cannot be sub-grouped in an island. Instead, all EJBs within the cluster are in one group. Also, only session beans are clustered.

The state of all beans is replicated at the end of every method call to all nodes in the cluster using a multicast topic. Each node included in the EJB cluster is configured to use the same multicast address.

The concepts for understanding how EJB object state is replicated within a cluster are described in the *Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide*.

To configure EJB replication, you must do the following:

1. Select the Administration link on the OC4J Home Page.
2. Select **Replication Properties** in the Instance Properties column.
3. In the EJB Applications section, select the **Replicate State** checkbox.

Figure 4–6 shows this section.

4. Provide the username and password, which is used to authenticate itself to other hosts in the cluster. If the username and password are different for other hosts in the cluster, they will fail to communicate. You can have multiple username and password combinations within a multicast address. Those with the same username/password combinations will be considered a unique cluster.

Optionally, you can provide the multicast host IP address and port number. If you do not provide the host and port for the multicast address, it defaults to host IP

address 230.0.0.1 and port number 9127. The host IP address must be between 224.0.0.2 through 239.255.255.255. Do not use the same multicast address for both HTTP and EJB multicast addresses.

Note: When choosing a multicast address, ensure that the address does not collide with the addresses listed in

<http://www.iana.org/assignments/multicast-addresses>

Also, if the low order 23 bits of an address is the same as the local network control block, 224.0.0.0 – 224.0.0.255, then a collision may occur. To avoid this provide an address that does not have the same bits in the lower 23 bits of the address as the addresses in this range.

5. Configure the type of EJB replication within the `orion-ejb-jar.xml` file within the JAR file. See "Configuring Stateful Session Bean Replication for OracleAS Clusters" on page 4-22 for full details. You can configure these within the `orion-ejb-jar.xml` file before deployment or add this through the Application Server Console screens after deployment. To add this after deployment, drill down to the JAR file from the application page.

Figure 4–6 EJB State Replication Configuration

Configuring Stateful Session Bean Replication for OracleAS Clusters

For stateful session beans, you may have you modify the `orion-ejb-jar.xml` file to add the state replication configuration. Since you configure the replication type for the stateful session bean within the bean deployment descriptor, each bean can use a different type of replication.

Stateful session beans require state to be replicated among nodes. In fact, stateful session beans must send all their state between the nodes, which can have a noticeable effect on performance. Thus, the following replication modes are available to you to decide on how to manage the performance cost of replication:

End of Call Replication The state of the stateful session bean is replicated to all nodes in the cluster, with the same multicast address, at the end of each EJB method call. If a node loses power, then the state has already been replicated.

To use end of call replication, set the `replication` attribute of the `<session-deployment>` tag in the `orion-ejb-jar.xml` file to "endOfCall".

For example,

```
<session-deployment replication="EndOfCall" .../>
```

JVM Termination Replication The state of the stateful session bean is replicated to only one other node in the cluster, with the same multicast address, when the JVM is terminating. This is the most performant option, because the state is replicated only once. However, it is not very reliable for the following reasons:

- The state is not replicated if the power is shut off unexpectedly. The JVM termination replication mode does not guarantee state replication in the case of lost power.
- The state of the bean exists only on a single node at any time; the depth of failure is equal to one node.

To use JVM termination replication, set the `replication` attribute of the `<session-deployment>` tag in the `orion-ejb-jar.xml` file to "VMTermination".

For example,

```
<session-deployment replication="VMTermination" .../>
```

Configuring OC4J Instance-Specific Parameters

This section covers the instance-specific parameters that are not replicated across OracleAS Clusters that are managed using repositories.

This section covers the following:

- Configuring OC4J Islands and OC4J Processes
- Configuring Port Numbers and Command Line Options

See Also: *Oracle Application Server Containers for J2EE User's Guide* for complete information OC4J configuration and application deployment

Configuring OC4J Islands and OC4J Processes

To provide a redundant environment and to support high availability using OracleAS Clusters, you need to configure multiple OC4J processes within each OC4J instance.

Using OracleAS Clusters, state is replicated in OC4J islands with the same name within OC4J instances with the same name across the cluster. To assure high availability, with stateful applications, OC4J island names within an OC4J instance must be the same in corresponding OC4J instances across the cluster. It is the administrator's responsibility to make sure that island names match where session state replication is needed in a cluster.

The number of OC4J processes on an OC4J instance within a cluster is an instance-specific parameter since different hosts running application server instances in the cluster could each have different capabilities, such as total system memory. Thus, it could be appropriate for cluster to contain application server instances that each run different numbers of OC4J processes within an OC4J instance.

To modify OC4J islands and the number of processes each OC4J island contains, do the following:

1. Select the Administration link on the OC4J Home Page of the application server instance of interest in the cluster.
2. Select **Server Properties** in the Instance Properties area.

3. Scroll down to the Multiple VM Configuration section. This section defines the islands and the number of OC4J processes that should be started on this application server instance in each island.

Figure 4–7 displays the Multiple VM Configuration Islands section.

Figure 4–7 OC4J instance Island and Number of Processes Configuration

Multiple VM Configuration
 TIP If OC4J is running, newly added islands and associated processes will be automatically started.

Islands

Select	Island ID	Number of Processes
<input checked="" type="radio"/>	default_island	2
<input type="radio"/>	island2	3

Buttons: Add Another Row, Remove

Related Links: [Virtual Machine Metrics](#)

4. Create any islands for this OC4J instance within the cluster by clicking **Add Another Row**. You can supply a name for each island within the Island ID field. You can designate how many OC4J processes should be started within each island by the number configured in the Number of Processes field.

Configuring Port Numbers and Command Line Options

Figure 4–8 shows the section where you can modify these ports and set command line options.


To modify OC4J ports or the command line options, do the following:

1. Select the Administration link on the OC4J Home Page of the application server instance of interest in the cluster.
2. Select **Server Properties** in the Instance Properties area.
3. Scroll down to the Multiple VM Configuration section. This section defines the ports and the command line options for OC4J and for the JVM that runs OC4J processes.

Figure 4–8 shows the Ports and Command line options areas on the Server Properties page.

Figure 4–8 OC4J Ports and Command Line Options Configuration

Multiple VM Configuration

 **TIP** If OC4J is running, newly added islands and associated processes will be automatically started.

Islands

Island ID	Number of Processes	Related Links
default_island	1	Virtual Machine Metrics

[Add Another Row](#)

Ports

RMI Ports:

JMS Ports:

AJP Ports:

RMI-IIOP Ports

IIOP Ports:

IIOP SSL (Server only):

IIOP SSL (Server and Client):

Command Line Options

Java Executable:

OC4J Options:

Java Options:

Oracle HTTP Server Configuration with OracleAS Clusters

This section describes Oracle HTTP Server configuration for OracleAS Clusters that are managed using repositories.

This section covers the following:

- `mod_oc4j` Load Balancing With OracleAS Clusters
- Configuring Oracle HTTP Server Instance-Specific Parameters

`mod_oc4j` Load Balancing With OracleAS Clusters

This section covers the following:

- Load Balancing Overview
- Setting Load Balancing Options

Load Balancing Overview

Using OracleAS Clusters, the Oracle HTTP Server module `mod_oc4j` load balances requests to OC4J processes. The Oracle HTTP Server, using `mod_oc4j` configuration options, supports different load balancing policies. By providing configurable load balancing policies, OracleAS Clusters can provide performance benefits along with failover and high availability for different types of systems, depending on the network topology and host machine capabilities.

By default, `mod_oc4j` uses weights to select a node to forward a request to. Each node has a default weight of 1 unless specified otherwise. A node's weight is taken as a ratio compared to the weights of the other available nodes to define the number of requests the node should service compared to the other nodes in the cluster. Once a node is selected to service a particular request, by default, `mod_oc4j` uses the `roundrobin` policy to select OC4J processes on the node. If an incoming request belongs to an established session, the request is forwarded to the same node and the same OC4J process that started the session.

The OC4J load balancing policies do not take into account the number of OC4J processes running on a node when calculating which node to send a request to. Node selection is based on the configured weight for the node, and its availability. The number of OC4J processes to run is configured using Application Server Console.

See Also: "Configuring OC4J Instance-Specific Parameters" on page 4-23

Setting Load Balancing Options

To modify the `mod_oc4j` load balancing policy, Administrators use the `Oc4jSelectMethod` and `Oc4jRoutingWeight` configuration directives in the `mod_oc4j.conf` file.

To configure the `mod_oc4j.conf` file, using Application Server Console, select the `HTTP_Server` component in an application server instance. Then, select the Administration link and select the Advanced Server Properties link. On the Advanced Server Properties page, select the `mod_oc4j.conf` link. On the Edit `mod_oc4j.conf` page, within the `<IfModule mod_oc4j.c>` section, modify `Oc4jSelectMethod` and `Oc4jRoutingWeight` to select the desired load balancing option.

If you do not use Application Server Console, then edit `mod_oc4j.conf` and use the `dcmctl` command to propagate the changes to other `mod_oc4j.conf` files across the OracleAS Clusters as follows:

```
% dcmctl updateconfig -ct ohs
% opmnctl @cluster:<cluster_name> restartproc ias-component=HTTP_Server
    process-type=HTTP_Server
```

The `opmnctl restartproc` command is required to restart all the Oracle HTTP Server instances in the OracleAS Clusters for the changes to take effect.

See Also:

- "Configuring OC4J Instance-Specific Parameters" on page 4-23
- *Oracle HTTP Server Administrator's Guide* for information on using `mod_oc4j` load balancing directives
- *Oracle Application Server 10g Performance Guide*

Configuring Oracle HTTP Server Instance-Specific Parameters

The following are instance-specific parameters used by Oracle HTTP Server.

- `ApacheVirtualHost`
- `Listen`
- `OpmnHostPort`
- `Port`
- `User`
- `Group`
- `NameVirtualHost`
- `ServerName`
- `PerlBlob`

You can modify the HTTP Server ports and listening addresses on the Server Properties Page, which can be accessed from the HTTP Server Home Page. You can

modify the virtual host information by selecting a virtual host from the Virtual Hosts section on the HTTP Server Home Page.

Security – Configuring Single Sign-On

To enable Oracle Application Server Single Sign-On to work with an OracleAS Cluster, the Single Sign-On server needs to be aware of the entry point into the cluster, which is commonly the load balancing mechanism in front of the Oracle HTTP Servers. This mechanism could exist as Oracle Application Server Web Cache, a network load balancer appliance, or an Oracle HTTP Server installation.

In order to register an OracleAS Cluster's entry point with the Single Sign-On server, use the `SSORegistrar` tool, which can be executed through `ossoreg.jar`.

In order to participate in Single Sign-On functionality, all Oracle HTTP Server instances in a cluster must have an identical Single Sign-On registration.

- Each Oracle HTTP Server is registered with the same Single Sign-On server.
- Each Oracle HTTP Server redirects a success, logout, cancel, or home message to the public network load balancer. In a clustered environment, each Oracle HTTP Server should redirect message URLs to the network load balancer. Since the client cannot access an Oracle HTTP Server directly, the client interacts with the network load balancer.

As with all cluster-wide configuration information, the Single Sign-On configuration is propagated among all Oracle HTTP server instances in the cluster. However, the initial configuration is manually configured and propagated. On one of the application server instances, define the configuration with the `ossoreg.jar` tool. Then, DCM propagates the configuration to all other Oracle HTTP Servers in the cluster.

If you do not use a network load balancer, then the Single Sign-on configuration must originate with whatever you use as the incoming load balancer— Oracle Application Server Web Cache, Oracle HTTP Server, and so on.

To configure a cluster for Single Sign-On, execute the `ossoreg.jar` command against one of the application server instances in the cluster. This tool registers the Single Sign-On server and the redirect URLs with all Oracle HTTP Servers in the cluster.

Run the `ossoreg.jar` command with all of the options as follows, substituting information for the italicized portions of the parameter values.

The values are described fully in Table 4-5.

- Specify the host, port, and SID of the database used by the Single Sign-On server.
- Specify the host and port of the front-end load balancer in `mod_osso_url` parameter. This should be a HTTP or HTTPS URL depending on the site security policy regarding SSL access to OracleAS Single Sign-On protected resources.
- Specify the root user of the host that you are executing this tool on in the `-u` option.

```
$ORACLE_HOME/jdk/bin/java -jar $ORACLE_HOME/sso/lib/ossoreg.jar
-oracle_home_path <orcl_home_path>
-site_name <site_name>
-config_mod_osso TRUE
-mod_osso_url <URL>
-u <userid>
[-virtualhost <virtual_host_name>]
[-update_mode CREATE | DELETE | MODIFY]
[-config_file <config_file_path>]
[-admin_info <admin_info>]
```

[-admin_id <adminid>]

Table 4–5 SSORRegistrar Parameter Values

Parameter	Value
oracle_home_path <orcl_home_path>	Absolute path to the Oracle home of the application server instance, where you are invoking this tool.
site_name <site_name>	Name of the site typically, the effective host name and port of the partner application. For example, application.mydomain.com.
config_mod_osso TRUE	If set to TRUE, this parameter indicates that the application being registered is mod_osso. You must include config_mod_osso for osso.conf to be generated.
mod_osso_url <URL>	The effective URL of the partner application. This is the URL that is used to access the partner application. The value should be specified in this URL format: http://oracle_http_host.domain:port
u <userid>	The user name that will start the Oracle HTTP Server. In UNIX, this name is usually "root." On Windows NT/2000, it is SYSTEM. The parameter u is mandatory.
virtualhost <virtual_host_name>	Optional. Use this parameter only if registering an Oracle HTTP virtual host with the OracleAS Single Sign-On server. If you create a virtual host, be sure, in the httpd.conf file, to fill in the following directive for each protected URL: <VirtualHost host_name> OssoConfigFile \$ORACLE_HOME/Apache/Apache/conf/osso/host_name/osso.conf OssoIpCheck off #<Location /your_protected_url> # AuthType basic # Require valid-user #</Location> #Other configuration information for the virtual host </VirtualHost> The commented lines must be uncommented before the application is deployed.
update_mode CREATE DELETE MODIFY	Optional. Creates, deletes, or modifies the partner registration record. CREATE, the default, generates a new record. DELETE removes the existing record. MODIFY deletes the existing record and then creates a new one.
config_file <config_file_path>	Optional. Location of the osso.conf file for the virtual host if one is being configured. It may, for example, be \$ORACLE_HOME/Apache/Apache/conf/osso/virtual_host_name/osso.conf. Note that the osso.conf for the non-virtual host is located at \$ORACLE_HOME/Apache/Apache/conf/osso.
admin_id <name>	(Optional) User name of the mod_osso administrator. This shows up in the Single Sign-On tool as contact information.
admin_info <text>	(Optional) Additional information about the mod_osso administrator, such as e-mail address. This shows up in the Single Sign-On tool as contact information.

The `SSORegistrar` tool establishes all information necessary to facilitate secure communication between the Oracle HTTP Servers in the cluster and the Single Sign-On server.

When using Single Sign-On with the Oracle HTTP Servers in the cluster, the `KeepAlive` directive must be set to `OFF` since the Oracle HTTP Servers are behind a network load balancer. If the `KeepAlive` directive is set to `ON`, then the network load balancer maintains state with the Oracle HTTP Server for the same connection, which results in an HTTP 503 error. Modify the `KeepAlive` directive in the Oracle HTTP Server configuration. This directive is located in the `httpd.conf` file of the Oracle HTTP Server.

See Also: *Oracle Application Server Single Sign-On Administrator's Guide*

Advanced Clustering Configuration

You can configure a cluster of OracleAS instances to provide only certain limited advantages of clustering.

This section describes how to configure these advanced types of clusters.

- Routing Between Instances in Same Farm
- Routing Between Instances Across Firewalls

Routing Between Instances in Same Farm

If you have more than a single OracleAS instance in a farm, you can configure one of the Oracle HTTP Servers to be the load balancer for all of the instances. This eliminates the need for all but one of the Oracle HTTP Servers in the OracleAS instances. When you configure a single Oracle HTTP Server as a load balancer, the Oracle HTTP Server must be configured to know about all the OC4J instances in the farm and route the incoming requests appropriately.

Configure the following:

1. Retrieve the OracleAS instance name and its components of all instances in the farm.

- a. Change to the DCM directory of each OracleAS instance in the farm.

```
cd ORACLE_HOME_Instance/dcm/bin
```

- a. Retrieve the OracleAS instance name and list all of its components.

```
dcmctl whichInstance
dcmctl listComponents
```

2. Update the `mod_oc4j.conf` configuration file with the OC4J instance information for each root context, which enables `mod_oc4j` to route to each deployed application.

- a. Change to the Apache directory of each OracleAS instance in the cluster.

```
cd ORACLE_HOME_Instance/Apache/Apache/conf
```

- b. Edit the `mod_oc4j.conf` to include mount points for the root context of each deployed application in the other OC4J instances in the cluster. Each `mod_`

oc4j configuration file contains mount points for each root context of the deployed applications to which it routes incoming requests.

To route to applications deployed in another instance, you must add a mount point for the other instances' application root context with the additional keyword of "instance://". The syntax for this keyword requires the OracleAS Instance name and the OC4J instance name.

Note: We suggest that you mount all clustered applications to the same root context to avoid multiple entries in each mod_oc4j configuration file involved in the cluster.

To route to applications deployed in another cluster, you must add a mount point for the application root context with the additional keyword of "cluster://". The syntax for this keyword requires the cluster name and the OC4J instance name.

Examples of routing to another instance, multiple instances, or another cluster are as follows:

```
Oc4jMount /myapp/* instance://Inst2:OC4J_Home
Oc4jMount /myapp1/* instance://Inst2:OC4J_Home, Inst3:OC4J_Home
Oc4jMount /myapp2/* cluster://Cluster1:OC4J_Home
```

- c. Inform DCM of the configuration changes and restart DCM.

```
dcmctl updateConfig
dcmctl restart
```

Once configuration for the cluster is complete, you must ensure that each OracleAS instance and OC4J instance has the same configuration. This type of cluster does not replicate configuration across all instances. You must manage the configuration manually.

You can configure for OC4J state replication through the Application Server Console in the same way as for managed clustering.

See Also: "Configuring OC4J Instance-Specific Parameters" on page 4-23

Routing Between Instances Across Firewalls

Firewalls protect a company's infrastructure by restricting illegal network traffic. Firewall configuration typically involves restricting the ports that are available to one side of the firewall. In addition, it can be set up to restrict the type of traffic that can pass through a particular port, such as HTTP. If a client attempts to connect to a restricted port or uses a protocol that is not allowed, then the client is disconnected immediately by the firewall. Firewalls can also be used within a company Intranet to restrict user access to specific servers.

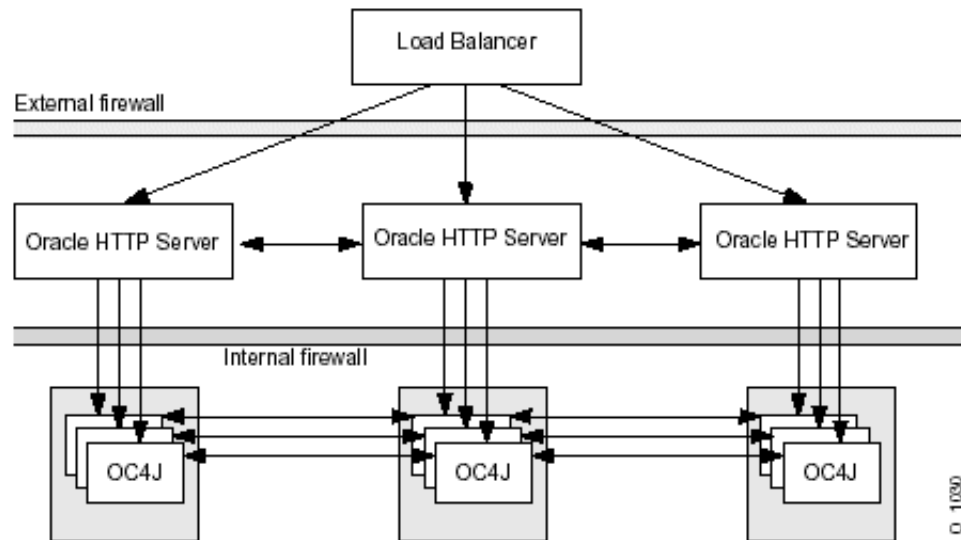
Some of the components of OracleAS can be deployed on different nodes, which can be separated by firewalls. Figure 4–9 demonstrates one recommended organization of OracleAS components between two firewalls:

- An external firewall protects the Oracle HTTP Servers from external misuses.

- The internal firewall protects the OC4J processes within an intranet in case the first firewall is penetrated.

All communication between the Oracle HTTP Servers and the OC4J processes behind the second firewall should use SSL encryption. Authorization should be provided using SSL client certificates.

Figure 4–9 Routing Between Oracle HTTP Servers and OC4J Processes Through Multiple Firewalls



However, the Oracle HTTP Server and OC4J processes communicate through several ports using DCM, OPMN, and `mod_oc4j` for this communication. This communication must continue, even if a firewall exists between them. You can continue the communication by exposing the OracleAS component ports through the firewall that are needed to communicate between the OC4J components. You can either manually open each port needed for this communication or you can use the OracleAS Port Tunnel, which opens a single port to handle all communication that normally occurs through several ports. These options are discussed in the following sections:

- Opening Intranet Communication through the OracleAS Port Tunnel
- Opening OracleAS Ports To Communicate Through Intranet

Opening Intranet Communication through the OracleAS Port Tunnel

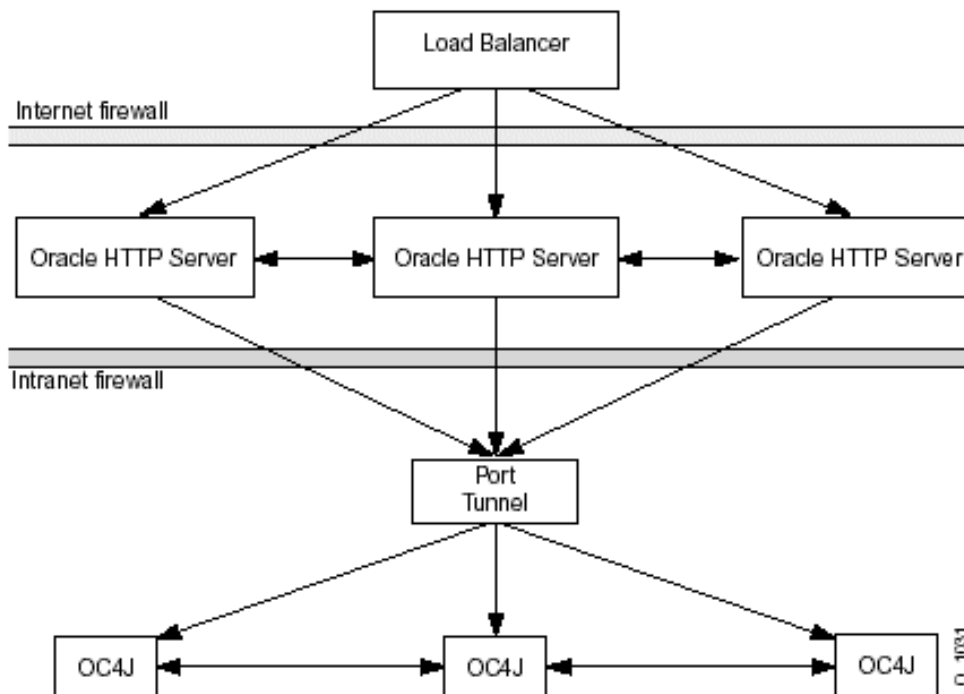
Instead of opening multiple ports on the intranet firewall, you can use the OracleAS Port Tunnel. The Port Tunnel is a process that facilitates the communication between Oracle HTTP Server and OC4J, including the communication for DCM, OPMN and `mod_oc4j`, using a single port exposed on the intranet firewall. Thus, you do not have to expose several ports for communication for a single OC4J process. Instead, the Port Tunnel exposes a single port and can handle all of the port requirements for several OC4J processes.

All communication between the Oracle HTTP Servers and the Port Tunnel is encrypted using SSL.

Figure 4–10 shows how three Oracle HTTP Servers communicate with three OC4J processes through the Port Tunnel. Only a single port is exposed on the intranet

firewall. The Oracle HTTP Servers exist on a single machine; the Port Tunnel and OC4J processes exist on a separate machine.

Figure 4–10 OracleAS Port Tunnel



However, if you have only a single process managing the communication between the Oracle HTTP Servers and the OC4J processes, you cannot guarantee high availability or failover. You can add multiple Port Tunnel processes, each listening on their own port, to manage the availability and the failover. We recommend that you use two Port Tunnel processes for each machine. You want to minimize the number of ports exposed on the intranet for security, but you also should provide for failover and availability.

Once the Port Tunnel processes are configured and initialized, then the Oracle HTTP Servers automatically balance the load among the port tunnel processes, just as they would among OC4J processes.

While you are risking exposure of a single port for each Port Tunnel process, the number of ports exposed using the Port Tunnel are much less than if you expose all of the ports needed for straight communication between Oracle HTTP Server and OC4J processes, as you can see in "Opening OracleAS Ports To Communicate Through Intranet" on page 4-32.

All of the details for configuring and initializing Port Tunnel processes are documented in the HTTP Security chapter in the *Oracle Application Server 10g Security Guide*.

Opening OracleAS Ports To Communicate Through Intranet

You can route between Oracle HTTP Servers and OC4J processes that are located on either side of an intranet firewall by exposing each of the OracleAS component ports through the firewall that are needed to communicate between the OC4J components.

The ports that should be opened on the firewall depend on the services that you are using. Table 4–6 describes the ports that you should open for each service.

Table 4–6 Ports that Each Service Uses

Service Name	Description	Configuration XML File
Oracle HTTP Server	Any incoming requests uses HTTP or HTTPS.	The ports listed in the listen directives in the <code>httpd.conf</code> configuration file.
OPMN	OPMN uses HTTP ports to communicate between other OPMN processes in a OracleAS Cluster. OPMN communication is bidirectional, so the ports for all OPMN processes must be opened to each other and to the OC4J processes.	The <code>ons.conf</code> configuration file, which is modified either through hand editing or through the Application Server Console GUI. You can also find out these port numbers by executing <code>dcmdctl getOPMNPort</code> .
DCM	DCM uses JDBC to talk to the back-end Oracle-based repository. If it is not desirable to open up a port to the database, then you can use a file-based repository, instead of a database repository. See "Routing Between Instances in Same Farm" on page 4-29 for directions on setting up a file-based repository.	The JDBC default port number is 1521. The JDBC database port number is defined in the <code>listener.ora</code> file in the Net8 database configuration.
	DCM bootstraps with information from the Oracle Internet Directory over an LDAP port.	The default ports are 389 for LDAP and 636 for LDAP over SSL. If these are taken, then the next in the range are selected; the range is 4031-4040. You can change the port numbers in the <code>ORACLE_HOME/config/ias.properties</code> file.
mod_oc4j module	Communicates with each OC4J process over an AJP port. The port range default is 3001-3100.	Defined in the <code><port></code> element either specifically or within a range in the <code>opmn.xml</code> file. We recommend that you specify exactly the number of ports needed for the number of OC4J processes used.
RMI or JMS	You may use RMI or JMS to communicate with OC4J. The default for the port range for RMI ports is 3101 to 3200. The default for the port range for JMS ports is 3201 to 3300.	Defined in the <code><port></code> element either specifically or within a range in the <code>opmn.xml</code> file. We recommend that you specify exactly the number of ports needed for the number of OC4J processes used.
Infrastructure	The Infrastructure database only executes on port 1521.	N/A
Portal	Uses the same AJP port range as configured for OC4J processes.	Defined in the <code><port></code> element either specifically or within a range in the <code>opmn.xml</code> file.

At installation time, the Oracle Installer picks available ports and assigns them to relevant processes. You can see the assigned ports for all components by selecting Ports in the default home page using Application Server Console.

Note: Some port numbers have multiple dependencies. If you change a port number, you may be required to alter other components. See the "Managing Ports" chapter in the *Oracle Application Server 10g Administrator's Guide* for a full discussion on how to manage your port numbers.

You can view all of the ports that are in use through Application Server Console. From the OracleAS Home Instance, select Ports at the top left corner of the page. Figure 4–11 shows all of the ports in use for this OracleAS instance, including all Oracle HTTP Server and OC4J instances. See the "Managing Ports" chapter in the *Oracle Application Server 10g Administrator's Guide* for more information on managing ports.

Figure 4–11 Ports used in OracleAS

Home		Ports		
Filter By:		All Ports	Go	
Component	Type of Port	Port Number Range	Port In Use	Configur
Oracle 9iAS Containers for J2EE - home	JMS server port	0-65535		Edit
Oracle 9iAS Containers for J2EE - OC4J_EM	JMS server port	0-65535	3201	Edit
Oracle 9iAS Containers for J2EE - home	JMS Queue Connection Factory port	0-65535	9127	
Oracle 9iAS Containers for J2EE - home	JMS Queue Connection Factory port	0-65535	9127	
Oracle 9iAS Containers for J2EE - OC4J_EM	JMS Queue Connection Factory port	0-65535	9127	
Oracle 9iAS Containers for J2EE - OC4J_EM	JMS Queue Connection Factory port	0-65535	9127	
Oracle 9iAS Containers for J2EE - home	RMI server port	0-65535		Edit
Oracle 9iAS Containers for J2EE - OC4J_EM	RMI server port	0-65535	3101	Edit
Oracle 9iAS Containers for J2EE - home	Web site port	0-65535		Edit
Oracle 9iAS Containers for J2EE - OC4J_EM	Web site port	0-65535	3301	Edit
Oracle HTTP Server	The main server port	7777-7877	7777	Edit
Oracle HTTP Server	The VirtualHost listen port	77777-88888		Edit
Oracle Enterprise Manager	Console	1810-1830	1810	
Oracle Enterprise Manager	Agent	1810-1830	5125	

Home Ports

Managing Infrastructure High Availability

This section provides instructions to manage your Infrastructure's high availability environment. Instructions for operations such as stopping, starting, and recovering from scheduled and unplanned outages are provided. The two high availability solutions are discussed:

- Oracle Application Server Cold Failover Clusters
- Oracle Application Server Active Failover Cluster (UNIX)

Note: For details on installing for high availability, refer to the *Oracle Application Server 10g Installation Guide*.

Oracle Application Server Cold Failover Clusters

The instructions in this section detail the steps for starting and stopping the OracleAS Infrastructure in an OracleAS Cold Failover Cluster.

Starting Up

Use the following steps to start the Infrastructure in an OracleAS Cold Failover Cluster:

1. Set the `ORACLE_HOME` environment variable to the Infrastructure's Oracle home.
2. Set the `ORACLE_SID` environment variable to the metadata repository's system identifier.
3. Set the `PATH` environment variable to include the Infrastructure's `$ORACLE_HOME/bin` directory.

Important: Specify the path of the working Oracle home as the first entry in the `PATH` environment variable if there are several Oracle homes installed on the machine. Also, ensure that the full paths of the executables you use are specified.

4. Enable volume management software and mount the file system (if necessary).
5. Enable the virtual IP address.
6. Start the metadata repository listener.

```
$ORACLE_HOME/bin/lsnrctl start
```

7. Start the metadata repository.

8. Start OPMN and all OPMN-managed processes for each OracleAS instance locally.

If OPMN daemon is not running, start both OPMN daemon and OPMN-managed processes:

```
$ORACLE_HOME/opmn/bin/opmnctl startall
```

If OPMN daemon is running, start all OPMN-managed processes collectively:

```
$ORACLE_HOME/opmn/bin/opmnctl startproc
```

Alternatively, to individually start up OPMN-managed processes:

- a. Start Oracle HTTP Server:

```
$ORACLE_HOME/opmn/bin/opmnctl startproc ias-component=HTTP_Server
```

- b. Start Oracle Internet Directory:

```
$ORACLE_HOME/opmn/bin/opmnctl startproc ias-component=OID
```

- c. Start the Delegated Administration Services instance:

```
$ORACLE_HOME/opmn/bin/opmnctl startproc ias-component=OC4J instancename=OC4J_SECURITY
```

- d. Check the status of the OPMN-managed processes using the following command:

```
$ORACLE_HOME/opmn/bin/opmnctl status
```

9. Start the Application Server Console. Use one of the following commands:

```
$ORACLE_HOME/bin/emctl start iasconsole
```

Or:

```
$ORACLE_HOME/bin/emctl startifdown iasconsole
```

Stopping

Use the following steps to stop the OracleAS Infrastructure in an OracleAS Cold Failover Cluster:

1. Set the `ORACLE_HOME` environment variable to the Infrastructure's Oracle home.
2. Set the `ORACLE_SID` environment variable to the metadata repository's system identifier.
3. Stop OPMN and all OPMN-managed processes for each OracleAS instance locally.

To shutdown the OPMN daemon and all OPMN-managed processes:

```
$ORACLE_HOME/opmn/bin/opmnctl stopall
```

To shutdown all OPMN-managed processes but leave the OPMN daemon running:

```
$ORACLE_HOME/opmn/bin/opmnctl stopproc
```

Alternatively, to individually shutdown all OPMN-managed processes:

- a. Stop the Delegated Administration Services instance:

```
$ORACLE_HOME/opmn/bin/opmnctl stopproc ias-component=OC4J  
instancename=OC4J_SECURITY
```

- b. Stop Oracle Internet Directory:

```
$ORACLE_HOME/opmn/bin/opmnctl stopproc ias-component=OID
```
- c. Stop Oracle HTTP Server.

```
$ORACLE_HOME/opmn/bin/opmnctl stopproc ias-component=HTTP_Server
```
4. Stop the metadata repository
5. Stop the metadata repository listener.

```
$ORACLE_HOME/bin/lsnrctl stop
```
6. Stop the Application Server Console.

```
$ORACLE_HOME/bin/emctl stop iasconsole
```
7. Disable volume management software and unmount the file system (if necessary).
8. Disable the virtual IP address.

Oracle Application Server Active Failover Cluster (UNIX)

Note: Check OracleMetalink (<http://metalink.oracle.com>) for the most current certification status of this feature or consult your Oracle sales representative before deploying this feature in a production environment.

The instructions in this section detail the steps for starting and stopping the Infrastructure in the OracleAS Active Failover Cluster high availability solution.

Starting Up

For an OracleAS Active Failover Cluster-enabled Infrastructure, each node in the cluster is functionally equivalent to the other nodes. All nodes access a common repository. The database instance and the individual OracleAS processes need to be started on each node of the cluster. At any given time, the load balancer should be configured to direct traffic to only the active nodes. The order of starting up Infrastructure instances on all nodes is:

1. On each node, start the global services daemon:

```
$ORACLE_HOME/bin/gsd
```

2. Start the database instances and listeners on all nodes with the following command (can be run from any node in the cluster):

```
$ORACLE_HOME/bin/srvctl start -p <database_name>
```

The global services daemon on each node ensures that the local database processes on each node are started.

3. Start OPMN and all OPMN-managed processes.

If OPMN daemon is not running, start both OPMN daemon and OPMN-managed processes (following command need only be run once):

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2> startall
```

Note: For more efficient startup of the OracleAS Active Failover Cluster nodes, you can configure each node's operating system to start up the OPMN daemon whenever the node starts up. The procedures for performing this task are specific to each operating system. For example, in UNIX, the rc scripts can be configured by the system administrator for this purpose.

If OPMN daemon is running, you can start all OPMN-managed processes on all nodes (following command need only be run once):

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2> startproc
```

For example, assuming there are two nodes in the cluster:

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:infra_node1:infra_node2 startproc
```

Alternatively, to individually start up OPMN-managed processes on all nodes (following commands need only be run once):

a. Start Oracle HTTP Server:

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2> startproc
  ias-component=HTTP_Server
```

b. Start Oracle Internet Directory.

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2> startproc
  ias-component=OID
```

c. Start the Delegated Administration Services instance:

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2> startproc
  ias-component=OC4J instancename=OC4J_SECURITY
```

4. Configure the load balancer and enable traffic to the current node.
5. Check the status of OPMN-managed processes on all nodes using the following command:

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2> status
```

6. Start the Application Server Console. Run one of the following commands on each node in the cluster:

```
$ORACLE_HOME/bin/emctl start iasconsole
```

Or:

```
$ORACLE_HOME/bin/emctl startifdown iasconsole
```

Shutting Down

The OracleAS Active Failover Cluster-enabled Infrastructure provides better availability since individual Infrastructure instances can be shut down while others continue to be available. The order of shutting down an instance is:

1. Configure the load balancer and disable traffic to the current node.
2. Stop all OPMN and OPMN-managed processes.

To stop all OPMN-managed processes but leave the OPMN daemon running, run the following command once:

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2> stopproc
```

To individually stop OPMN-managed processes on all cluster nodes, run the following commands once:

- a. Stop the Delegated Administration Services instance:

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2> stopproc
  ias-component=OC4J instancename=OC4J_SECURITY
```

- b. Stop Oracle Internet Directory:

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2> stopproc
  ias-component=OID
```

- c. Stop Oracle HTTP Server:

```
$ORACLE_HOME/opmn/bin/opmnctl
  @instance:<instancename_on_node1>:<instancename_on_node2>
  stopproc ias-component=HTTP_Server
```

3. Stop the database instances and listeners on all nodes with the following command (can be run from any node in the cluster):

```
$ORACLE_HOME/bin/srvctl stop -p <database_name>
```

The global services daemon on each node ensures that the local database processes on each node are stopped.

4. Stop the Application Server Console. Run the following command on each node in the cluster:

```
$ORACLE_HOME/bin/emctl stop iasconsole
```

Monitoring

Monitoring the OracleAS Active Failover Cluster-enabled Infrastructure is similar to monitoring any regular Infrastructure deployment. The only special consideration is monitoring the load balancer and ensuring that it is directing traffic to the active nodes. Please contact your load balancer vendor on monitoring the availability of the load balancer as well as the sanity of its configuration.

Failing Over During an Outage

The OracleAS Active Failover Cluster-enabled Infrastructure provides continued availability under many scheduled and unplanned outages. The outages handled automatically by this solution are listed in Table 5–1 below.

Table 5–1 Outages handled automatically by OracleAS Active Failover Cluster solution

Outage Type	Outages
Scheduled	Node hardware and operating system maintenance
	Database instance maintenance
	Infrastructure software maintenance
	Fault tolerant load balancer maintenance
Unplanned	Node failure
	Database instance failure
	Infrastructure process failure
	Fault tolerant load balancer partial failure

For the outages in Table 5–2, there may be a small downtime. Having a disaster recovery site can mitigate but not eliminate this downtime. A standby site can be activated while the production site experiences an outage. Refer to the section "Oracle Application Server 10g Disaster Recovery Solution" on page 6-2 for more information.

Table 5–2 Outages not handled automatically by OracleAS Active Failover Cluster solution

Outage Type	Outages
Scheduled	Cluster maintenance
	Database maintenance
Unplanned	Cluster failure
	Data error
	User error
	Fault tolerant load balancer complete failure

System behavior under these outages is as follows:

- When a node fails:
 - All processes in the node are unavailable.
 - Middle tier and Infrastructure processes from other nodes connected to the database instance on the failed node lose their database session.
 - A surviving instance on another node begins instance recovery. Requests that were directed to the failed instance experience a brief interruption in service. If they are disconnected, they can retry the connections again until successful.
 - The processes connected to the active instance performing the recovery experience uninterrupted service. These processes may experience a lag.
 - If only one node is available to service requests and it fails, connections that are retrying will only succeed when at least one instance becomes available.
- When a database instance fails, the following need to be performed:

- The load balancer should be notified of the failure. It should stop directing non Oracle Net traffic to the node with the failed instance.
- All non-database processes on the node with the failed instance should be brought down. This includes Oracle HTTP Server, OPMN, Application Server Console, and Oracle Internet Directory processes.
- Middle tier and Infrastructure processes from other nodes connected to the failed instance lose their database session.
- A surviving instance on another node begins instance recovery.
- Middle tier requests that were directed to the failed instance experience a brief interruption in service. If they are disconnected, they can retry the connections again until successful.
- The processes connected to the active instance performing the recovery experience uninterrupted service.
- All new middle tier requests are directed to the surviving instance.

Restoring Resiliency After an Outage

Restoration of resiliency post-outage involves the addition of an Oracle Application Server 10g instance to the current set of active Oracle Application Server 10g instances. The primary steps involved are:

1. Fix the problem that caused the outage.
2. Startup an Oracle database instance on the node.
3. Startup Oracle Application Server 10g Infrastructure processes on the node. Refer to *Oracle Application Server 10g Administrator's Guide* on how to start and stop the Infrastructure.
4. Configure the load balancer to direct traffic to the currently started node.

Synchronizing Configuration Files Using the Oracle Application Server Active Failover Cluster Runtime Control Utility (afctl)

Each node of an OracleAS Active Failover Cluster has in its file system configuration files that are part of the Infrastructure but are not stored in the OracleAS Metadata Repository. These files are likely to change as administration operations such as the following are performed on each node:

- Manual changes made to these configuration files
- Application Server Console-based changes to these configuration files
- Association of a middle tier instance with the Infrastructure

A primary requirement for an OracleAS Active Failover Cluster deployment is that all nodes in the cluster are configured similarly. The configuration files should be similar, if not, identical.

In order to maintain a consistent Infrastructure across all nodes in the OracleAS Active Failover Cluster, a command line utility is provided to synchronize these configuration files across the nodes. This utility is called *Oracle Application Server Active Failover Cluster Runtime Control Utility* and can be invoked using the command `afctl`. Synchronization of files using this utility should be performed at least everytime an administration change is made to Oracle Application Server.

Setting Up afcctl

This section describes how to download and install the afcctl utility and perform initial configuration. The overall steps are:

- Obtain the afcctl Utility
- Install the afcctl Utility

Obtain the afcctl Utility

The afcctl utility is available with the utility CD that comes with your Oracle Application Server 10g product. The file containing the utility is `<mount-point>/utilities/ha/afcctl.zip` (where `<mount-point>` is the mount point of the CD-ROM drive. This file is installed along with the Oracle Application Server 10g Backup and Recovery Tool. Hence, you need to install the latter tool first. Refer to *Oracle Application Server 10g Administrator's Guide* for instructions on installing this tool.

After the Oracle Application Server 10g Backup and Recovery Tool has been installed on a node, create the directory `<ORACLE_HOME>/afcctl/` on that node, and copy `afcctl.zip` to the new directory. Do this for every node in the OracleAS Active Failover Cluster.

Note: You can also create a directory to store `afcctl.zip` outside of `ORACLE_HOME`. The instructions in this section assumes you have `afcctl` in `<ORACLE_HOME>/afcctl/`.

Install the afcctl Utility

1. Change to the `<ORACLE_HOME>/afcctl/` directory and unzip `afcctl.zip`. It should contain the following files:

```
afcctl
afcctl.pl
afcctl.jar
afcctl_exclude.inp
README
```

2. In UNIX, run the following command to enable execute permissions:

```
> chmod 755 afcctl
```
3. The afcctl utility relies on the Oracle Application Server 10g Backup & Recovery Tool being available and installed. Make sure that the latter tool has been installed and the `.inp` files of the tool are accessible by the afcctl utility.

Using afcctl

Run the afcctl utility on any node in the OracleAS Active Failover Cluster to perform the following tasks:

- Setting the Default Baseline Timestamp
- Synchronizing Files From a Node to Other Nodes in an OracleAS Active Failover Cluster
- Listing Modified Files on a Node Since the Last Synchronization
- Excluding Specific Configuration Files from Synchronization

Note: Ensure that ORACLE_HOME is set before running afcctl.

Setting the Default Baseline Timestamp

Immediately after installation of OracleAS and afcctl, you should set the default configuration timestamp across the OracleAS Active Failover Cluster. This baseline timestamp marks the default configuration after installation.

After this baseline is set, the next synchronization performed using afcctl with the sync option synchronizes only those configuration files that have changed since the baseline and the time the afcctl sync command is run.

To create a timestamp baseline, use the following command:

```
afcctl createbase -p <dbname>|-r <host1>,<host2>,...,<hostN> [-c <cp_exec>]
```

where:

<dbname> is the name of the Infrastructure database

<host1>, <host2>, . . . , <hostN> is a comma separated list of remote hosts in the OracleAS Active Failover Cluster

<cp_exec> is the full path to a remote copy utility to be used to copy files from the current node to other nodes in the cluster. By default, afcctl uses /usr/bin/rcp, or /usr/local/bin/scp if the former (rcp) is not found. If neither of these are present or if you wish scp to precede rcp in the invocation order, use the -c <cp_exec> option to specify the copy utility to be used.

Note: Running afcctl with the createbase option is highly recommended right after installation of OracleAS Active Failover Cluster software.

Synchronizing Files From a Node to Other Nodes in an OracleAS Active Failover Cluster

After the initial configuration baseline is set using the createbase option, you can synchronize any configuration changes across the cluster using the sync option. This option synchronizes changed configuration by copying only the modified configuration files from the current node to all nodes in the OracleAS Active Failover Cluster.

The command syntax for invoking a synchronization is:

```
afcctl sync -p <dbname>|-r <host1>,<host2>,...,<hostN> -f <filename>|<file_list_dir>
[-c <cp_exec>] [-l <hostname>]
```

where :

<dbname> is the name of the Infrastructure database

<host1>, <host2>, . . . , <hostN> is a comma separated list of remote hosts in the OracleAS Active Failover Cluster

<filename> is the name of the file to be synchronized

<file_list_dir> is the name of the directory where the .inp files reside

<cp_exec> is the full path to a remote copy utility to be used to copy files from the current node to other nodes in the cluster. By default, afcctl uses /usr/bin/rcp, or

`/usr/local/bin/scp` if the former (`rcp`) is not found. If neither of these are present or if you wish `scp` to precede `rcp` in the invocation order, use the `-c <cp_exec>` option to specify the copy utility to be used.

`<hostname>` is the hostname of the local host at installation time of the Infrastructure. Take note of the following when using the above command line:

- The `-p` option can only be used if the `gsd` process has been started and is running. This option also requires that the Infrastructure database and its instances have been registered with the `srvm` repository, which is the case by default. The `-p` option automatically determines the nodes of the OracleAS Active Failover Cluster deployment and propagates the necessary files to the other nodes.
- The `-r` option can be used in lieu of `-p` to explicitly specify the hostnames of the nodes that need to be synchronized with the node the utility is run on. No other process dependencies exist in this case. Ensure that the hostname(s) specified are valid nodes of the OracleAS Active Failover Cluster deployment.
- `<file_list_dir>` for the `-f` option is the directory where the `.inp` files of the Oracle Application Server 10g Backup and Recovery Tool exist. The `afctl` utility references these files for the list of configuration files that need to be synchronized.
- `<filename>` for the `-f` option is used only when a single file needs to be synchronized across the cluster.
- Before using `afctl`, set up user equivalence so that the `rcp` and `scp` copy utilities can be used without further authentication for the remote hosts. User equivalence is also required for the OracleAS Active Failover Cluster installation. Refer to the high availability chapter in *Oracle Application Server 10g Installation Guide* for instructions on how to set up user equivalence.
- `-l` is optional. It is used to specify the host where `afctl` is run and should be specified only in cases where the local hostname during Infrastructure installation is different from the default hostname.
- It is strongly recommended that the utility is installed on all nodes of the OracleAS Active Failover Cluster deployment. The utility can be invoked from any node of the cluster on which it has been installed. However, as a best practice, designate one node as an administration node and perform all administrative operations and subsequent synchronizations from it.

Listing Modified Files on a Node Since the Last Synchronization

To find out which configuration files on a node in the OracleAS Active Failover Cluster has changed since the last synchronization, use the following command line syntax:

```
afctl list -f <filename>|<file_list_dir>
```

where

`<filename>` is the name of a file that is to be checked for any updates since the last synchronization.

`<file_list_dir>` is the name of the directory where the `.inp` files of the Oracle Application Server 10g Backup and Recovery Tool exist. Files in that directory which have been changed since the last synchronization are listed.

A text file containing a list of files that have changed since the last synchronization is created in the `/tmp` directory. See an example in the section "Example" on page 5-11.

The syntax above can be used on any node of the OracleAS Active Failover Cluster deployment. It displays the files that have changed, since the last synchronization, on

the node it is executed on. The returned list of files can be different depending on the site. To synchronize the listed file(s) individually, the `-f <filename>` option of the `afctl sync` command can be used after determining which version of the file is the latest.

Excluding Specific Configuration Files from Synchronization

Oracle recommends that all nodes are configured similarly. If, however, some configuration files need to be different, their names can be added to the exclude file, `afctl_exclude.inp`, so that they are not synchronized across the cluster when `afctl` is run. `afctl_exclude.inp` is found in the same directory where you uncompressed `afctl.zip`.

- Excluding files may be necessary in situations such as when you want to turn debugging on for only a particular node or change a configuration file temporarily to measure impact on the system. Since the files listed in the exclude file are not synchronized, any changes to them have to be propagated manually to the equivalent files on the other nodes until they are removed from the exclude file. If you do not want the exclusions to be permanent, remove the filenames after performing synchronization.

Note: You can also include custom files to be synchronized across the OracleAS Active Failover Cluster nodes each time the `afctl` utility is run. The file `config.inp` contains rules for this task.

Example

After any administrative operation to Oracle Application Server 10g (through Application Server Console or DCM), which can change any of the configuration files, do the following on each node of the OracleAS Active Failover Cluster:

1. Set the `ORACLE_HOME` environment variable. For example, in a Bourne shell environment, type:

```
$ export ORACLE_HOME=/home/oracle/test1
```

2. Invoke `afctl` with the `list` option to display the configuration files that have changed on the current node.

```
$ afctl list -f ./br_inp_dir
Oracle Application Server Active Failover Cluster Run Time Control Utility
Copyright (c) 2002, 2003 Oracle Corporation. All rights reserved.
```

```
Last Sync up time was Mon Sep 8 11:09:11 2003
Check the following for list of files that have changed since last sync
work/Files_to_Change_and_Copy.23123
```

```
Please look at log/afctl.log for more information.
Exiting...
```

The file `work/Files_to_Change_and_Copy.23123` is created to contain the list of configuration files that have changed since the last synchronization.

3. View the created file to validate that the list of files in it are the ones you want to propagate to the other nodes in the OracleAS Active Failover Cluster. For example:

```
$ cat work/Files_to_Change_and_Copy.23123
```

```
/home/oracle/test1/Apache/Apache/conf/ssl.wlt/default/ewallet.p12  
  
/home/oracle/test1/ldap/admin/oidpwdlldap1  
  
/home/oracle/test1/ldap/admin/oidpwrigit11
```

4. Invoke `afcctl` with the `sync` option to synchronize files from one node in the OracleAS Active Failover Cluster to another.

```
$ afcctl sync -r hasun26 -f ./backup_scripts  
Oracle Application Server Active Failover Cluster Run Time Control Utility  
Copyright (c) 2002, 2003 Oracle Corporation. All rights reserved.
```

```
Files to massage & copy are listed in work/Files_to_Change_and_Copy.22339  
Files to copy are listed in work/Files_to_Copy.22339
```

```
Do you want to sync up files from hasun25.us.oracle.com to  
hasun26.us.oracle.com (y/n) ? y
```

```
Syncing up files  
.....  
.....!  
Syncing completed
```

```
Do you want to update the dcm repository with configuration files from  
"hasun25.us.oracle.com" (y/n) ? y
```

```
DCM update repository started
```

```
DCM update repository completed
```

```
Please look at log/afcctl.log for more information.  
Exiting....
```

Note: Typing `afcctl` without any options displays a list of all options available.

Best Practises for Using `afcctl`

- Avoid making manual changes to configuration files as far as possible. Use Application Server Console or the automated configuration tools (`dcmctl`) to make any changes to the Infrastructure configuration.

If you have to make changes manually, designate a node as the administration node and perform all changes on it. Then, run the `afcctl` tool from this node to propagate the changes to other nodes in the OracleAS Active Failover Cluster.
- When synchronizing configuration files, preferably perform any administration changes and consequent synchronization between the nodes when all nodes of the OracleAS Active Failover Cluster are up. If this is not possible, remember to perform the synchronization with the down nodes after they are up again.
- Use the `list` option regularly on all nodes of the cluster to verify that nothing has been changed locally since the last synchronization. Reconcile any changes across the nodes, if required.
- If sharing `.inp` files with the Oracle Application Server 10g Backup & Recovery Tool, keep in mind that any changes to the exclude and personalization files of the

tool impacts the `afccctl` utility as well. See the *Oracle Application Server 10g Administrator's Guide* for more information on the tool.

Oracle Application Server Disaster Recovery

Disaster recovery refers to how a system recovers from catastrophic site failures caused by natural or unnatural disasters. Examples of catastrophic failures include earthquakes, tornadoes, floods, or fire. Additionally, disaster recovery can also refer to how a system is managed for planned outages. For most disaster recovery situations, the solution involves replicating an entire site, not just pieces of hardware or subcomponents. This also applies to the Oracle Application Server Disaster Recovery (OracleAS Disaster Recovery) solution.

This chapter describes the OracleAS Disaster Recovery solution, how to configure and set up its environment, and how to manage the solution for high availability. The discussion involves both OracleAS middle and Infrastructure tiers in two sites: production and standby. The standby site is configured identically to the production site. Under normal operation, the production site actively services requests. The standby site is maintained to mirror the applications and content hosted by the production site.

The sites are synchronized using the Oracle Application Server Backup and Recovery Tool (for configuration files in the file system) and Oracle Data Guard (for the Infrastructure database). The following table provides a summary of the OracleAS Disaster Recovery strategy:

Table 6–1 Overview of OracleAS Disaster Recovery strategy

Coverage	Procedure	Purpose
Middle Tier Configuration Files	Backup and Recovery Tool	To backup OracleAS configuration files in the production site middle tier nodes and restore the files to the standby site middle tier nodes.
Infrastructure Configuration Files	Backup and Recovery Tool	To backup OracleAS configuration files in the production site Infrastructure node and restore them to the standby site Infrastructure node.
Infrastructure Database	Oracle Data Guard	To ship archive logs from production site Infrastructure database to standby site Infrastructure database. Note that logs are not applied immediately.

In addition to the recovery strategies, configuration and installation of both sites are discussed. For these tasks, two different ways of naming the middle tier nodes are covered as well as two ways of resolving hostnames intra-site and inter-site.

With OracleAS Disaster Recovery, planned outages of the production site can be performed without interruption of service by switching over to the standby site. Unplanned outages are managed by failing over to the standby site. Procedures for switchover and failover are covered in this chapter.

This chapter is organized into the following main sections:

- Oracle Application Server 10g Disaster Recovery Solution
- Setting Up the OracleAS Disaster Recovery Environment
- Installing Oracle Application Server 10g Software
- Synchronizing Baseline Installation with Standby Site
- Backing Up Production Site
- Restoring to Standby Site
- Scheduled Outages
- Unplanned Outages
- Wide Area DNS Operations

See Also: *Oracle Application Server 10g Installation Guide* for instructions on how to install the OracleAS Disaster Recovery solution.

Oracle Application Server 10g Disaster Recovery Solution

The Oracle Application Server Disaster Recovery solution consists of two identically configured sites - one primary (production/active) and one secondary (standby). Both sites have the same number of middle tier and Infrastructure nodes and the same number and types of components installed. In other words, the installations on both sites, middle tier and Infrastructure are identical. Both sites are usually dispersed geographically, and if so, they are connected via a wide area network.

This section describes the overall layout of the solution, the major components involved, and the configuration of these components. It has the following sections:

- Terminology
- Requirements
- Topology

Terminology

Before describing and detailing the OracleAS Disaster Recovery solution, several terms used in this chapter require clear definition in order for the concepts described in this chapter to be understood properly.

Note: The definitions below apply specifically to OracleAS Disaster Recovery. They may have a varying definition outside this context.

- **physical hostname**

For the purpose of discussion in this chapter, a differentiation is made between the terms *physical hostname* and *logical hostname*. Physical hostname is used to refer to

the "internal name" of the current machine. In UNIX, this is the name returned by the command `hostname`.

Physical hostname is used by Oracle Application Server 10g components that are installed on the current machine to reference the local host. During the installation of these components, the installer retrieves the physical hostname from the current machine and stores it in Oracle Application Server 10g configuration metadata on disk.

- **logical hostname**

Logical hostname is a name assigned to an IP address either through the `/etc/hosts` file (in UNIX), `C:\WINDOWS\system32\drivers\etc\hosts` file (in Windows), or through DNS resolution. This name is visible on the network that the host to which it refers to is connected. Often, the logical hostname and physical hostname are literally identical. However, their usage in the OracleAS Disaster Recovery solution necessitates them to be clearly distinct.

- **virtual hostname**

Virtual hostname is used to refer to the name for the Infrastructure host that is specified in the Specify High Availability screen of the OracleAS installer. The virtual hostname is used by the middle tier and Infrastructure components to access the Infrastructure regardless of whether the Infrastructure is a single node installation or part of the OracleAS Cold Failover Cluster solution. Virtual hostname, as used in this chapter, applies only to the Infrastructure host(s).

Requirements

To ensure that your implementation of the OracleAS Disaster Recovery solution performs as designed, the following requirements need to be adhered to:

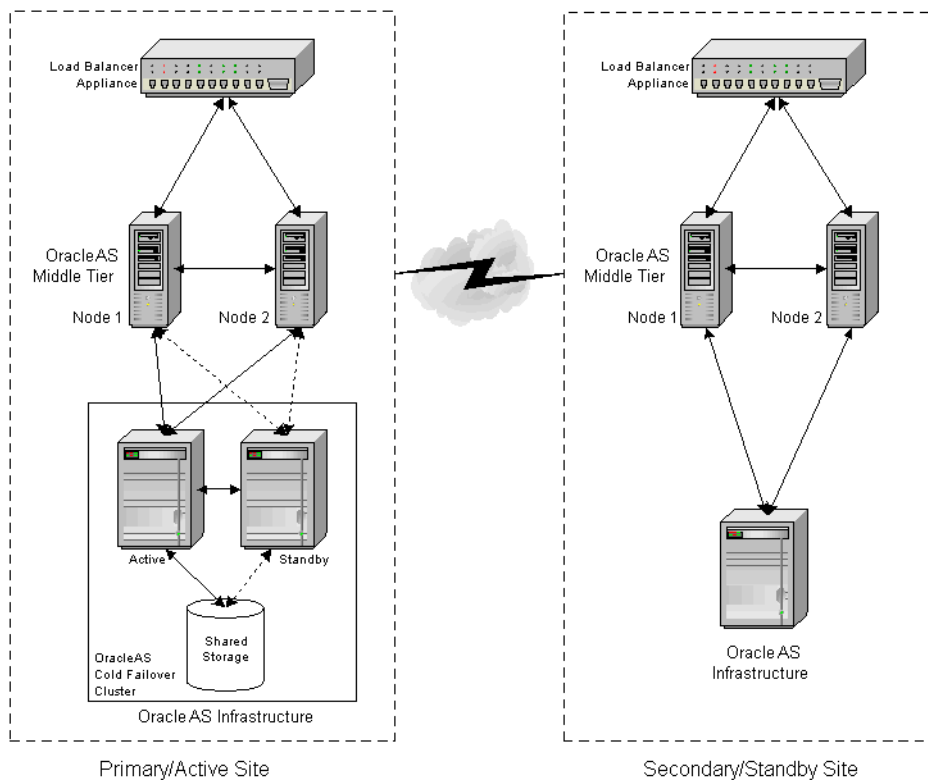
- On each host in the standby site, make sure the following is identical to its equivalent peer in the production site:
 - For the middle tier hosts, physical hostnames
 - Virtual hostname for the Infrastructure. The virtual hostname can be specified in the Specify High Availability screen presented by the installer.
 - Hardware platform.
 - Operating system release and patch levels.
- All installations conform to the requirements listed in the *Oracle Application Server 10g Installation Guide* to install Oracle Application Server.
- Oracle Application Server software is installed in identical directory paths between each host in the production site and its equivalent peer in the standby site.
- Username and password of the user who installed Oracle Application Server must be the same between a host in the production site and its peer in the standby site.
- Numerical user ID of the user who installed Oracle Application Server on that particular node
- Group name of the user who installed Oracle Application Server on that particular node

- Numerical group ID of the group of the user who installed Oracle Application Server on that particular node
- Environment profile
- Shell (command line environment)
- Directory structure and path of the Oracle home for each OracleAS installation on a node. Do not use symbolic links anywhere in the path.
- OracleAS installation types:
 - Middle tier: J2EE and Web Cache, Portal and Wireless, and Business Intelligence and Forms
 - Infrastructure: Metadata Repository and Identity Management (both are required to be installed with the Infrastructure installation type in both sites for the OracleAS Disaster Recovery solution)

Topology

Figure 6–1 depicts the topology of the OracleAS Disaster Recovery solution.

Figure 6–1 Oracle Application Server 10g site-to-site disaster recovery solution (load balancer appliance is optional if only one middle tier node is present)



The procedures and steps for configuring and operating the OracleAS Disaster Recovery solution support 1 to n number of middle tier installations in the production site. The same number of middle tier installations must exist in the standby site. The middle tiers must mirror each other in the production and standby sites.

For the Infrastructure, a uniform number of installations is not required between the production and standby sites. For example, the Oracle Application Server Cold

Failover Clusters solution can be deployed in the production site, and a single node installation of the Infrastructure can be deployed in the standby site. This way, the production site's Infrastructure has protection from host failure using an OracleAS Cold Failover Cluster. Refer to the section "Oracle Application Server Cold Failover Clusters" in Chapter 3 for more information on OracleAS Cold Failover Clusters.

The following are important characteristics of the OracleAS Disaster Recovery solution:

- Middle tier installations are identical between the production and standby sites. In other words, each middle tier installation in the production site has an identically equivalent installation in the standby site. More than one middle tier node is recommended because this enables each set of middle tier installations on each site to be redundant. Being on multiple machines, problems and outages within a site of middle tier installations are transparent to clients.
- The OracleAS Disaster Recovery solution is restricted to identical site configuration to ensure that processes and procedures are kept the same between sites, making operational tasks easier to maintain and execute. Identical site configuration also allows for a higher success rate for manually maintaining the synchronization of Oracle Application Server 10g component configuration files between sites.
- When the production site becomes unavailable due to a disaster, the standby site can become operational within a reasonable time. Client requests are always routed to the site that is operating in the production role. After a failover or switchover operation occurs due to an outage, client requests are routed to another site that assumes the production role. The quality of service offered by the new production site should be the same as that offered by the original production site before the outage.
- The sites are set up in active-passive configuration. An active-passive setup has one primary site used for production and one secondary site that is initially passive (on standby). The secondary site is made active only after a failover or switchover is made to it. Since the sites are symmetrical, after failover or switchover, the original standby site can be kept active as the new production site. After repairing or upgrading the original production site, it can be made into the new standby site. Either site should offer the same level of service to clients as the other.
- The site playing the standby role contains a physical standby of the Oracle Application Server Infrastructure managed by Oracle Data Guard. Oracle Data Guard together with procedures for backing up and restoring Infrastructure configuration files provide configuration synchronization between the production and standby sites. Switchover and failover procedures allow the roles to be traded between the Infrastructures in the two sites. Refer to the section "Setting Up Oracle Data Guard" on page 6-15 for instructions on how to set up Oracle Data Guard to work in the OracleAS Disaster Recovery solution.

Setting Up the OracleAS Disaster Recovery Environment

Prior to the the installation of OracleAS software for the OracleAS Disaster Recovery solution, a number of system level configurations are required. The tasks that accomplish these configurations are:

- Planning and Assigning Hostnames
- Configuring Hostname Resolution

- Secure Shell (SSH) Port Forwarding

This section covers the steps needed to perform these tasks.

Planning and Assigning Hostnames

Before performing the steps to set up the physical and logical hostnames, plan the physical and logical hostnames you wish to use with respect to the entire OracleAS Disaster Recovery solution. The overall approach to planning and assigning hostnames is to meet the following goals:

- OracleAS components in the middle tier and Infrastructure can use the same physical hostnames in their configuration settings regardless of whether the components are in the production or standby site.

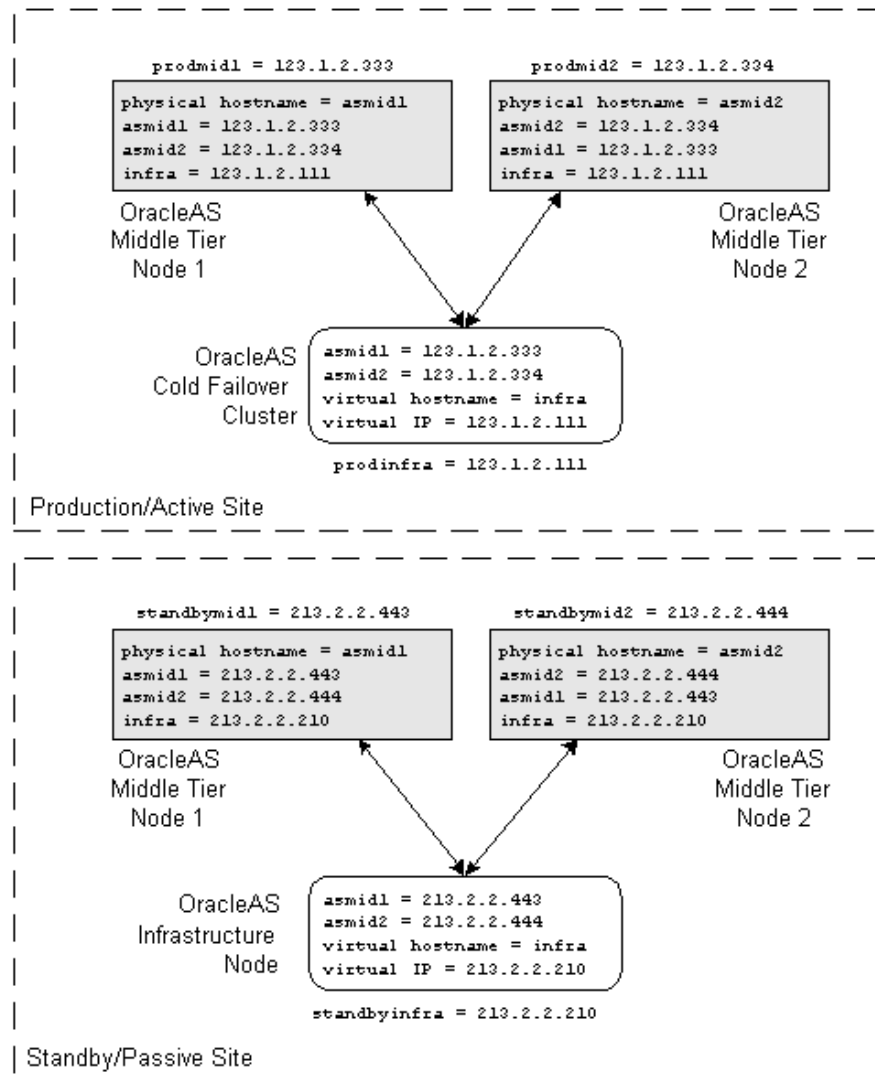
For example, if a middle tier component in the production site uses the name "asmidl" to reach a host in the same site, the same component in the standby site can use the same name to reach asmidl's equivalent peer in the standby site.

- No changes to hostnames (physical, logical, or virtual) are required when the standby site takes over the production role.

Note: Although the physical hostnames in the production and standby sites must remain uniform between the two sites, the resolution of these physical hostnames to the correct hosts can be different. The section "Configuring Hostname Resolution" on page 6-9 explains more on hostname resolution.

To illustrate what should be done to plan and assign hostnames, let us use an example as shown in Figure 6-2.

Figure 6–2 Name assignment example in the production and standby sites



In Figure 6–2, two middle tier nodes exist in the production site. The Infrastructure can be a single node or an OracleAS Cold Failover Cluster solution (represented by a single virtual hostname and a virtual IP, as for a single node Infrastructure). The common names in the two sites are the physical hostnames of the middle tier nodes and the virtual hostname of the Infrastructure. Table 6–2 below details what the physical, logical, and virtual hostnames are in the example:

Table 6–2 Physical, logical, and virtual hostnames in Figure 6–2

Physical Hostnames	Virtual Hostname	Logical Hostnames
asmid1	-	prodmid1, standbymid1
asmid2	-	prodmid2, standbymid2
-	infra	prodinfra, standbyinfra

- *Co-hosting non OracleAS applications*

If the hosts in the production site are running non OracleAS applications, and you wish to co-host OracleAS on the same hosts, changing the physical hostnames of

these hosts may break these applications. In such a case, you can keep these hostnames in the production site and modify the physical hostnames in the standby site to the same as those in the production site. The non OracleAS applications can then also be installed on the standby hosts so that they can act in a standby role for these applications.

As explained in the section "Terminology" on page 6-2, physical, logical, and virtual hostnames have differing purposes in the OracleAS Disaster Recovery solution. They are also set up differently. Information on how the three types of hostnames are set up follow.

Physical Hostnames

The naming of middle tier hosts in both the production and standby sites require the changing of the physical hostname in each host.

In Solaris, to change the physical hostname of a host:

Note: For other UNIX variants, consult your system administrator for equivalent commands in each step.

1. Check to see what the existing physical hostname is set to. Type:

```
prompt> hostname
```
2. Use a text editor, such as `vi`, to edit the name in `/etc/nodename` to your planned physical hostname.
3. For each middle tier host, reboot it for the change to take effect.
4. Repeat step 1 to verify the correct hostname has been set.
5. Repeat the above steps for each host in the production and standby sites.

In Windows, to change the physical hostname of a host:

Note: The user interface elements in your version of Windows may vary from those described in the following steps.

1. In the Start menu, select Control Panel.
2. Double-click the System icon.
3. Select the Advance tab.
4. Select Environment variables.
5. Under the User Environment variables for the installer account, select New to create a new variable.
6. Enter the name of the variable as "`_CLUSTER_NETWORK_NAME_`".
7. For the value of this variable, enter the planned physical hostname.

Logical Hostnames

The logical hostnames used in the OracleAS Disaster Recovery solution are defined in DNS. These hostnames are visible in the network that the solution uses and are resolved through DNS to the appropriate hosts via the assigned IP address in the DNS system. You need to add these logical hostnames and their corresponding IP addresses to the DNS system.

Using the example in Figure 6–2, the following should be the additions made to the DNS system serving the entire network that encompasses the production and standby sites:

```

prodmid1.oracle.com      IN      A       123.1.2.333
prodmid2.oracle.com      IN      A       123.1.2.334
prodfra.oracle.com       IN      A       123.1.2.111
standbymid1.oracle.com   IN      A       213.2.2.443
standbymid2.oracle.com   IN      A       213.2.2.444
standbyinfra.oracle.com  IN      A       213.2.2.210

```

Virtual Hostname

As defined in the Terminology section, virtual hostname applies to the Infrastructure only. It is specified during installation of the Infrastructure. When you run the Infrastructure installation type, a screen called "Specify High Availability" appears to provide a textbox to enter the virtual hostname of the Infrastructure that is being installed. Refer to the *Oracle Application Server 10g Installation Guide* for more details.

For the example in Figure 6–2, when you install the production site's Infrastructure, enter its virtual hostname, "infra", when you see the Specify High Availability screen. Enter the same virtual hostname when you install the standby site's Infrastructure.

Note: If the Infrastructure is installed in a OracleAS Cold Failover Cluster solution, the virtual hostname is the name that is associated with the virtual IP of the OracleAS Cold Failover Cluster.

Configuring Hostname Resolution

In the Oracle Application Server Disaster Recovery solution, one of two ways of hostname resolution can be configured to resolve the hostnames you planned and assigned in the previous section. These are:

- Using Local Hostnaming File Resolution
- Using DNS Resolution

In UNIX, the order of the method of name resolution can be specified using the "hosts" parameter in the file `/etc/nsswitch.conf`. The following is an example of the `hosts` entry:

```
hosts:      files dns nis
```

In the above statement, local hostnaming file resolution is preferred over DNS and NIS (Network Information Service) resolutions. When a hostname is required to be resolved to an IP address, the `/etc/hosts` file (UNIX) or `C:\WINDOWS\system32\drivers\etc\hosts` file is consulted first. In the event that a hostname cannot be resolved using local hostnaming resolution, DNS is used. (NIS resolution is not used for the OracleAS Disaster Recovery solution.) Refer to your UNIX system's documentation if you wish to find out more about `/etc/nsswitch.conf`.

Using Local Hostnaming File Resolution

This method of hostname resolution relies on a local hostnaming file to contain the requisite hostname-to-IP address mappings. In UNIX, this file is `/etc/hosts`. In Windows, this file is `C:\WINDOWS\system32\drivers\etc\hosts`.

To use the local hostnaming file to resolve hostnames for the OracleAS Disaster Recovery solution in UNIX, for each middle tier and Infrastructure host in both the production and standby sites, perform the following:

1. Use a text editor, such as `vi`, to edit the `/etc/nsswitch.conf` file. With the "hosts:" parameter, specify "files" as the first choice for hostname resolution.
2. Edit the `/etc/hosts` file to include the following:
 - The physical hostnames and their correct IP addresses of all middle tier nodes in the current site. Ensure that the first entry is the hostname and IP address of the current node.

For example, if you are editing the `/etc/hosts` file of a middle tier node in the production site, enter all the middle tier physical hostnames and their IP addresses in the production site beginning the list with the current host. (Note that you should also include fully qualified hostnames in addition to the abbreviated hostnames. See Table 6–3.)
 - The virtual hostname of the Infrastructure in the current site.

For example, if you are editing the `/etc/hosts` of a middle tier node in the standby site, enter the virtual hostname, fully qualified and abbreviated, and IP address of the Infrastructure host in the standby site.
3. Reboot each host after editing the above files.
4. From each host, ping each physical hostname that is valid in its particular site to ensure that the IP addresses have been assigned correctly.

For the example in Figure 6–2, on the `asmid1` host, use the following commands in:

```
ping asmid1
```

The returned IP address should be 123.1.2.333.

```
ping asmid2
```

The returned IP address should be 123.1.2.334.

```
ping infra
```

The returned IP address should be 123.1.2.111.

Note: Some UNIX variants, such as Solaris, require the `-s` option to return an IP address.

In Windows, the method of ordering hostname resolution varies depending on the Windows version. Refer to the documentation for your version of Windows for the appropriate steps.

Using the example in Figure 6–2, Table 6–3 contains the required entries in the `/etc/hosts` file of each UNIX host. The entries in the Windows `C:\WINDOWS\system32\drivers\etc\hosts` file should reflect similarly.

Table 6–3 Logical and virtual hostname entries in each `/etc/hosts` file of example in Figure 6–2

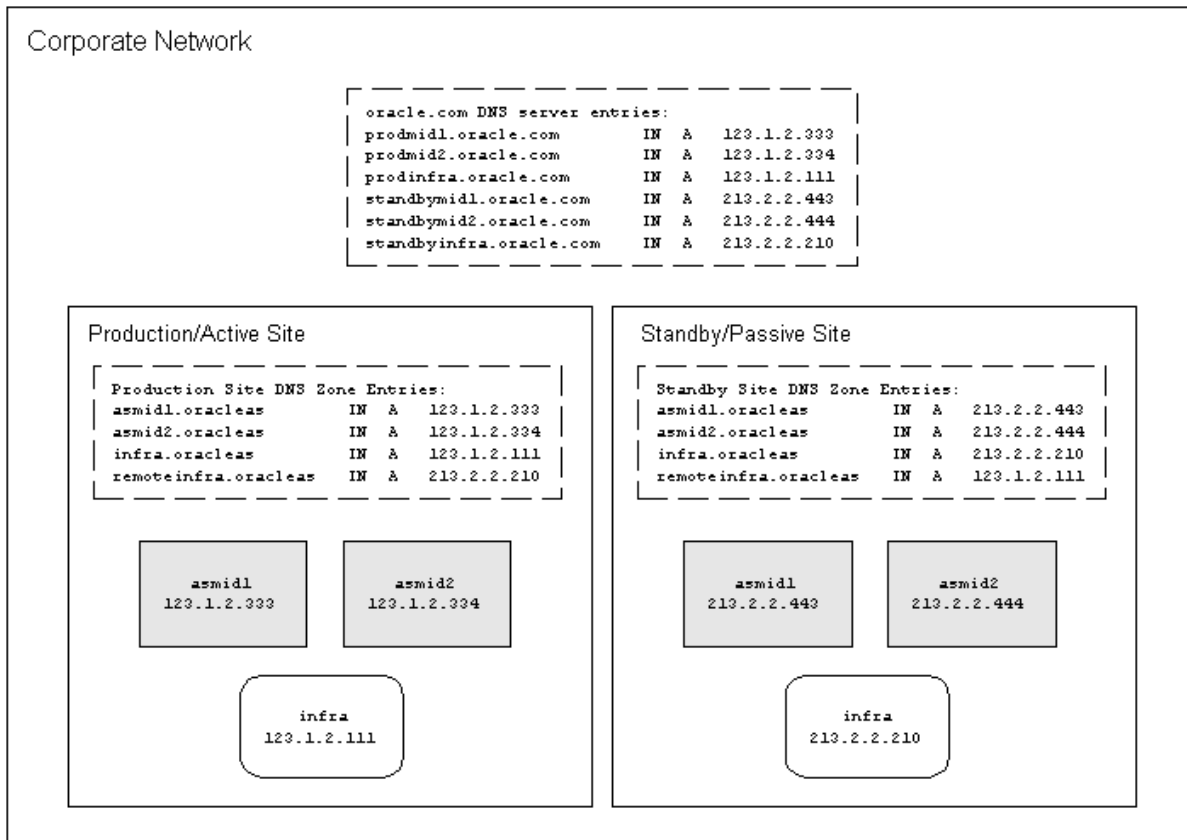
Host	Entries in <code>/etc/hosts</code>
asmid1 in production site	123.1.2.333 asmid1.oracle.com asmid1 123.1.2.334 asmid2.oracle.com asmid2 123.1.2.111 infra.oracle.com infra
asmid2 in production site	123.1.2.334 asmid2.oracle.com asmid2 123.1.2.333 asmid1.oracle.com asmid1 123.1.2.111 infra.oracle.com infra
infra in production site	123.1.2.111 infra.oracle.com infra 123.1.2.333 asmid1.oracle.com asmid1 123.1.2.334 asmid2.oracle.com asmid2
asmid1 in standby site	213.2.2.443 asmid1.oracle.com asmid1 213.2.2.444 asmid2.oracle.com asmid2 213.2.2.210 infra.oracle.com infra
asmid2 in standby site	213.2.2.444 asmid2.oracle.com asmid2 213.2.2.443 asmid1.oracle.com asmid1 213.2.2.210 infra.oracle.com infra
infra in standby site	213.2.2.210 infra.oracle.com infra 213.2.2.443 asmid1.oracle.com asmid1 213.2.2.444 asmid2.oracle.com asmid2

Using DNS Resolution

To set up the OracleAS Disaster Recovery solution to use DNS hostname resolution, site-specific DNS servers must be set up in the production and standby sites in addition to the overall corporate DNS servers (usually more than one DNS server exists in a corporate network for redundancy). Figure 6–3 provides an overview of this setup.

See Also: Appendix A, "Setting Up a DNS Server" for instructions on how to set up a DNS server in a UNIX environment.

Figure 6–3 DNS resolution topology overview



For the above topology to work, the following requirements and assumptions are made:

- The production and standby sites' DNS servers are not aware of each other. They make non authoritative lookup requests to the overall corporate DNS servers if they fail to resolve any hostnames within their specific sites.
- The production site and standby site DNS servers contain entries for middle tier physical hostnames and Infrastructure virtual hostnames. Each DNS server contain entries of hostnames within their own site only. The sites have a common domain name that is different from that of the overall corporate domain name.
- The overall corporate DNS servers contain logical hostname entries for the middle tier hosts and Infrastructure hosts of both production and standby sites.
- In UNIX, the `/etc/hosts` file in each host does not contain any entries for the physical, logical, or virtual hostnames of any host in either site. In Windows, this applies to the file `C:\WINDOWS\system32\drivers\etc\hosts`.

To set up the OracleAS Disaster Recovery solution for DNS resolution:

1. Configure each of the overall corporate DNS servers with the logical hostnames of all the hosts in the production and standby sites. Using the example in Figure 6–2, the following entries are made:

```

prodmid1.oracle.com      IN A 123.1.2.333
prodmid2.oracle.com      IN A 123.1.2.334
prodfinfra.oracle.com    IN A 123.1.2.111
standbymid1.oracle.com   IN A 213.2.2.443
standbymid2.oracle.com   IN A 213.2.2.444
  
```

```
standbyinfra.oracle.com IN A 213.2.2.210
```

2. For each site, production and standby, create a unique DNS zone by configuring a DNS server as follows:
 - a. Select a unique domain name to use for the two sites that is different from the corporate domain name. As an example, let's use the name "oracleaseas" for the domain name for the two sites in Figure 6–2. The high level corporate domain name is oracle.com.
 - b. Configure the DNS server in each site to point to the overall corporate DNS servers for unresolved requests.
 - c. Populate the DNS servers in each site with the physical hostnames of each middle tier host and the virtual hostname of each Infrastructure host. Include the domain name selected in the previous step.

For the example in Figure 6–2, the entries are as follows:

For the production site's DNS:

```
asmid1.oracleaseas IN A 123.1.2.333
asmid2.oracleaseas IN A 123.1.2.334
infra.oracleaseas IN A 123.1.2.111
```

For the standby site's DNS:

```
asmid1.oracleaseas IN A 213.2.2.443
asmid2.oracleaseas IN A 213.2.2.444
infra.oracleaseas IN A 213.2.2.210
```

Note: If you are using the OracleAS Cold Failover Cluster solution for the Infrastructure in either site, enter the cluster's virtual hostname and virtual IP address. For example, in the previous step above, *infra* is the virtual hostname and 123.1.2.111 is the virtual IP of the cluster in the production site. For more information on the OracleAS Cold Failover Cluster solution, see "Oracle Application Server Cold Failover Clusters" on page 3-7.

Additional DNS Server Entries for Oracle Data Guard

Because Oracle Data Guard technology is used to synchronize the production and standby Infrastructure databases, the production Infrastructure must be able to reference the standby Infrastructure and vice versa.

For this to work, the IP address of the standby Infrastructure host must be entered in the production site's DNS server with a unique hostname with respect to the production site. Similarly, the IP address of the production Infrastructure host must be entered in the standby site's DNS server with the same hostname. The reason for these DNS entries is that Oracle Data Guard uses TNS Names to direct requests to the production and standby Infrastructures. Hence, the appropriate entries must be made to the `tnsnames.ora` file as well.

Using the example in Figure 6–2 and assuming that the selected name for the remote Infrastructure is "remoteinfra", the entries in the DNS server in the production site are:

```
asmid1.oracleaseas IN A 123.1.2.333
```

asmid2.oracleas	IN	A	123.1.2.334
infra.oracleas	IN	A	123.1.2.111
remoteinfra.oracleas	IN	A	213.2.2.210

And, for the standby site, its DNS server should have the following entries:

asmid1.oracleas	IN	A	213.2.2.443
asmid2.oracleas	IN	A	213.2.2.444
infra.oracleas	IN	A	213.2.2.210
remoteinfra.oracleas	IN	A	123.1.2.111

Secure Shell (SSH) Port Forwarding

Oracle Data Guard sends redo data across the network to the standby system using OracleNet. SSH tunneling should be used with Oracle Data Guard as an integrated way to encrypt and compress the redo data before it is transmitted by the production system and subsequently decrypt and uncompress the redo data when it is received by the standby system.

See Also:

- Implementing SSH port forwarding with Data Guard:
<http://metalink.oracle.com/metalink/plsql/showdoc?db=NOT&id=225633.1>
- Troubleshooting Data Guard network issues:
<http://metalink.oracle.com/metalink/plsql/showdoc?db=NOT&id=241925.1>

Installing Oracle Application Server 10g Software

This section provides an overview of the steps for installing the OracleAS Disaster Recovery solution. After following the instructions in the previous section to set up the environment for the solution, go through this section for an overview of the installation process. Thereafter, follow the detailed instructions in the *Oracle Application Server 10g Installation Guide* to install the solution.

Note: To assign identical ports to be used by symmetrical hosts in the production and standby sites, static port definitions can be used. These definitions are defined in a file, for example, named `staticports.ini`, that is declared in the command that starts the installer (see below). Detailed information on this static ports file is found in the *Oracle Application Server 10g Installation Guide*.

The following is the overall sequence for installing the OracleAS Disaster Recovery solution:

1. Install OracleAS Infrastructure in the production site (refer to *Oracle Application Server 10g Installation Guide*).
2. Install OracleAS Infrastructure in the standby site (refer to *Oracle Application Server 10g Installation Guide*).
3. Start the Infrastructure in each site before installing the middle tiers for that site.
4. Install the middle tiers in the production site (refer to *Oracle Application Server 10g Installation Guide*).

5. Install the middle tiers in the standby site (refer to *Oracle Application Server 10g Installation Guide*).

Note the following important points when you perform the installation:

- The Infrastructure Identity Management and OracleAS Metadata Repository components must be installed on the same host. These components cannot be distributed over multiple hosts. (This requirement also applies to the OracleAS Cold Failover Cluster and OracleAS Active Failover Cluster solutions.)
- Ensure that the same ports are used by equivalent peer hosts in both sites. For example, the `asmid1` host in the standby site must use the same ports as the `asmid1` host in the production site. Utilize a static ports definition file for this purpose (see note above and the next point).
- Start the installer from the command line to use a static ports definition file. The command syntax is different for the middle tier and Infrastructure hosts.

For each middle tier host, use the following syntax:

In UNIX:

```
runInstaller oracle.iappserver.iapptop:s_staticPorts=staticports.ini
```

In Windows:

```
setup oracle.iappserver.iapptop:s_staticPorts=staticports.ini
```

For each Infrastructure host, use the following syntax:

In UNIX:

```
runInstaller oracle.iappserver.infrastructure:s_staticPorts=staticports.ini
```

In Windows:

```
setup oracle.iappserver.infrastructure:s_staticPorts=staticports.ini
```

- In the installer's Select Configuration Options screen, ensure that you select the High Availability Addressing option.
- During Infrastructure installation, specify the virtual address assigned to the Infrastructure in the Specify High Availability screen.
- For the middle tier hosts, any of the available middle tier installation types can be installed. (Ensure that the Infrastructure services have been started for a site before installing any middle tiers in that site.)
- During each middle tier installation, specify the Infrastructure's virtual hostname as the Infrastructure database.
- Start the OracleAS services on the hosts in each site starting with the Infrastructure.

Setting Up Oracle Data Guard

For OracleAS Disaster Recovery purposes, the metadata information maintained within the the Infrastructure database is kept in synchronization by utilizing Oracle Data Guard technology. This technology propagates all database changes at the production site to the standby site for disaster tolerance.

Note that for OracleAS Disaster Recovery, archive logs are shipped from the production Infrastructure database to the standby Infrastructure database but are not applied. The application of these logs have to be done with the synchronization of file

system configuration information, which is discussed in the section "Backing Up Configuration Files (Infrastructure and Middle Tier)".

Note: For configuration information that is stored outside the Infrastructure database in the file system, the OracleAS Backup and Recovery Tool is used to synchronize this information between the two sites.

The setup of Oracle Data Guard for OracleAS Disaster Recovery involves the following steps:

- Enable ARCHIVELOG Mode for Production Database
- Identifying the Production Database Datafiles
- Make a Copy of the Production Database
- Create a Control File for the Standby Database
- Prepare the Initialization Parameter File to be Copied to the Standby Database
- Copy Files from the Production System to the Standby System
- Set Initialization Parameters for the Physical Standby Database
- Create a Windows Service (for Microsoft Windows systems)
- Create a New Password File on the Standby System
- Configure Listeners for the Production and Standby Databases
- Enable Dead Connection Detection on the Standby System
- Create Oracle Net Service Names
- Create a Server Parameter File for the Standby Database
- Start the Physical Standby Database
- Enable Archiving to the Physical Standby Database
- Start Remote Archiving
- Verify the Physical Standby Database

Enable ARCHIVELOG Mode for Production Database

By default, the production database does not have ARCHIVELOG mode enabled. However, it needs to be in ARCHIVELOG mode in order to ship archive logs to the standby database. The default destination directory for archive logs is:

UNIX:

```
<INFRA_ORACLE_HOME>/dbs/arch/
```

Windows:

```
<INFRA_ORACLE_HOME>\database\archive
```

To enable ARCHIVELOG mode:

1. Make sure the ORACLE_HOME and ORACLE_SID (the default is asdb) environment variables are properly set.

2. Ensure that the database is not being used by stopping all usage of the Infrastructure database. Execute the following commands on the Infrastructure database host:

UNIX:

```
<ORACLE_HOME>/bin/emctl stop iasconsole
<ORACLE_HOME>/opmn/bin/opmnctl stopall
```

Windows:

```
<ORACLE_HOME>\bin\emctl stop iasconsole
<ORACLE_HOME>\opmn\bin\opmnctl stopall
```

3. Ensure that Enterprise Manager has been stopped using the following command:

UNIX:

```
<ORACLE_HOME>/bin/emctl status iasconsole
```

Windows:

```
<ORACLE_HOME>\bin\emctl status iasconsole
```

4. Execute the following commands to connect and confirm that ARCHIVELOG mode is not enabled:

```
<ORACLE_HOME>/bin/sqlplus /nolog
SQL> connect sys/<password> as sysdba
SQL> archive log list
Database log mode                No Archive Mode
Automatic archival                Disabled
Archive destination               /private/oracle/oracleas/dbs/arch
Oldest online log sequence       4
Current log sequence              6
```

In Windows, the `sqlplus` command can be executed as:

```
<ORACLE_HOME>\bin\sqlplus /nolog
```

In Windows, the archive destination should be `<ORACLE_HOME>\database\archive`.

5. Shutdown the database instance. Execute:


```
SQL> shutdown immediate
```
6. For Windows only, create an `spfile` using the following commands:


```
SQL> connect sys/<password> as sysdba
SQL> create spfile from pfile;
```
7. Start up the instance, mount it, but do not open the database.


```
SQL> startup mount;
```
8. Enable database ARCHIVELOG mode.


```
SQL> alter database archivelog;
SQL> alter system set log_archive_start=true scope=spfile;
SQL> alter system set LOG_ARCHIVE_DEST_1=
'LOCATION=/private/oracle/oracleas/oradata MANDATORY' SCOPE=BOTH;
```

In Windows, substitute the archive log path shown above appropriately (<ORACLE_HOME>\oradata).

9. Shut down and restart the database instance.

```
SQL> shutdown
SQL> connect sys/<password> as sysdba
SQL> startup
```

10. Verify that the database is now in ARCHIVELOG mode.

Execute the following command and verify that Database log mode is in Archive Mode and Automatic archival is Enabled.

```
SQL> archive log list
Database log mode           Archive Mode
Automatic archival         Enabled
Archive destination        /private/oracle/oracleas/oradata
Oldest online log sequence 4
Next log sequence to archive 6
Current log sequence       6
```

In Windows, substitute the archive destination path shown above appropriately.

Identifying the Production Database Datafiles

On the production database, query the V\$DATAFILE view to list the files that will be used to create the physical standby database as follows (UNIX paths are shown):

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
-----
/private/oracle/oracleas/oradata/asdb/system01.dbf
/private/oracle/oracleas/oradata/asdb/undotbs01.dbf
/private/oracle/oracleas/oradata/asdb/drsys01.dbf
/private/oracle/oracleas/oradata/asdb/dcm.dbf
/private/oracle/oracleas/oradata/asdb/portal.dbf
.
.
.
24 rows selected.
```

Make a Copy of the Production Database

On the production database, perform the following steps to make a closed backup copy of the production database:

1. Shut down the production database. Issue the following SQLPLUS statement to shut down the production database:

```
SQL> SHUTDOWN IMMEDIATE;
```

2. Copy the datafiles to a temporary location. Copy the datafiles that you identified in the previous section, "Identifying the Production Database Datafiles", to a temporary location using an operating system utility copy command. The following example uses the UNIX cp command (<ORACLE_HOME> is /private/oracle/oracleas):

```
mkdir /private/standby
cp /private/oracle/oracleas/oradata/asdb/system01.dbf /private/standby/system01.dbf
cp /private/oracle/oracleas/oradata/asdb/undotbs01.dbf /private/standby/undotbs01.dbf
cp /private/oracle/oracleas/oradata/asdb/drsys01.dbf /private/standby/drsys01.dbf
cp /private/oracle/oracleas/oradata/asdb/dcm.dbf /private/standby/dcm.dbf
```



```
cp /private/oracle/oracleas/oradata/asdb/portal.dbf /private/standby/portal.dbf
```

In Windows, use the `md` command to create a new directory and the `copy` command to copy the files.

- Restart the production database. Issue the following SQLPLUS statement to restart the production database:

```
SQL> STARTUP;
```

Create a Control File for the Standby Database

On the production database, create the control file for the standby database, as shown in the following example in UNIX (in Windows, substitute with your appropriate path):

```
SQL> alter database create standby controlfile as '/private/standby/asdb.ctl';
```

The filename for the newly created standby control file must be different from the filename of the current control file of the production database. In the example above, it is created in the new temporary directory for the standby. The control file must also be created after the last time stamp for the backup datafiles.

Prepare the Initialization Parameter File to be Copied to the Standby Database

Create a traditional text initialization parameter file from the server parameter file used by the production database; a traditional text initialization parameter file can be copied to the standby location and modified. For example in UNIX (in Windows, substitute with your appropriate path):

```
SQL> create pfile='/private/standby/initasdb.ora' from spfile
```

Later, in the section "Create a Server Parameter File for the Standby Database" and thereafter, you will convert this file back to a server parameter file after it is modified to contain the parameter values appropriate for use with the physical standby database.

Note: You cannot use a single control file for both the production and standby databases.

Copy Files from the Production System to the Standby System

On the production system, use an operating system copy utility to copy the binary files mentioned in the following steps from the production system to the standby system. Before copying, make sure the following tasks have been performed:

- Stop any processing against the infrastructure by using `opmnctl` and `emctl` as specified in the section "Enable ARCHIVELOG Mode for Production Database".
- Backup datafiles created in the section "Make a Copy of the Production Database".
- Standby control file created in the section "Create a Control File for the Standby Database".
- Initialization parameter file created in the section "Prepare the Initialization Parameter File to be Copied to the Standby Database".

- Clean up standby database files:

Note: The commands in this step must be run on the standby database.

In UNIX:

```
standby> rm /private/oracle/oracleas/dbs/spfileasdb.ora
standby> rm /private/oracle/oracleas/dbs/orapwasdb
standby> rm /private/oracle/oracleas/oradata/asdb/*.*
```

In Windows:

```
del <ORACLE_HOME>\database\spfileasdb.ora
del <ORACLE_HOME>\database\PWDasdb.ora
del <ORACLE_HOME>\oradata\asdb\*
```

2. Identify the location of the init parameter file and clean this up (this is performed on the standby system).

In UNIX:

```
standby> ls -l initasdb.ora
lrwxrwxrwx 1 nedcias svrtech      54 Nov 10 09:25 initasdb.ora ->
/private/oracle/oracleas/admin/asdb/pfile/initasdb.ora
standby> rm /private/oracle/oracleas/admin/asdb/pfile/initasdb.ora
```

In Windows:

```
del <ORACLE_HOME>\database\initasdb.ora
```

Note: These copy steps are examples and depending on the network configuration other utilities may need to be used.

3. Copy the parameter initialization file in the previous step from the production machine to the standby machine.

In UNIX:

```
production> cd /private/standby
production> cp initasdb.ora
/net/standby/private/oracle/oracleas/admin/asdb/pfile/initasdb.ora
```

In Windows, use Windows Explorer or FTP to perform the copy operation.

4. Copy the control files to the standby machine:

For example, in UNIX:

```
production> cp asdb.ctl /net/standby/private/oracle/oracleas/oradata/asdb/control01.ctl
production> cp asdb.ctl /net/standby/private/oracle/oracleas/oradata/asdb/control02.ctl
production> cp asdb.ctl /net/standby/private/oracle/oracleas/oradata/asdb/control03.ctl
```

In Windows, you can use Windows Explorer to copy and paste the files or use FTP to copy them to the following location: <STANDBY_ORACLE_HOME>\oradata\asdb\

5. Copy the data files to the standby machine:

In UNIX:

```
production> cp <ORACLE_HOME>/oradata/asdb/*.dbf
```

```
/net/standby/private/oracle/oracleas/oradata/asdb/.
```

In Windows, use Windows Explorer to copy and paste the files or use FTP to copy the *.dbf files to the following location: <STANDBY_ORACLE_HOME>\oradata\asdb\

Set Initialization Parameters for the Physical Standby Database

Although most of the initialization parameter settings in the text initialization parameter file that you copied from the production system are also appropriate for the physical standby database, some modifications need to be made.

The following steps detail the parameters to modify or add to the standby initialization parameter file:

1. Edit the following parameters in the `initasdb.ora` file that was copied over from the production system:
 - *.standby_archive_dest - Specify the location of the archived redo logs that will be received from the production database.
 - *.standby_file_management - Set to AUTO.
 - *.remote_archive_enable - Set to TRUE.

For example, in UNIX:

```
*.standby_archive_dest='/private/oracle/oracleas/standby/'
*.standby_file_management=AUTO
*.remote_archive_enable=TRUE
```

In Windows, substitute the appropriate path for the `standby_archive_dest` parameter.

2. Create the directory that is specified for the `standby_archive_dest` parameter. For example, in UNIX:

```
Standby> mkdir /private/oracle/oracleas/standby
```

In Windows, use Windows Explorer or the `md` command to create a new directory.

3. At the standby site, make sure the `ORACLE_HOME` and `ORACLE_SID` (the default is `asdb`) environment variables are properly set.

For example, in UNIX:

```
Standby> setenv ORACLE_HOME /private/oracle/oracleas
Standby> setenv ORACLE_SID asdb
```

In Windows, these variables should be set correctly in the registry.

Create a Windows Service (for Microsoft Windows systems)

If the standby system is running on a Windows system, use the `ORADIM` utility to create a Windows Service. For example:

```
<ORACLE_HOME>\bin\oradim -NEW -SID payroll2 -STARTMODE manual
```

Create a New Password File on the Standby System

A new password file has to be created on the standby system. Use the commands in the following example:

In UNIX:

```
standby> cd $ORACLE_HOME/dbs
standby> $ORACLE_HOME/bin/orapwd file=orapwasdb password=<passwd>
```

In Windows:

```
cd %ORACLE_HOME%\database
%ORACLE_HOME%\bin\orapwd file=PWDasdb.ora password=<passwd>
```

Configure Listeners for the Production and Standby Databases

On both the production and standby sites, the Oracle Net Manager can be used to configure a listener for the respective databases. This is completed during the installation of the Infrastructure. During the installation process, the listeners were configured and started. The following is the configuration file that maintains the listener configuration information.

In UNIX:

```
<ORACLE_HOME>/network/admin/listener.ora
```

In Windows:

```
<ORACLE_HOME>\network\admin\listener.ora
```

Any modifications to this file requires the listeners to be restarted using the following commands:

In UNIX:

```
<ORACLE_HOME>/bin/lsnrctl stop
<ORACLE_HOME>/bin/lsnrctl start
```

In Windows:

```
<ORACLE_HOME>\bin\lsnrctl stop
<ORACLE_HOME>\bin\lsnrctl start
```

Enable Dead Connection Detection on the Standby System

Enable dead connection detection by setting the `SQLNET.EXPIRE_TIME` parameter to 2 in the `SQLNET.ORA` parameter file on the standby system. For example:

```
SQLNET.EXPIRE_TIME=2
```

Create Oracle Net Service Names

On both the production and standby systems, use Oracle Net Manager to create a network service name for the production and standby databases that is to be used by log transport services.

The Oracle Net service name must resolve to a connect descriptor that uses the same protocol, host address, port, and SID that are specified in the listener configuration file `listener.ora`. The connect descriptor must also specify that a dedicated server be used.

The following steps illustrate how the above should be set up:

1. On both the production and standby hosts, there needs to be an entry to point at the local database as well as the remote copy. Execute the `TNSPING` command on both nodes to confirm that the entries in the `TNSNAMES.ORA` file are correct.

2. Add the following to the `TNSNAMES.ORA` file on the production database host that points to the standby database (the standby hostname in this example is `standby.oracle.com`):

```
ASDB_REMOTE =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (HOST = standby.oracle.com) (PORT=1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = asdb.oracle.com)
    )
  )
```

3. TNSPING the remote host to verify that it can be reached:

In UNIX:

```
production> /private/oracle/oracleas/bin/tnsping asdb_remote
```

In Windows:

```
<ORACLE_HOME>\bin\tnsping asdb_remote
```

Note: The `tnsping` command may require a fully qualified hostname. The `NAMES.DEFAULT_DOMAIN` setting in `sqlnet.ora` determines whether a fully qualified hostname is required or not.

4. Add the following entry to the `TNSNAMES.ORA` file on the standby host that points to the production database (the production host in this example is assumed to be `production.oracle.com`):

```
ASDB_REMOTE =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (HOST = production.oracle.com) (PORT=1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = asdb.oracle.com)
    )
  )
```

5. TNSPING the production host to verify that it can be reached:

In UNIX:

```
standby> /private/oracle/oracleas/bin/tnsping asdb_remote
```

In Windows:

```
<ORACLE_HOME>\bin\tnsping asdb_remote
```

Create a Server Parameter File for the Standby Database

On an idle standby database, use the `SQLPLUS CREATE` statement to create a server parameter file from the text initialization parameter file that was edited in the section "Set Initialization Parameters for the Physical Standby Database". For example:

1. Make sure the ORACLE_HOME and ORACLE_SID (the default is asdb) environment variables are properly set.
2. Connect and create an spfile.

In UNIX:

```

$ORACLE_HOME/bin/sqlplus /nolog
SQL> connect sys/password as sysdba
SQL> create spfile='/private/oracle/oracleas/dbs/spfileasdb.ora' from
pfile='/private/oracle/oracleas/dbs/initasdb.ora';
```

In Windows:

```

%ORACLE_HOME%\bin\sqlplus /nolog
SQL> connect sys/password as sysdba
SQL> create spfile='<ORACLE_HOME>\database\spfileasdb.ora' from
pfile='<ORACLE_HOME>\database\initasdb.ora';
```

Start the Physical Standby Database

On the standby database, issue the following SQLPLUS statements to start and mount the database in standby mode:

```

SQL> STARTUP NOMOUNT;
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

Enable Archiving to the Physical Standby Database

This section describes the minimum amount of work you must do on the production database to set up and enable archiving to the physical standby database.

To configure archive logging from the production database to the standby site, the LOG_ARCHIVE_DEST_n and LOG_ARCHIVE_DEST_STATE_n parameters must be defined. The service name used must be the same as that set up in the "Create Oracle Net Service Names" section.

The following statements executed on the production database set the initialization parameters needed to enable archive logging to the standby site:

```

SQL> alter system set log_archive_dest_2='SERVICE=asdb_remote' scope=both;
SQL> alter system set log_archive_dest_state_2=enable scope=both;
```

Start Remote Archiving

Archiving of redo logs to the remote standby location does not occur until after a log switch. A log switch occurs, by default, when an online redo log becomes full.

To force the current redo logs to be archived immediately, use the SQLPLUS ALTER SYSTEM statement on the production database:

```

SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

Verify the Physical Standby Database

Once you create the physical standby database and set up log transport services, you should verify that database modifications are being successfully shipped from the production database to the standby database.

To see the new archived redo logs that were received on the standby database, you should first identify the existing archived redo logs on the standby database, archive a few logs on the production database, and then check the standby database again. The following steps illustrate how to perform these tasks:

1. Identify the existing archived redo logs.

On the standby database, query the V\$ARCHIVED_LOG view to identify existing archived redo logs:

```
SQL> SELECT SEQUENCE#, FIRST_TIME, NEXT_TIME FROM V$ARCHIVED_LOG ORDER BY
SEQUENCE#;
SEQUENCE#  FIRST_TIME          NEXT_TIME
-----
8          11-JUL-02 17:50:45 11-JUL-02 17:50:53
9          11-JUL-02 17:50:53 11-JUL-02 17:50:58
10         11-JUL-02 17:50:58 11-JUL-02 17:51:03
3 rows selected.
```

2. Archive the current log.

On the production database, archive the current log using the following SQLPLUS statement:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

3. Verify that the new archived redo log has been received.

On the standby database, query the V\$ARCHIVED_LOG view to verify that the redo log has been received using the following statement:

```
SQL> SELECT SEQUENCE#, FIRST_TIME, NEXT_TIME FROM V$ARCHIVED_LOG ORDER BY
SEQUENCE#;
SEQUENCE#  FIRST_TIME          NEXT_TIME
-----
8          11-JUL-02 17:50:45 11-JUL-02 17:50:53
9          11-JUL-02 17:50:53 11-JUL-02 17:50:58
10         11-JUL-02 17:50:58 11-JUL-02 17:51:03
11         11-JUL-02 17:51:03 11-JUL-02 18:34:11
4 rows selected.
```

The logs have now been shipped and are available on the standby database. To confirm that they are there, list out the contents of the directory <ORACLE_HOME>/standby.

4. Verify that the new archived redo log has NOT been applied.

On the standby database, query the V\$ARCHIVED_LOG view to verify that the archived redo log has not been applied.

```
SQL> SELECT SEQUENCE#,APPLIED FROM V$ARCHIVED_LOG ORDER BY SEQUENCE#;
SEQUENCE#  APP
-----
8          NO
9          NO
10         NO
11         NO
4 rows selected.
```

See Also: *Oracle Data Guard Concepts and Administration Release 2 (9.2)* (part number A96653-02) for more information on Oracle Data Guard.

Synchronizing Baseline Installation with Standby Site

Once Oracle Data Guard has been set up between the production and standby sites, the procedure for synchronizing the two sites can be carried out. An initial synchronization should be done, before the production site is used, in order to obtain a baseline snapshot of the post-installation production site onto the standby site. This baseline can then be used to recover the production site configuration on the standby site if needed later.

In order to obtain a consistent point-in-time snapshot of the production site, the information stored in the Infrastructure database and the Oracle Application Server-related configuration files in the middle tier and Infrastructure hosts must be synchronized at the same time. Synchronization of the configuration files can be done by backing up the files and restoring them on the standby hosts using the Oracle Application Server Backup and Recovery Tool. For the Infrastructure database, synchronization is done using Oracle Data Guard by shipping the archive logs to the standby Infrastructure and applying these logs in coordination with the restoration of the configuration files.

The sequence of steps for the baseline synchronization (which can also be used for future synchronizations) are:

- Shipping Infrastructure Database Archive Logs
- Backing Up Configuration Files (Infrastructure and Middle Tier)
- Restoring Configuration Files (Infrastructure and Middle Tier)
- Restoring the Infrastructure Database - Applying Log Files

These steps are detailed in the following two main sections.

Backing Up Production Site

The main strategy and approach to synchronizing configuration information between the production and standby sites is to synchronize the backup of Infrastructure and middle tier configuration files with the application of log information on the standby Infrastructure database.

For Oracle Application Server, not all the configuration information is in the Infrastructure database. The backup of the database files needs to be kept synchronized with the backup of the middle tier and Infrastructure configuration files. Due to this, log-apply services should not be enabled on the standby database. The log files from the production Infrastructure are shipped to the standby Infrastructure but are not applied.

The backup process of the production site involves backing up the configuration files in the middle tier and Infrastructure nodes. Additionally, the archive logs for the Infrastructure database are shipped to the standby site.

The procedures to perform the backups and the log ship are discussed in the following sections:

- Shipping Infrastructure Database Archive Logs
- Backing Up Configuration Files (Infrastructure and Middle Tier)

IMPORTANT: Ensure that no configuration changes are going to be made to the Oracle Application Server system (underlying configuration files and Infrastructure database) as you perform the steps in this section.

Note: At the minimum, the backup and restoration steps discussed in this section and the "Restoring to Standby Site" section should be performed whenever there is any administration change in the production site (inclusive of changes to the Infrastructure database and configuration files on the middle tier and Infrastructure nodes). On top of that, scheduled regular backups and restorations should also be done (for example, on a daily or twice weekly basis). See the *Oracle Application Server 10g Administrator's Guide* for more backup and restore procedures.

Shipping Infrastructure Database Archive Logs

After installing the OracleAS Disaster Recovery solution, Oracle Data Guard should have been installed in both the production and standby databases. The steps for shipping the archive logs from the production Infrastructure database to the standby Infrastructure database involve configuring Oracle Data Guard and executing several commands for both the production and standby databases. Execute the following steps to ship the logs for the Infrastructure database:

1. If not disabled already, disable log-apply services by running the following SQLPLUS statement on the standby host:

```
SQL> alter database recover managed standby database cancel;
```

2. Run the following command to perform a log switch on the production Infrastructure database. This ensures that the latest log file is shipped to the standby Infrastructure database

```
SQL> alter system switch logfile;
```

3. In normal operation of the production site, the production database frequently ships log files to the standby database but are not applied. At the standby site, you want to apply the logs that are consistent up to the same time that the production site's configuration files are backed up. The following SQL statement encapsulates all Infrastructure database changes into the latest log and allows the Oracle Data Guard transport services to transport this log to the Infrastructure in the standby site:

```
SQL> select first_change# from v$log where status='CURRENT';
```

A SCN or sequence number is returned, which essentially represents the timestamp of the transported log.

4. Note down the SCN number as you will need this for the restoration of the production database changes on the standby site.

Continue to the next section to back up the configuration files on the middle tier host(s) and Infrastructure host.

Backing Up Configuration Files (Infrastructure and Middle Tier)

Use the instructions in this section to back up the configuration files. The instructions require the use of the Oracle Application Server Backup and Recovery Tool. They assume you have installed and configured the tool on each OracleAS installation (middle tier and Infrastructure) as it needs to be customized for each installation. Refer to *Oracle Application Server 10g Administrator's Guide* for more details about that tool, including installation and configuration instructions.

For each middle tier and Infrastructure installation, perform the following steps (the same instructions can be used for the middle tier and Infrastructure configuration files):

1. After performing the installation and configuration steps detailed in the *Oracle Application Server 10g Administrator's Guide*, for the Oracle Application Server Backup and Recovery Tool, the variables `oracle_home`, `log_path`, and `config_backup_path` in the tool's configuration file, `config.inp`, should have the appropriate values. Also, the following command for the tool should have been run to complete the configuration:

```
perl bkp_restore.pl -m configure_nodb
```

In Windows, the Perl executable can be found in `<ORACLE_HOME>\perl\<perl_version>\bin\MSWin32-x86`.

If you have not completed these tasks, do so before continuing with the ensuing steps.

2. Execute the following command to back up the configuration files from the current installation:

```
perl bkp_restore.pl -v -m backup_config
```

This command creates a directory in the location specified by the `config_backup_path` variable specified in the `config.inp` file. The directory name includes the time of the backup. For example: `config_bkp_2003-09-10_13-21`.

3. A log of the backup is also generated in the location specified by the `log_path` variable in the `config.inp` file. Check the log files for any errors that may have occurred during the backup process.
4. Copy the Backup and Recovery Tool's directory structure and contents from the current node to its equivalent in the standby site. Ensure that the path structure on the standby node is identical to that on the current node.
5. Copy the backup directory (as defined by `config_backup_path`) from the current node to its equivalent in the standby site. Ensure that the path structure on the standby node is identical to that on the current node.
6. Repeat the steps above for each Oracle Application Server installation in the production site (middle tier and Infrastructure).

Note: There are two important items that should be maintained consistently between the production and standby sites. The directory names should be the same and the correlation of SCN to a given backup directory should be noted at both sites in administration procedures.

Restoring to Standby Site

After backing up the configuration files from the middle tier Oracle Application Server instances and Infrastructure together with the Infrastructure database, restore the files and database in the standby site using the instructions in this section, which consists of the following sub-sections:

- Restoring Configuration Files (Infrastructure and Middle Tier)
- Restoring the Infrastructure Database - Applying Log Files

Restoring Configuration Files (Infrastructure and Middle Tier)

Restoring the backed up files from the production site requires the Oracle Application Server Backup and Recovery Tool that was used for the backup. The instructions in this section assume you have installed and configured the tool on each OracleAS installation in the standby site, both in the middle tier and Infrastructure nodes. Refer to *Oracle Application Server 10g Administrator's Guide* for instructions on how to install the tool.

For each middle tier and Infrastructure installation in the standby site, perform the following steps (the same instructions can be used for the middle tier and Infrastructure configuration files):

1. Check that the Backup and Recovery Tool's directory structure and the backup directory from the equivalent installation in the production site are present in the current node.
2. Stop the Oracle Application Server instances and their processes so that no modification of configuration files can occur during the restoration process. Use the following OPMN command:

In UNIX:

```
<ORACLE_HOME>/opmn/bin/opmnctl stopall
```

In Windows:

```
<ORACLE_HOME>\opmn\bin\opmnctl stopall
```

Check that all relevant processes are no longer running. In UNIX, use the following command:

```
ps -ef | grep <ORACLE_HOME>
```

In Windows, press `<ctrl><alt>` to bring up the Task Manager and verify that the processes have stopped.

3. Configure the backup utility for the Oracle home.

This can be accomplished either by configuring the Backup and Recovery Tool for the Oracle home or copying the backup configuration file, `config.inp`, from the production site peer. Below is an example of running the Backup and Recovery Tool configuration option:

```
perl bkp_restore.pl -v -m configure_nodb
```

In Windows, the Perl executable can be found in `<ORACLE_HOME>\perl\<perl_version>\bin\MSWin32-x86`.

4. Execute the following command to view a listing of the valid configuration backup locations:

```
perl bkp_restore.pl -v -m restore_config
```

- Restore the configuration files using the following command:

```
perl bkp_restore.pl -v -m restore_config -t <backup_directory>
```

where *<backup_directory>* is the name of the directory with the backup files that was copied from the production site. For example, this could be `config_bkp_2003-09-10_13-21`.

- Check the log file specified in `config.inp` for any errors that may have occurred during the restoration process.
- Repeat the steps above for each Oracle Application Server installation in the production site (middle tier and Infrastructure).

Restoring the Infrastructure Database - Applying Log Files

During the backup phase, you executed several instructions to ship the database log files from the production site to the standby site up to the SCN number that you recorded as per instructed. To restore the standby database to that SCN number, apply the log files to the standby Infrastructure database using the following SQLPLUS statement:

```
SQL> alter database recover automatic from '/private/oracle/oracleas/standby/' standby
database until change <SCN>;
```

(In Windows, substitute the path shown above appropriately.)

With this command executed and the instructions to restore the configuration files completed on each middle tier and Infrastructure installation, the standby site is now synchronized with the production site. However, there are two common problems that can occur during the application of the log files: errors caused by the incorrect specification of the path and gaps in the log files that have been transported to the standby site.

The following are methods of resolving these problems:

- Find the correct log path.

On the standby Infrastructure database, try to determine location and number of received archive logs using the following SQLPLUS statement:

```
SQL> show parameter standby_archive_dest
```

NAME	TYPE	VALUE
standby_archive_dest	string	/private/oracle/oracleas/standby/

(The above example shows the UNIX path. The Windows equivalent path is shown in Windows systems.)

- Use the log path obtained from the previous step to ensure that all log files have been transported.

At the standby Infrastructure database, perform the following:

```
standby> cd /private/oracle/oracleas/standby
standby> ls
1_13.dbf 1_14.dbf 1_15.dbf 1_16.dbf 1_17.dbf 1_18.dbf 1_19.dbf
```

(In Windows, use the command `cd` to change to the appropriate directory and `dir` to view the directory contents.)

At the production Infrastructure database, execute the following SQLPLUS statement:

```
SQL> show parameter log_archive_dest_1

NAME                                TYPE        VALUE
-----                                -
log_archive_dest_1                   string      LOCATION=/private/oracle/oracleas/oradata
                                         MANDATORY
log_archive_dest_10                  string
```

(The above example shows the UNIX path. The Windows equivalent path is shown in Windows systems.)

- Using the path specified in step 1, note the number and sequence of the log files. For example:

```
production> cd /private/oracle/oracleas/oradata
production> ls
1_10.dbf 1_12.dbf 1_14.dbf 1_16.dbf 1_18.dbf asdb
1_11.dbf 1_13.dbf 1_15.dbf 1_17.dbf 1_19.dbf
```

(In Windows, use the command `cd` to change to the appropriate directory and `dir` to view the directory contents.)

In the above example, note the discrepancy where the standby Infrastructure is missing files `1_10.dbf` through `1_12.dbf`. Since this gap in the log files happened in the past, it could be due to a problem with the historic setup involving the network used for the log transport. This problem has obviously been corrected and subsequent logs have been shipped. To correct the problem, copy (FTP) the log files to the corresponding directory on the standby Infrastructure database host and re-attempt the SQLPLUS recovery statement shown earlier in this section.

Scheduled Outages

Scheduled outages are planned outages. They are required for regular maintenance of the technology infrastructure supporting the business applications and include tasks such as hardware maintenance, repair and upgrades, software upgrades and patching, application changes and patching, and changes to improve performance and manageability of systems. Scheduled outages can occur either for the production or standby site. Descriptions of scheduled outages that impact the production or standby site are:

- Site-wide maintenance

The entire site where the current production resides is unavailable. Examples of site-wide maintenance are scheduled power outages, site maintenance, and regular planned switchovers.
- OracleAS Cold Failover Cluster cluster-wide maintenance

This is scheduled downtime of the OracleAS Cold Failover Cluster for hardware maintenance. The scope of this downtime is the whole hardware cluster. Examples of cluster-wide maintenance are repair of the cluster interconnect and upgrade of the cluster management software.
- Testing and validating the standby site as a means to test disaster recovery readiness.

For scheduled outages, a site switchover has to be performed, which is explained in the following section.

Site Switchover Operations

A site switchover is performed for planned outages of the production site. Both the production and standby sites have to be available during the switchover. The application of the database redo logs is synchronized to match the backup and restoration of the configuration files for the middle tier and Infrastructure installations.

During site switchover, considerations to avoid long periods of cached DNS information have to be made. Modifications to the site's DNS information, specifically time-to-live (TTL), have to be performed. Refer to "Manually Changing DNS Names" on page 6-38 for instructions.

Note: All sessions to the production and standby databases need to be closed in order to perform the switchover operation. This requires that all middle tier and Infrastructure instances need to be shut down. Also, the CJQ0 and QMN0 database processes have sessions that need to be stopped.

To switchover from the production to standby site, perform the following:

1. Reduce the wide area DNS TTL value for the site.
2. On the production site, backup the middle tier and Infrastructure configuration files as described in the section "Backing Up Configuration Files (Infrastructure and Middle Tier)" on page 6-28.
3. On the standby site, restore the backed up configuration files as described in the section "Restoring Configuration Files (Infrastructure and Middle Tier)" on page 6-29.
4. Execute the following SQLPLUS statement to enable log apply services on the standby Infrastructure database so that all of the archive redo logs are applied:

```
SQL> alter database recover managed standby database disconnect from session;
```

5. Shut down all Oracle Application Server instances to close all sessions to the databases. Use the following command on all hosts:

In UNIX:

```
<ORACLE_HOME>/opmn/bin/opmnctl stopall
```

In Windows:

```
<ORACLE_HOME>\opmn\bin\opmnctl stopall
```

6. Stop the CJQ0 and QMN0 database processes as these also have open sessions to the database that need to be closed.

To stop the CJQ0 process, run the following query for the production and standby databases:

```
SQL> ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0;
```

To stop the QMN0 process, run the following query for the production and standby databases:

```
SQL> ALTER SYSTEM SET AQ_TM_PROCESSES=0;
```

(The changes effected by the above statements need not require a database restart.)

7. Perform the switchover steps for the Infrastructure database in each site.

On the production database, perform the following:

- a. Verify that it is possible to perform a switchover operation.

On the current production database, query the `SWITCHOVER_STATUS` column of the `V$DATABASE` fixed view on the production database to verify that it is possible to perform a switchover operation. For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----
TO STANDBY
1 row selected
```

The `TO STANDBY` value in the `SWITCHOVER_STATUS` column indicates that it is possible to switch the production database to the standby role. If the `TO STANDBY` value is not displayed, then verify that the Data Guard configuration is functioning correctly (for example, verify that all `LOG_ARCHIVE_DEST_n` parameter values are specified correctly).

- b. Initiate the switchover operation on the production database.

To transition the current production database to a physical standby database role, use the following `SQLPLUS` statement on the production database:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY;
```

After this statement completes, the production database is converted into a standby database. The current control file is backed up to the current `SQLPLUS` session trace file before the switchover operation. This makes it possible to reconstruct a current control file, if necessary.

- c. Shut down and restart the former production instance.

Shut down the former production instance and restart it without mounting the database:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP NOMOUNT;
SQL> ALTER SYSTEM SET STANDBY_ARCHIVE_DEST='/private/oracle/oracleas/standby/' SCOPE=BOTH;
SQL> ALTER SYSTEM SET STANDBY_FILE_MANAGEMENT='auto' SCOPE=BOTH;
```

(In Windows, substitute the path above appropriately.)

Mount the database as a physical standby database:

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

Create the standby archive destination directory. For example:

In UNIX:

```
mkdir /private/oracle/oracleas/standby/
```

In Windows, use Windows Explorer or the following command:

```
md <ORACLE_HOME>\standby\
```

At this point in the switchover process, both databases are configured as standby databases.

On the original standby database, perform the following:

- d. Verify the switchover status in the `V$DATABASE` view.

After you transition the production database to the physical standby role and the switchover notification is received by the standby databases in the configuration, you should verify if the switchover notification was processed by the original standby database by querying the `SWITCHOVER_STATUS` column of the `V$DATABASE` fixed view on the original standby database.

For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----
SWITCHOVER PENDING
1 row selected
```

The `SWITCHOVER PENDING` value of the `SWITCHOVER_STATUS` column indicates the standby database is about to switch from the standby role to the production role. If the `SWITCHOVER PENDING` value is not displayed and the `TO PRIMARY` value is displayed, this indicates all redo has been received and applied and the standby is now a candidate for switchover to a production role. Verify that the Data Guard configuration is functioning correctly (for example, verify that all `LOG_ARCHIVE_DEST_n` parameter values are specified correctly).

- e. Switch the original standby database to the production role.

You can switch a physical standby database from the standby role to the production role when the standby database instance is either mounted in managed recovery mode or open for read-only access. It must be mounted in one of these modes so that the production database switchover operation request can be coordinated.

The `SQL ALTER DATABASE` statement used to perform the switchover automatically creates online redo logs if they do not already exist. Use the following `SQLPLUS` statements on the physical standby database that you want to transition to the production role:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
SQL> SHUTDOWN IMMEDIATE;
SQL> CONNECT sys/<PASSWORD> AS SYSDBA
SQL> STARTUP MOUNT;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=asdb_remote' SCOPE=BOTH;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=enable SCOPE=BOTH;
SQL> ALTER DATABASE OPEN;
```

- f. Shut down and restart the new production database.

Shut down the original standby instance and restart it using the appropriate initialization parameters for the production role:

```
SQL> SHUTDOWN;
SQL> STARTUP;
```

The original physical standby database is now transitioned to the production database role.

- g. Begin sending redo data to the standby databases.

Issue the following statement on the new production database:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```


8. On the new production database, start the Oracle Application Server instances using the following command:
In UNIX:

```
<ORACLE_HOME>/opmn/bin/opmnctl startall
```


In Windows:

```
<ORACLE_HOME>\opmn\bin\opmnctl startall
```
9. Perform a wide area DNS switchover to direct requests to the new production site based on one of the options presented in the section "Wide Area DNS Operations" on page 6-37.
10. Adjust the wide area DNS TTL to an appropriate value.

Unplanned Outages

An unplanned outage that impacts either or both the production and standby sites can be one of the following:

- **Site-wide failure** The entire site where the current production resides is unavailable. Examples of site-wide outages are disasters at the production or standby site such as fire, flood, earthquake, or power outages.

Note: A site-wide or OracleAS Cold Failover Cluster cluster-wide unplanned outage on the standby site does impact availability; a production database outage is required to restore the standby outage.

- **Complete failure of the middle tier** The entire middle tier is not available. Either all nodes are down or the application server instances on all nodes are down. The last surviving node of the Oracle Application Server 10g Farm for a service is no longer available.
- **OracleAS Cold Failover Cluster cluster-wide failure.** The entire hardware cluster hosting the OracleAS Infrastructure is unavailable or crashes. This includes failure of nodes in the cluster as well as any other components that results in the hardware cluster and the Infrastructure on this site not being available.

Unplanned outages warrant the failover of the production site to the standby site. Configuration files restoration and an Oracle Data Guard failover operation are required. Failover restores the Oracle Application Server environment to the point of the last successful backup.

Site Failover Operations

A site failover is performed for unplanned outages for the production site. Failover operations require the restoration on the standby site of the last backup of the configuration files of all hosts and the synchronized application of equivalent point-in-time redo logs (using the correct SCN number) to the standby database.

To failover the production site to the standby site:

1. On the standby site, restore the most recently backed up configuration files as described in the section "Restoring Configuration Files (Infrastructure and Middle Tier)" on page 6-29.

2. Initiate the failover operation on the target physical standby database. Execute the following SQLPLUS statement:

```
SQL> alter database recover automatic from '/private/oracle/oracleas/standby/' standby
database until change <SCN>;
```

(In Windows, substitute the path above appropriately.)

3. Convert the physical standby database to the production role.

Once the statement in the previous step completes successfully, the standby database is recovered to a consistent level with the configuration files that were restored in step 1.

Execute the following statements to transform the standby database to the production role:

```
SQL> connect sys/internal as sysdba
SQL> select OPEN_MODE, STANDBY_MODE, DATABASE_ROLE from v$database;

OPEN_MODE  STANDBY_MOD  DATABASE_ROLE
-----
MOUNTED    UNPROTECTED  PHYSICAL STANDBY

SQL> alter database activate standby database;

Database altered.

SQL> alter database mount;

Database altered.

SQL> select OPEN_MODE, STANDBY_MODE, DATABASE_ROLE from v$database;

OPEN_MODE  STANDBY_MOD  DATABASE_ROLE
-----
MOUNTED    UNPROTECTED  PRIMARY

SQL> alter database open resetlogs;
alter database open resetlogs
*
ERROR at line 1:
ORA-01139: RESETLOGS option only valid after an incomplete database recovery
```

Note: The last statement, "alter database open resetlogs;", may generate an ORA-01139 message (as shown) depending on the completeness of the recovery command in step 2. The message appears if the database recovery is complete and can be ignored.

Also, after issuing the last statement, you can no longer use this database as a standby database and subsequent redo logs from the original production database cannot be applied.

4. Shut down and restart the new production database.

To complete the failover operation, you need to shut down the new production database and restart it in read/write mode using the proper traditional initialization parameter file (or server parameter file) for the production role:

```
SQL> SHUTDOWN IMMEDIATE;
```

```

SQL> STARTUP;
ORACLE instance started.

Total System Global Area 143427356 bytes
Fixed Size                 280348 bytes
Variable Size             92274688 bytes
Database Buffers         50331648 bytes
Redo Buffers              540672 bytes
Database mounted.
Database opened.
SQL> select OPEN_MODE, STANDBY_MODE, DATABASE_ROLE from v$database;

OPEN_MODE  STANDBY_MOD DATABASE_ROLE
-----
READ WRITE UNPROTECTED PRIMARY

```

5. Perform a wide area DNS switchover to direct requests to the new production site based on one of the options presented in the section "Wide Area DNS Operations" on page 6-37.

Setting Up the New Standby Database

After starting the new production database, a new standby site needs to be created. The steps for performing this are documented in this chapter starting from the section "Setting Up Oracle Data Guard" on page 6-15 to the section "Backing Up Configuration Files (Infrastructure and Middle Tier)" on page 6-28.

Once a new standby site has been established, a planned switchover may be performed to migrate production quality processing to the correct geographical site. Perform the steps in the section "Site Switchover Operations" on page 6-32.

Wide Area DNS Operations

In order for client requests to be directed to the entry point of a production site, DNS resolution is used. When a site switchover or failover is performed, client requests have to be redirected transparently to the new site playing the production role. To accomplish this redirection, the wide area DNS that resolves the requests to the production site has to be switched over to the standby site. The DNS switchover can be accomplished in one of the following two ways:

Note: A hardware load balancer is assumed to be front-ending each site. Check <http://metalink.oracle.com> for supported load balancers.

- Using a Wide Area Load Balancer
- Manually Changing DNS Names

Using a Wide Area Load Balancer

When a wide area load balancer (global traffic manager) is deployed in front of the production and standby sites, it provides fault detection services and performance-based routing redirection for the two sites. Additionally, the load balancer can provide authoritative DNS name server equivalent capabilities.

During normal operations, the wide area load balancer can be configured with the production site's load balancer name-to-IP mapping. When a DNS switchover is

required, this mapping in the wide area load balancer is changed to map to the standby site's load balancer IP. This allows requests to be directed to the standby site, which should have been brought up and now has the production role.

This method of DNS switchover works for both site switchover and failover. One advantage of using a wide area load balancer is that the time for a new name-to-IP mapping to take effect can be almost immediate. The downside is that an additional investment needs to be made for the wide area load balancer.

Manually Changing DNS Names

This method of DNS switchover involves the manual change of the name-to-IP mapping that is originally mapped to the IP address of the production site's load balancer. The mapping is changed to map to the IP address of the standby site's load balancer. Follow these instructions to perform the task:

1. Note the current time-to-live (TTL) value of the production site's load balancer mapping. This mapping is in the DNS cache and will be there until the TTL expires. For the purposes of discussion and providing an example, let's assume this TTL to be 3600 seconds.
2. Modify the TTL value to a short interval. For example, 60 seconds.
3. Wait one interval of the original TTL. This is the original TTL of 3600 seconds that we noted in step 1.
4. Ensure that the standby site is switched over to receive requests.
5. Modify the DNS mapping to resolve to the standby site's load balancer giving it the appropriate TTL value for normal operation (for example, 3600 seconds).

This method of DNS switchover works for planned site switchovers only. The TTL value set in step 2 should be a reasonable time period where client requests cannot be fulfilled. The modification of the TTL is effectively modifying the caching semantics of the address resolution from a long period of time to a short period. Due to the shortened caching period, an increase in DNS requests can be observed.

Setting Up a DNS Server

This appendix provides instructions on setting up a DNS server in UNIX. These instructions are applicable for setting up the site-specific DNS zones used for hostname resolution in the example in Figure 6-3 on page 6-12.

Note: The DNS setup information provided in this appendix is an example to aid in the understanding of OracleAS Disaster Recovery operations. It is generic to DNS, and other appropriate DNS documentation should be consulted for comprehensive DNS information.

For the discussion in this chapter, the DNS server that is set up creates and services a new DNS zone with the unique domain `oracleas`. Within the zone, this DNS server resolves all requests for the `oracleas` domain and forwards other requests to the overall wide area company DNS server(s).

On the UNIX host that will act as the DNS zone server, perform the following steps:

1. Create the name server configuration file `/var/named.conf`. Assuming the wide area company DNS server IP address is `123.1.15.245`, the contents of this file should be as follows:

```
options {
    directory "/var/named";
    forwarders {
        123.1.15.245;
    };
};

zone "." in {
    type hint;
    file "named.ca";
};

zone "oracleas" {
    type master;
    file "oracleas.zone";
};

zone "0.0.127.IN-ADDR.ARPA" {
    type master;
    file "127.zone";
};
```

-
2. Create the root hint file `/var/named/named.ca`, which has the following contents (123.1.2.117 is the IP of the zone DNS server):

```
.          999999   IN      NS      ourroot.private.
ourroot.private.  IN      A       123.1.2.117
```

3. Create the loopback address file `/var/named/127.zone`, which has the following contents (assume the zone DNS server's hostname is `aszone1`):

```
$ORIGIN    0.0.127.IN-ADDR.ARPA.
0.0.127.IN-ADDR.ARPA.  IN      SOA    aszone1.oracleas.  root.aszone1.oracleas.
(
    25          ; serial number
    900         ; refresh
    600         ; retry
    86400       ; expire
    3600        ) ; minimum TTL

0.0.127.IN-ADDR.ARPA.  IN      NS      aszone1.oracleas.
1                      IN      PTR     localhost.oracleas.
```

4. Create the zone data file `/var/named/oracleas.dns`, which has the following contents (values shown are applicable to the example of the production site in Figure 6–3):

```
;
; Database file oracleas.dns for oracleas zone.
; Zone version: 25
;
$ORIGIN oracleas.
oracleas.      IN      SOA    aszone1.oracleas.  root.aszone1.oracleas (
    25          ; serial number
    900         ; refresh
    600         ; retry
    86400       ; expire
    3600        ) ; minimum TTL

;
; Zone NS records
;
oracleas.      IN      NS      aszone1.oracleas.

;
; Zone records
;
localhost      IN      A       127.0.0.1

asmid1         IN      A       123.1.2.333
asmid2         IN      A       123.1.2.334
infra          IN      A       123.1.2.111
remoteinfra    IN      A       213.2.2.210
```

5. Run the following command to start the name server:

```
/sbin/in.named
```

6. On all the hosts in the domain that is serviced by this DNS server, edit the `domain` and `nameserver` settings in the file `/etc/resolv.conf` as follows (all previous `nameserver` settings should be removed; 123.1.2.117 is assumed to be the zone DNS server's IP address):

```
domain    oracleas  
nameserver 123.1.2.117
```



Index

A

AC4J, 2-21
afctl, 5-7
AJP
 port number, 4-19
 ports, 4-33
Application Server Console, 2-10, 2-16, 3-5
Application Server Control, 4-5, 4-12, 4-14, 4-18,
 4-26, 4-33, 5-2, 5-4, 5-5, 5-7
 stop, 5-3
archive logs, 6-1, 6-15, 6-26
 shipping, 6-27
authentication credentials, 3-1
automatic recovery, 2-6
availability, 2-8

B

backup and recovery, 1-3
Business Intelligence and Forms, 2-1, 3-3
Business Intelligence and Forms installation
 type, 6-4

C

centralized repository, 3-4
client certificates, 4-31
cloning, 2-29
clusterable, 2-10
clustering
 configure Web application state replication, 4-20
 configuring
 EJB application state replication, 4-21
 OC4J processes, 4-23
 configuring islands, 4-23
 EJB applications, 4-21
 managing
 application server instances in a cluster, 4-7
 OracleAS Clusters, 4-4
 OracleAS Cluster configuration, 4-4
 removing application server instance, 4-8
cluster-wide configuration, 2-9
cold backup, 2-27
component instance, 2-2
config.inp, 6-28

configuration cloning, 2-5
configuration files, 6-1, 6-5
 backup, 6-28
configuring SSL, 4-15

D

data maintenance, 1-3
data replication, 2-5
database provider, 2-23
database repository, 2-7
DCM, 2-5, 2-7, 2-9, 2-16, 3-1, 3-2, 3-3, 3-6, 4-17, 4-29,
 4-30, 4-31
 cloning, 2-29
 daemon, 2-17
 dcmctl, 2-10, 2-17
 file-based repository, 3-3
 ports, 4-33
dcmCache.xml, 4-16
dcmctl, 2-17, 4-5, 4-8, 4-9, 4-10, 4-11, 4-12, 4-13, 4-14,
 4-16, 4-18, 4-20, 4-26, 4-29, 5-12
Delegated Administration Services
 start, 5-2, 5-4
 stop, 5-2, 5-5
directory service, 3-4
disaster recovery, 2-28, 6-1
Distributed Configuration Management, see
 DCM, 2-10
DNS, 6-9, 6-11, 6-12
 mapping, 6-38
 switchover, 6-35, 6-37
DNS resolution, 6-3, 6-11

E

EJB
 client routing, 2-21
 cluster, 2-15
 replication, 2-14
 stateful session, 2-14
EJB application state replication, 4-21
EJB session, 2-5
emctl, 4-12
exporting configuration information, 4-13
external load balancer, 2-17

F

failover, 3-7, 6-5, 6-35
failure types, 1-3
file-based repository, 2-7, 3-3
 repository host, 2-7
firewall
 routing between, 4-30
Forms Listener Servlet, 2-25
Forms Runtime Engine, 2-25
Forms Servlet, 2-25

G

global services daemon, 5-3, 5-5

H

hardware cluster, 3-7
hostname
 logical, 6-3, 6-9
 physical, 6-2, 6-7, 6-10
 virtual, 6-3, 6-7, 6-9, 6-10
hostname resolution, 6-9
HTTP, 3-15, 4-27
HTTPS, 3-15, 4-27
human error, 1-3

I

identity management metadata, 3-2
importing configuration information, 4-13
installation type
 Business Intelligence and Forms, 2-1, 2-24, 3-3, 6-4
 J2EE and Web Cache, 2-1, 2-7, 3-3, 6-4
 Portal and Wireless, 2-1, 3-3, 6-4
instance-specific parameters, 2-9
intelligent routing, 2-5

J

J2EE, 2-21
J2EE 1.3, 2-1
J2EE and Web Cache, 2-1, 2-7, 3-3
J2EE and Web Cache installation type, 6-4
JAAS, 4-14
Java Object Cache, 2-5, 2-15
JDBC, 3-15
JDK keytool, 4-16
JMS, 2-21
 port, 4-33
 port number, 4-19
JNDI namespace
 replication, 2-15

K

Kerberos Security Tickets, 3-4
keystore, 4-16
keytool, 4-16

L

LDAP, 3-4, 3-15, 3-17
load balancer, 3-16, 5-5, 6-4
 external, 2-17
 hardware, 2-18, 2-26
load balancing
 mod_oc4j, 2-19, 4-25, 4-26
local affinity, 2-20
log apply, 6-26
logical hostname, 6-3, 6-9
logical IP, 3-7

M

manageability, 2-8
management metadata, 3-2
management service, 3-1
managing
 application server instances in a cluster, 4-7
 adding, 4-7
 removing, 4-8
 OracleAS Clusters, 4-4
manually configured, 2-7, 2-8
metadata
 identity management, 3-2
 management, 3-2
 product, 3-2
metric-based, 2-20
middle tier recovery, 2-27
mod_oc4j, 2-5, 2-12, 2-21, 2-22
 load balancing, 2-19, 2-20, 4-25, 4-26
mod_oc4j.conf, 4-26, 4-29
mod_oradav, 2-22
mod_osso, 2-4, 2-22
mod_plsql, 2-22
multicast, 2-15
multicast address, 4-20, 4-21

N

Netegrity Site Minder, 3-4
NIS, 6-9

O

OC4J, 2-7, 2-15, 3-5
 cluster-wide parameters, 4-18
 distributed caching, 2-15
 instance, 2-5, 2-10, 2-15, 4-19
 instance-specific parameters, 4-18, 4-23
 islands, 4-19
 number of processes, 4-19
 port numbers, 4-19
 island, 2-2, 2-5, 2-11, 2-12, 2-13, 2-14, 4-19, 4-23
 islands, 4-23
 processes, 4-23
 Java Object Cache, 2-5, 2-15
 Oracle Delegated Administration Services
 instance, 3-6
 port numbers, 4-24

- process, 2-12, 2-14, 2-16, 2-20, 4-18, 4-19, 4-23, 4-25
- processes
 - configuring, 4-23
- OC4J load balancing, 2-19
- OC4J process, 4-31
- online backup, 2-27
- operating system cluster, 2-18
- OPMN, 2-6, 2-7, 2-12, 2-15, 2-19, 2-24, 3-6, 3-9, 3-15, 4-31, 5-2, 5-3, 5-5, 5-7
 - opmnctl, 2-16
 - Oracle Notification System, 2-16
 - Oracle Process Manager, 2-16
 - ports, 4-33
- opmnctl, 2-16, 4-26, 6-29, 6-32
- opmn.xml, 2-16
- Oracle HTTP Server, 2-15
- Oracle Application Server
 - Active Failover Cluster, 2-25, 2-26, 3-6, 3-14, 5-3, 5-6, 6-15
 - Active Failover Cluster Runtime Control Utility, 5-7
 - Backup and Recovery Tool, 5-8, 6-1, 6-16, 6-26, 6-28, 6-29
 - Certificate Authority, 2-4, 3-2
 - Cluster, 2-5, 2-6, 2-13, 2-14, 2-15, 2-16, 3-5, 4-19, 4-25, 4-26, 4-27
 - adding instances, 4-7
 - cluster-wide configuration, 2-9
 - manually configured, 2-7, 2-8
 - using database repository, 2-7, 4-10, 4-14
 - using file-based repository, 2-7, 4-5, 4-6, 4-9, 4-10, 4-11, 4-12, 4-13, 4-15, 4-17
 - Cold Failover Cluster, 2-25, 2-26, 3-6, 3-7, 5-1, 6-3, 6-5, 6-15, 6-31, 6-35
 - Containers for J2EE, 2-7
 - Disaster Recovery, 6-1
 - Discoverer, 2-1, 2-3, 2-25
 - load balancing, 2-25
 - process monitoring and restart, 2-25
 - Farm, 4-4, 4-5, 4-9, 4-10, 4-12, 4-14, 6-35
 - Form Services, 2-4
 - Forms Services, 2-1, 2-25
 - Infrastructure, 3-1, 4-14, 6-35
 - ports, 4-33
 - stopping, 5-2
 - instance, 2-2, 2-5, 2-11
 - Integration, 2-26, 3-3
 - InterConnect, 2-2, 2-26, 3-3
 - Java Object Cache, 4-13
 - Metadata Repository, 2-2, 2-5, 2-10, 2-28, 3-3, 3-5, 5-7, 6-4, 6-15
 - Personalization, 2-1, 2-4
 - Port Tunnel, 4-31
 - Portal, 2-22
 - DAD, 2-22
 - ports, 4-33
 - repository, 2-22
 - ports used, 4-34
 - ProcessConnect, 2-2, 2-26, 3-3
 - Reports Services, 2-1, 2-3, 2-15, 2-24
 - Single Sign-On, 2-4, 2-22, 3-2, 3-4, 3-6, 3-15, 4-14, 4-27
 - third party authentication, 3-4
 - using file-based repository, 4-15
 - Web Cache, 2-4, 2-5, 2-15, 2-18, 2-26, 4-27
 - Web Cache clusters, 2-11
 - Wireless, 2-23
- Oracle Data Guard, 1-3, 6-1, 6-5, 6-13, 6-14, 6-27, 6-35
- Oracle Delegated Administration Services, 2-22, 3-1, 3-4, 3-5, 3-6, 3-15
- Oracle Directory Integration and Provisioning, 2-22
- Oracle Directory Manager, 3-4
- Oracle Enterprise Manager, 3-2, 3-9, 3-15
 - Application Server Console, 3-5
- Oracle HTTP Server, 2-2, 2-3, 2-4, 2-5, 2-7, 2-10, 2-12, 2-13, 2-16, 2-21, 2-23, 3-4, 3-6, 3-9, 3-15, 4-25, 4-26, 4-27, 4-30, 4-31, 5-3, 5-7
 - ports, 4-33
 - start, 5-4
 - stateful load balancing, 2-11
 - stateless load balancing, 2-11
 - stop, 5-5
- Oracle HTTP Sever
 - start, 5-2
- Oracle Identity Management, 2-2, 2-4, 2-22, 3-5, 3-6, 6-4
- Oracle Internet Directory, 2-22, 2-24, 3-1, 3-4, 3-15, 4-14, 5-7
 - start, 5-2, 5-4
 - stop, 5-3, 5-5
- Oracle Management Services, 3-6
- Oracle Net, 3-15
- Oracle Net listener, 3-6
- Oracle Notification System, 2-16
- Oracle Process Management and Notification, see OPMN, 2-15
- Oracle Process Manager, 2-16
- Oracle Workflow, 3-3
- Oracle9i, 3-1, 3-2
- OracleAS Active Failover Cluster, 1-3
- OracleAS Cluster
 - configuration, 4-4
- OracleAS Cold Failover Cluster, 1-3
- OracleAS Single Sign-On
 - configuring, 4-27
- OracleNet, 6-14
- ORASSO DAD, 2-22
- orion-ejb-jar.xml, 4-22
- ossoreg.jar, 4-27
- outage, 5-6

P

- physical hostname, 6-2, 6-7, 6-10
- physical standby database, 6-33
- port numbers, 3-13, 4-19, 4-24
- Portal and Wireless, 2-1, 3-3
- Portal and Wireless installation type, 6-4
- Portal Page Engine, 2-22

- process monitoring, 2-5, 2-15
- process restart, 2-27
- product metadata, 3-2
- product metadata service, 3-1
- production site, 6-2, 6-12
 - backup, 6-26
- Public Key Infrastructure, 3-4

R

- RAID disks, 4-13
- random, 2-20
- Real Application Clusters, 3-3
- redo logs, 6-35
- Reports Engine, 2-24
- Reports Server, 2-24
- Reports Servlet, 2-24
- repository host, 2-7, 4-9, 4-10, 4-11
 - initializing, 4-11
- repository ID, 4-11, 4-15
- RMI
 - port, 4-33
 - port number, 4-19
- round robin, 2-20

S

- scalability, 2-6, 2-8
- scheduled outages, 6-31
- SCN, 6-27, 6-30, 6-35
- secure shell port forwarding, 6-14
- security service, 3-1
- sequence number, 6-27, 6-30, 6-35
- serializable, 4-21
- session state replication, 2-12, 2-13
- shared storage, 3-8
- single point of failure, 2-6, 2-16
- single sign-on, 2-4, 3-1
- smart routing, 2-5
- SQLPLUS, 6-30, 6-32
- SSH tunneling, 6-14
- SSL, 4-16, 4-17, 4-31
- SSORegistrar, 4-27
- SSOSDK, 3-4
- standby site, 6-2, 6-3, 6-12
 - restoration, 6-29
- state replication
 - configuring for EJB applications, 4-21
 - configuring for Web applications, 4-20
 - JVM termination, 4-23
- state safe applications, 2-12
- stateful applications, 2-12
- stateful HTTP requests, 2-19
- stateful session EJB, 4-22, 4-23
- stateful session replication, 2-5
- stateless applications, 2-12
- stateless HTTP requests, 2-19
- static ports file, 6-14
- staticports.ini, 3-13, 6-14
- switchover, 6-5, 6-32, 6-33

- system maintenance, 1-3

T

- time-to-live, 6-38
- TNS Names, 6-13

U

- unplanned outages, 6-35

V

- virtual hostname, 3-8, 6-3, 6-7, 6-9, 6-10
- virtual IP, 3-7, 6-7
- volume management software, 5-1, 5-3

W

- Web application session state replication, 2-12, 2-13
- Web provider, 2-23
- web.xml, 4-21
- weighted routing, 2-20
- wide area network, 6-2

X

- X.509 certificate, 3-4