

Oracle9i Real Application Clusters

Administration

Release 1 (9.0.1)

June 2001

Part No. A89869-01

Oracle9i Real Application Clusters Administration, Release 1 (9.0.1)

Part No. A89869-01

Copyright © 1999, 2001, Oracle Corporation. All rights reserved.

Primary Author: Mark Bauer.

Primary Contributors: Lance Ashdown, Jack Cai, Jonathan Creighton, Raj Kumar, and Francisco Sanchez.

Contributors: David Austin, Sohan DeMel, Mitch Flatland, Carmen Frank, Jim Rawles, Dipak Saggi, Deborah Steiner, Klaus Thielen, and Steve Wertheimer.

Graphic Designer: Valarie Moore.

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; 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 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. Oracle Corporation does not warrant that this document is 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, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions 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 Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and SQL*Loader, Secure Network Services, SQL*Plus, Real Application Clusters, Oracle Call Interface, Oracle9i, Oracle8i, Oracle8, Oracle Parallel Server, Oracle Forms, Oracle TRACE, Oracle Expert, Oracle Enterprise Manager, Oracle Server Manager, Oracle Net, Net8, PL/SQL, and Pro*C are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments	xv
Preface.....	xvii
What's New in Real Application Clusters Administration?.....	xxvii
Part I Introduction to Administering Real Application Clusters	
1 Introduction to Real Application Clusters Administration	
Administering Real Application Clusters Databases.....	1-2
Parameter Management in Real Application Clusters	1-2
Administering Storage in Real Application Clusters.....	1-2
General Administration Issues.....	1-2
Oracle Enterprise Manager in Real Application Clusters	1-3
Backup and Recovery in Real Application Clusters.....	1-3
Scaling Your Real Application Clusters Environment	1-3
Troubleshooting	1-4
2 Parameter Management in Real Application Clusters Environments	
Administering the Server Parameter File in Real Application Clusters Databases	2-2
Backing Up the Server Parameter File.....	2-2
Setting the Server Parameter File Parameter Values for Real Application Clusters	2-2
Exporting the Server Parameter File For Backward Compatibility	2-4

Setting Parameters within the Server Parameter File.....	2-4
Specifying Comments in the Server Parameter File	2-5
Using Client-Side Parameter Files.....	2-5
Client-Side Parameter File Naming Conventions.....	2-6
Purpose of <i>initsid.ora</i>	2-7
Purpose of <i>initdb_name.ora</i>	2-8
Placement and Use of IFILE Parameters in Instance-Specific Files.....	2-10
Using Multiple IFILE Entries in Client-Side Parameter Files.....	2-10
Parameter Settings in Real Application Clusters	2-11
Unique Identification of Instances with Parameters in the Sever Parameter File.....	2-11
Instance-Specific Parameter Settings Using <i>sid</i> in the Server Parameter File	2-11
Types of Parameters in Real Application Clusters.....	2-12
Multi-Valued Parameters	2-12
Parameters That Must Be Identical Across All Instances	2-12
Parameters That Must Be Unique Across All Instances.....	2-13
Multiple Instance Issues for Parameters	2-14
The Startup Process and Parameters in Real Application Clusters	2-18
Special Startup Considerations for Traditional Parameter File Use.....	2-18
Starting Two Instances on Remote Nodes	2-18
IFILE Use in Traditional Parameter File Scenarios.....	2-19
Setting Instance Numbers.....	2-19

3 Administering Storage Components in Real Application Clusters

File Management Issues in Real Application Clusters	3-2
Adding Data Files.....	3-2
Automatic Undo Management In Real Application Clusters	3-2
Using Automatic Undo Management	3-2
Switching Undo Tablespaces	3-3
System Rollback Segment.....	3-4
Overriding Automatic Undo Management by Using Rollback Segment Undo.....	3-4
Manually Creating Rollback Segments	3-5
Public and Private Rollback Segments	3-5
Using Redo Log Files.....	3-7
Managing Trace Files and the Alert Files.....	3-9
Associating Instances, Users, and Resources with Free List Groups.....	3-9

Associating Instances with Free Lists	3-10
Assignment of New Instances to Existing Free List Groups.....	3-10
FREELIST GROUPS and MAXINSTANCES	3-10
Associating User Processes with Free Lists	3-11
SQL Options for Managing Free Space.....	3-11
Preallocating Extents to Free List Groups	3-11
The Sequence Number Generator	3-12
The CREATE SEQUENCE Statement.....	3-12
The CACHE Option	3-12
The ORDER Option.....	3-13

4 Administering Real Application Clusters Databases with Server Control, SQL, and SQL *Plus

Using SRVCTL to Administer Real Application Clusters Instances.....	4-2
Administrative Tasks That You Can Accomplish with SRVCTL.....	4-2
SRVCTL Cluster Database Tasks	4-2
SRCVTL Cluster Database Configuration Tasks	4-3
SRVCTL Command Syntax.....	4-3
srvctl start	4-3
srvctl stop.....	4-4
srvctl status	4-4
srvctl config	4-5
srvctl get env	4-5
Updating Your Configuration Information with SRVCTL	4-5
srvctl add db.....	4-6
srvctl add instance	4-6
srvctl delete instance	4-6
srvctl delete db	4-6
srvctl rename instance.....	4-7
srvctl move instance	4-7
srvctl set env	4-7
srvctl unset env	4-7
Importing and Exporting Raw Device Configuration with SRVCONFIG.....	4-8
Migrating Oracle8i Configurations to Oracle9i.....	4-8
Global Services Daemon (GSD)	4-9

UNIX GSD Implementations	4-9
Windows GSD Implementations.....	4-9
Administering the GSD on Windows Platforms	4-9
Administering Real Application Clusters Databases Using SQL and SQL*Plus.....	4-10
Starting Databases in Cluster Mode on UNIX.....	4-10
Starting Databases in Cluster Mode on Windows NT and Windows 2000	4-11
Using RETRY to Mount a Database in Cluster Mode	4-12
Setting and Connecting to Instances.....	4-13
The SET INSTANCE and SHOW INSTANCE Commands.....	4-14
The CONNECT Command	4-14
Verifying That Instances are Running	4-15
Shutting Down Real Application Clusters Instances	4-15
Quiescing A Real Application Clusters Database.....	4-16
How SQL and SQL*Plus Commands Affect Instances	4-17

Part II Using Oracle Enterprise Manager to Administer Real Application Clusters

5 Administering Real Application Clusters Databases with Oracle Enterprise Manager

Overview of Oracle Enterprise Manager Administration	5-2
Starting the Console	5-2
Displaying Objects in the Navigator Pane	5-2
Database-Specific File Structure	5-3
Instance-Specific File Structure.....	5-4
Configuration.....	5-4
Stored Configurations.....	5-4
Sessions	5-4
Locks.....	5-4
Resource Plans	5-4
Resource Plan Schedule	5-4
Using the Cluster Database Right-Mouse Menu.....	5-7
Starting a Cluster Database.....	5-8
Shutting Down a Cluster Database.....	5-9
Viewing the Cluster Database Operation Results.....	5-11

Status Details Tab	5-12
Output Tab	5-13
Viewing Cluster Database Status	5-14
General Tab	5-15
Status Details Tab	5-16
Creating a Job for a Cluster Database or Instance.....	5-18
Specifying Job Details	5-18
General Tab	5-19
Tasks Tab	5-21
Tasks for Cluster Database Destinations	5-21
Parameters Tab	5-21
Cluster Database Startup Task	5-21
Parameters for the Cluster Database Shutdown Task.....	5-23
Registering Cluster Database Events.....	5-24

Part III Backup and Recovery in Real Application Clusters

6 Configuring RMAN for Real Application Clusters

Configuring RMAN in Real Application Clusters: Overview	6-2
Configuring RMAN Control File Snapshots and Autobackups.....	6-2
Configuring the Snapshot Control File Location in RMAN.....	6-3
Configuring the Control File Autobackup Feature in RMAN	6-3
Selecting An Archived Log Configuration Scheme	6-4
RMAN Archive Log Configuration Schemes.....	6-5
Non-Shared Local Archive Logging.....	6-6
Configuring Initialization Parameters for Non-Shared Archiving Destinations.....	6-7
Backups Using Non-Shared Archive Logging.....	6-8
Restore and Recovery Using Non-Shared Archive Logging	6-9
Shared Read Local Archive Logging	6-10
One Remote Archive Logging	6-13
Configuring Initialization Parameters for One Remote Archive Logging.....	6-14
Backups for One Remote Archive Logging	6-15
No Cross-Mounting Archive Logging	6-16
Backups for No Cross-Mounting	6-16
Restore and Recovery for No Cross-Mounting.....	6-17

Cross-Mounting for Writing Archive Logging	6-17
Configuring Initialization Parameters for Cross-Mounting for Writing	6-17
Backups for Cross-Mounting for Writing	6-19
Restore and Recovery for Cross-Mounting for Writing.....	6-19
Optimizing the Reading of Local Archivelog Destinations	6-20
Cross Shared Cross Writing with One Local and N Remote Archive Logging.....	6-21
Central NFS Directory for High Availability Archive Logging.....	6-23
Hybrid Archive Logging	6-24
Implementation Considerations of Archive Logging Schemes.....	6-25
Using NFS on UNIX Platforms	6-25
Soft-Mounted NFS Directories	6-25
NFS Implemented for High Availability.....	6-25
Using Shared Drives on Windows Platforms.....	6-26

7 Backing Up Real Application Clusters Databases

Backups in Real Application Clusters	7-2
Choosing a Backup Method and Backup Type	7-2
Open Database Backups	7-2
Closed Database Backups.....	7-3
Online Backups and Real Application Clusters	7-4
Archiving Considerations for Real Application Clusters Environments	7-4
Changing the Archiving Mode in Real Application Clusters	7-4
Monitoring the Archiving Process in Real Application Clusters	7-5
Archive File Format and Destinations in Real Application Clusters	7-5
Backing Up the Archive Logs	7-6
Checkpoints and Log Switches	7-6
Forcing a Checkpoint	7-6
Forcing a Log Switch.....	7-7
Forcing a Log Switch on a Closed Thread	7-8
RMAN Backups in Real Application Clusters.....	7-8
Node Affinity Awareness.....	7-8
Performing an Open Backup Using RMAN	7-9
Backup Examples for Real Application Clusters Using Shared Directories.....	7-10
Using RMAN to Backup to Multiple Nodes with Several Channels	7-10
Avoiding the Thread Statement for Backing Up Archivelogs	7-10

Making Backups to Shared Archive Log Destinations	7-11
Backing Up Local Files from Each Node Using Non-Shared Archive Log Destinations ...	7-11
Backing Up All Files from One Node	7-12
Recovering Archive Logs from One Node.....	7-12
Restoring and Recovering Archive Logs from All Local Nodes Using Oracle Release 1 (9.0.1) 7-13	
Restoring and Recovering Archive Logs from All Local Nodes Using Oracle 8.1.5 Or Earlier. 7-13	

8 Database Recovery in Real Application Clusters

Recovery from Instance Failures.....	8-2
Single Node Failure.....	8-2
Multiple-Node Failures	8-3
Fast-Start Checkpointing and Fast-Start On-Demand Rollback.....	8-3
Access to Data Files for Instance Recovery	8-4
Steps of Oracle Instance Recovery	8-4
Recovery from Media Failures	8-5
Complete Media Recovery	8-6
Block Media Recovery	8-7
Incomplete Media Recovery	8-7
Restoring and Recovering Redo Log Files.....	8-8
Recovery Using RMAN.....	8-8
Recovery Using Operating System Utilities	8-8
Disaster Recovery	8-9
Disaster Recovery Using RMAN.....	8-9
Parallel Recovery in Real Application Clusters.....	8-11
Parallel Recovery in Real Application Clusters	8-12
Parallel Recovery Using RMAN.....	8-12
Parallel Recovery Using Operating System Utilities.....	8-13
Setting the RECOVERY_ PARALLELISM Parameter.....	8-13
Specifying RECOVER Statement Options	8-13
Fast-Start Parallel Rollback in Real Application Clusters.....	8-13
RMAN Recovery Examples for Real Application Clusters	8-14
Recovery to Shared Archive Log Destinations.....	8-14
Restoring Archive Logs with RMAN	8-14

Part IV Scaling Your Real Application Clusters Environment

9 Adding Nodes and Instances and Deleting Instances in Real Application Clusters

Adding Nodes to a Cluster	9-2
Overview of Procedures for Adding Nodes	9-2
Adding a Node at the Clusterware Layer	9-3
Adding a Node at the Clusterware Layer on UNIX	9-3
Connect Node to Cluster	9-4
Install Vendor Cluster Software	9-4
Create Raw Devices	9-4
Adding a Node at the Clusterware Layer on Windows NT and Windows 2000	9-4
Connect Node to Cluster	9-5
Install Cluster Software	9-5
Create Disk Partitions	9-7
Create Links to Disk Partitions	9-7
Adding a Node at the Oracle Layer	9-9
Deleting Instances	9-19

Part V Reference

A Troubleshooting

Using Trace Files	B-1
Background Thread Trace Files	B-1
User Thread Trace Files	B-2
Alert File	B-2
Error Call Trace Stack	B-3
Contacting Oracle Support Services	B-4
Severe Errors	B-4

Glossary

List of Figures

2-1	Traditional Instance-Specific Initialization Files.....	2-6
2-2	Common Initialization File Example.....	2-8
3-1	Threads of Redo.....	3-7
5-1	Database Subfolders.....	5-5
5-2	Cluster Database Instances Folder.....	5-6
5-3	Right Mouse Menu.....	5-7
5-4	Status Details Tab.....	5-12
5-5	Successful Shutdown Results.....	5-13
5-6	Successful Shutdown Results.....	5-14
5-7	General Tab.....	5-16
5-8	Status Details Tab.....	5-17
5-9	General Tab.....	5-20
5-10	Shutdown Cluster Database Parameters.....	5-23
5-11	Event Management Tests Menu.....	5-25
6-1	Non-Shared Local Archiving Configuration Scheme.....	6-6
6-2	Shared Read Local Archiving Configuration Scheme.....	6-10
6-3	One Remote Archive Logging Scheme.....	6-13
6-4	Cross Shared Cross Writing, One Local and N Remote Archive Logging.....	6-21
6-5	Central NFS Directory Configuration Scheme.....	6-23
6-6	Hybrid Scheme Example.....	6-24
8-1	Steps of Oracle Instance Recovery.....	8-4
9-1	DBCA Welcome Page for Real Application Clusters.....	9-11
9-2	Operations Page.....	9-12
9-3	Instance Management Page.....	9-13
9-4	List of Databases.....	9-14
9-5	List of Instances.....	9-15
9-6	Adding an Instance.....	9-16
9-7	Instance Storage Page.....	9-17
9-8	DBCA Summary Dialog.....	9-18
9-9	DBCA Instance Management Operation Selection Page.....	9-19
9-10	List of Databases.....	9-20
9-11	List of Instances.....	9-21
9-12	DBCA Warning Dialog for Selecting the Local Instance.....	9-22
9-13	DBCA Summary Dialog.....	9-22

List of Tables

2-1	Naming Conventions and Descriptions for Optional Parameter Files.....	2-7
2-2	Example sids and Instance Names	2-11
2-3	Parameters That Must Be Identical on All Instances.....	2-12
2-4	Initialization Parameter Notes for Multiple Instances.....	2-14
4-1	Descriptions of V\$ACTIVE_INSTANCES Columns.....	4-15
4-2	How SQL*Plus Commands Apply to Instances	4-17
5-1	Cluster Database Right-Mouse Menu Functions Specific to Real Application Clusters	5-7
5-2	Cluster Database Startup Types.....	5-8
5-3	Shutdown Types.....	5-10
5-4	Possible Component States	5-13
5-5	Fields in Output Tab Form.....	5-16
5-6	Component States.....	5-17
5-7	Tabs Contained in the Create Job Property Sheet.....	5-18
5-8	General Tab Options	5-20
5-9	Parameters Tab for Startup	5-22
5-10	Parameters Tab for Shutdown.....	5-23
6-1	Cross Shared, Cross Writing, One Local N Remote Archiving Destinations Example	6-22
6-2	Central NFS Directory Archive Log Example.....	6-23
6-3	Example Letter-to-Node Assignment on Windows.....	6-26
6-4	Example Directory Hierarchy on Windows	6-27
6-5	Example Netshare Command Syntax	6-27
7-1	Archived Redo Log Filename Format Parameters	7-5
8-1	Database Status for Media Recovery	8-6
A-1	Background Thread Trace Files.....	B-2

Send Us Your Comments

Oracle9i Real Application Clusters Administration, Release 1 (9.0.1)

Part No. A89869-01

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

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

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

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7227 Attn: Server Technologies Documentation Manager
- Postal service:

Oracle Corporation
Server Technologies Documentation
500 Oracle Parkway, Mailstop 4op11
Redwood Shores, CA 94065
USA

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

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

Preface

Oracle9i Real Application Clusters Administration explains the Real Application Clusters-specific administrative tasks that supplement single **instance** administrative tasks. Information in this manual applies to Real Application Clusters as it runs on all operating systems. Where necessary, this manual refers to platform-specific documentation.

Note: Previous releases of Real Application Clusters are known as *Oracle Parallel Server*.

See Also: The *Oracle9i Real Application Clusters Documentation Online Roadmap* to help you use the online Real Application Clusters Documentation set

This preface contains these topics:

- [Audience](#)
- [Organization](#)
- [Related Documentation](#)
- [Conventions](#)
- [Documentation Accessibility](#)

Audience

Oracle9i Real Application Clusters Administration is written primarily for network or **Database Administrator (DBA)** responsible for the administration of Real Application Clusters.

To use this document you should first read the *Oracle9i Database Administrator's Guide* to become familiar with single-instance administrative procedures. You should then read *Oracle9i Real Application Clusters Concepts* for a conceptual understanding of Real Application Clusters processing. You should also have installed Real Application Clusters using the document *Oracle9i Real Application Clusters Installation and Configuration* and any platform-specific documentation.

Organization

This document contains the following five parts:

Part I: "Introduction to Administering Real Application Clusters"

Part One describes Real Application Clusters administration and your initial administrative tasks.

Chapter 1, "Introduction to Real Application Clusters Administration"

This chapter introduces the administrative tasks for Real Application Clusters software.

Chapter 2, "Parameter Management in Real Application Clusters Environments"

This chapter describes parameter files and Real Application Clusters-specific parameters.

Chapter 3, "Administering Storage Components in Real Application Clusters"

This chapter explains how to administer storage components in Real Application Clusters.

Chapter 4, "Administering Real Application Clusters Databases with Server Control, SQL, and SQL*Plus"

This chapter explains how to administer Real Application Clusters databases with Server Control, SQL, and SQL*Plus.

Part II: Using Oracle Enterprise Manager to Administer Real Application Clusters

Part Two describes how to use Oracle Enterprise Manager to administer Real Application Clusters databases.

Chapter 5, "Administering Real Application Clusters Databases with Oracle Enterprise Manager"

This chapter describes how to use Oracle Enterprise Manager to administer Real Application Clusters databases.

Part III: Backup and Recovery in Real Application Clusters

Part Three provides backup and recovery procedures for Real Application Clusters.

Chapter 6, "Configuring RMAN for Real Application Clusters"

This chapter describes how to configure RMAN for Real Application Clusters.

Chapter 7, "Backing Up Real Application Clusters Databases"

This chapter explains how to backup Real Application Clusters databases.

Chapter 8, "Database Recovery in Real Application Clusters"

This chapter explains database recovery in Real Application Clusters databases.

Part IV: Scaling Your Real Application Clusters Environment

Part Four provides information about adding nodes and instances to scale your Real Application Clusters environment.

Chapter 9, "Adding Nodes and Instances and Deleting Instances in Real Application Clusters"

This chapter explains how to add nodes and instances and how to delete instances in Real Application Clusters using the Oracle Universal Installer and the Oracle Database Configuration Assistant.

Part V: Reference

Part Five provides reference information for Real Application Clusters.

Appendix A, "Troubleshooting"

This chapter describes how to use trace files for troubleshooting Oracle installation issues. It also explains how to contact Oracle Customer Support.

Glossary

The glossary defines terms used in this book as well as terms relevant to the subject matter of this book.

Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Real Application Clusters Documentation Online Roadmap*
- *Oracle9i Real Application Clusters Concepts*
- *Oracle9i Real Application Clusters Installation and Configuration*
- *Oracle9i Real Application Clusters Deployment and Performance*
- *Oracle Real Application Clusters Guard Administration and Reference Guide*
- Your platform-specific Oracle Real Application Clusters Guard installation guide

Installation Guides

- Your platform-specific Oracle Real Application Clusters Guard installation guide
- *Oracle9i Installation Guide* for Compaq Tru64, Hewlett-Packard HPUX, IBM-AIX, Linux, and Sun Solaris-based systems
- *Oracle9i Database installation guide for Windows*
- *Oracle Diagnostics Pack Installation*

Operating System-Specific Administrative Guides

- *Oracle9i Administrator's Reference* for Compaq Tru64, Hewlett-Packard HPUX, IBM-AIX, Linux, and Sun Solaris-based systems
- *Oracle9i Database Administrator's Guide for Windows*

Oracle9i Real Application Clusters Management

- *Oracle9i Real Application Clusters Administration*

- *Oracle Enterprise Manager Administrator's Guide*
- *Getting Started with the Oracle Diagnostics Pack*

Generic Documentation

- *Oracle9i Database Concepts*
- *Oracle Net Services Administrator's Guide*
- *Oracle9i Database New Features*
- *Oracle9i Database Reference*

Many of the examples in this book use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle9i Sample Schemas* for information about how these schemas were created and how to use them.

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

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

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

<http://www.oraclebookshop.com/>

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

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

<http://technet.oracle.com/membership/index.htm>

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

<http://technet.oracle.com/docs/index.htm>

Conventions

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

- [Conventions in Text](#)
- [Conventions in Code Examples](#)

Conventions in Text

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

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an index-organized table .
<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.

Convention	Meaning	Example
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter <code>sqlplus</code> to open SQL*Plus. The password is specified in the <code>orapwd</code> file. Back up the data files and control files in the <code>/disk1/oracle/dbs</code> directory. The <code>department_id</code> , <code>department_name</code> , and <code>location_id</code> columns are in the <code>hr.departments</code> table. Set the <code>QUERY_REWRITE_ENABLED</code> initialization parameter to <code>true</code> . Connect as <code>oe</code> user. The <code>JRepUtil</code> class implements these methods.
lowercase monospace (fixed-width font) <i>italic</i>	Lowercase monospace italic font represents placeholders or variables.	You can specify the <i>parallel_clause</i> . Run <code>Uold_release.SQL</code> 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.	<code>DECIMAL (digits [, precision])</code>
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	<code>{ENABLE DISABLE}</code>
	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.	<code>{ENABLE DISABLE}</code> <code>[COMPRESS NOCOMPRESS]</code>

Convention	Meaning	Example
...	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 	<pre>CREATE TABLE ... AS subquery; SELECT col1, col2, ... , coln FROM employees;</pre>
.	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.	<pre>acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;</pre>
<i>Italics</i>	Italicized text indicates placeholders or variables for which you must supply particular values.	<pre>CONNECT SYSTEM/system_password DB_NAME = database_name</pre>
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.	<pre>SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;</pre>
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.	<pre>SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;</pre>

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. 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/>

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.

What's New in Real Application Clusters Administration?

This section describes the new administrative features of Oracle9i release 1 (9.0.1) of **Real Application Clusters**.

See Also: *Oracle9i Real Application Clusters Concepts* for a detailed explanation of this product's new features

The topic in this section is:

- [Oracle9i Release 1 \(9.0.1\) New Features for Administration of Real Application Clusters](#)

Oracle9i Release 1 (9.0.1) New Features for Administration of Real Application Clusters

The Oracle9i release 1 (9.0.1) Real Application Clusters features and enhancements described in this section comprise the overall effort to simplify the installation and configuration process.

- **Use the server parameter file to simplify parameter management in Real Application Clusters**

The **server parameter file** is easier to administer than the traditional client-side parameter files used in pre-9.0.1 releases.

See Also:

- [Chapter 2, "Parameter Management in Real Application Clusters Environments"](#) for information about administering parameters in Real Application Clusters and for details on using client-side parameter files
- *Oracle9i Real Application Clusters Installation and Configuration* for details on configuring the server parameter file in Real Application Clusters

- **Automatic undo management automatically controls undo space management**

Using **automatic undo management** simplifies your administrative duties by more efficiently using disk space to manage undo information.

See Also: [Chapter 3, "Administering Storage Components in Real Application Clusters"](#)

- **The Server Control Utility (SRVCTL) is fully documented; SRVCTL was formerly known as OPSCTL**

SRVCTL is fully documented for use with Real Application Clusters databases.

See Also: [Chapter 4, "Administering Real Application Clusters Databases with Server Control, SQL, and SQL*Plus"](#)

- **There are several usability enhancements to Recovery Manager in Real Application Clusters**

The procedures for configuring **Recovery Manager (RMAN)** in Real Application Clusters databases are simplified. For example, use the `CONFIGURE` command to save channel allocation settings; you do not have to manually allocate channels. You can also configure RMAN to automatically backup your **control file**.

You can execute a `BACKUP ARCHIVELOG ... DELETE ALL INPUT` command to delete the logs in all `LOG_ARCHIVE_DEST_n` locations. You can also execute commands such as `BACKUP`, `RESTORE`, and so on, directly from the RMAN prompt without having to use a `RUN { ... }` command.

See Also: Part III, for more information configuring RMAN and backup and recovery using RMAN in Real Application Clusters

- **The Oracle Database Configuration Assistant (DBCA) has Instance and Template Management features**

Use Instance Management to add or delete instances in a Real Application Clusters database. You can also use Template Management to manage database templates. For example, you can copy databases including the data files, or you can create a template from a database and move the template to another system and create a new database using that template.

See Also:

- [Chapter 9, "Adding Nodes and Instances and Deleting Instances in Real Application Clusters"](#) for information on adding and deleting instances
- *Oracle9i Real Application Clusters Installation and Configuration* for more information on Template Management in Real Application Clusters

- **Default user names expire as of this release**

Effective with this release, all default usernames *except* `SYS`, `SYSTEM`, and `SCOTT`, expire upon install. To use these names, you must explicitly unlock them.

Part I

Introduction to Administering Real Application Clusters

Part One provides information on parameter management, storage component administration, and general administrative procedures in Real Application Clusters. The chapters in Part One are:

- [Chapter 1, "Introduction to Real Application Clusters Administration"](#)
- [Chapter 2, "Parameter Management in Real Application Clusters Environments"](#)
- [Chapter 3, "Administering Storage Components in Real Application Clusters"](#)
- [Chapter 4, "Administering Real Application Clusters Databases with Server Control, SQL, and SQL*Plus"](#)

Introduction to Real Application Clusters Administration

This chapter provides an overview of the processes and components involved in administering **Real Application Clusters**. This chapter includes the following topics:

- [Administering Real Application Clusters Databases](#)
- [Parameter Management in Real Application Clusters](#)
- [Administering Storage in Real Application Clusters](#)
- [General Administration Issues](#)
- [Oracle Enterprise Manager in Real Application Clusters](#)
- [Backup and Recovery in Real Application Clusters](#)
- [Scaling Your Real Application Clusters Environment](#)
- [Troubleshooting](#)

Administering Real Application Clusters Databases

This book describes Real Application Clusters-specific administrative duties. These tasks are in addition to those described in the *Oracle9i Database Administrator's Guide* which you should read before you use this book to administer Real Application Clusters. The following is an overview of the Real Application Clusters-specific administrative issues.

Parameter Management in Real Application Clusters

You can establish **cluster**-wide parameter settings and specify instance-specific settings using the **server parameter file**. or you can use client-side parameter files to manage parameter settings.

There are also three types of parameters in Real Application Clusters. Some parameters can have different values for one or more instances while others must be either the same for all instances or unique among all instances. Parameter settings also have unique effects on **instance** startup within Real Application Clusters databases.

See Also: [Chapter 2, "Parameter Management in Real Application Clusters Environments"](#)

Administering Storage in Real Application Clusters

There are Real Application Clusters-specific storage management issues to consider such as data file administration and undo tablespace management. You must also understand **redo log file** management and the use of threads of redo in Real Application Clusters instances.

Storage management also involves associating instances, users, and resources with **free list groups**. Controlling **extent** allocation and using sequence number generators are other storage management issues you must consider.

See Also: [Chapter 3, "Administering Storage Components in Real Application Clusters"](#)

General Administration Issues

You can start and stop Real Application Clusters instances and databases using SQL, SQL*Plus, and command-line utilities. You can also set and connect to remote

instances and perform administrative operations by way of Oracle Net. However, SQL and SQL*Plus commands can have both *local* and *global* effects on instances.

You can also use the SRVCTL utility to perform many administrative duties in Real Application Clusters databases. You can also use SRVCTL to perform other tasks such as managing configuration information, and to delete, rename, and move instances.

See Also: [Chapter 4, "Administering Real Application Clusters Databases with Server Control, SQL, and SQL*Plus"](#)

Oracle Enterprise Manager in Real Application Clusters

You can use [Oracle Enterprise Manager](#) to perform most administrative duties in Real Application Clusters. Use Oracle Enterprise Manager to start and stop instances, monitor their performance, schedule jobs, and to perform backup and recovery operations.

See Also: [Chapter 5, "Administering Real Application Clusters Databases with Oracle Enterprise Manager"](#)

Backup and Recovery in Real Application Clusters

You can configure [Recovery Manager \(RMAN\)](#) for use in Real Application Clusters. You can also configure RMAN to automatically back up each [control file](#) and to automatically save channel settings.

You can use RMAN or operating system commands to perform both open and closed backups in Real Application Clusters. Your recovery options in Real Application Clusters include complete, incomplete, and block media recovery. You can also use parallel recovery.

See Also:

- [Chapter 6, "Configuring RMAN for Real Application Clusters"](#)
- [Chapter 7, "Backing Up Real Application Clusters Databases"](#)
- [Chapter 8, "Database Recovery in Real Application Clusters"](#)

Scaling Your Real Application Clusters Environment

To meet increased system demands or replace failed hardware, you can add [nodes](#) and instances and delete instances in Real Application Clusters. Use the [Oracle](#)

Universal Installer (OUI) and the **Oracle Database Configuration Assistant (DBCA)** to dynamically add nodes and to add and remove instances.

See Also: [Chapter 9, "Adding Nodes and Instances and Deleting Instances in Real Application Clusters"](#)

Troubleshooting

Oracle records processing information in various trace and log files. You can refer to these files for troubleshooting information and you should regularly copy these files to avoid overwriting them.

See Also: [Chapter A, "Troubleshooting"](#) for more information about these files

Parameter Management in Real Application Clusters Environments

Oracle uses parameter settings in the **server parameter file** to control database resources. You can also use the traditional client-side parameter files, however, Oracle Corporation recommends that you use the server parameter file. The server parameter file is easier to use

This chapter describes how to administer the server parameter and client-side parameter files. It also describes parameter use in **Real Application Clusters** and how parameters affect startup processing.

The topics in this chapter are:

- [Administering the Server Parameter File in Real Application Clusters Databases](#)
- [Using Client-Side Parameter Files](#)
- [Parameter Settings in Real Application Clusters](#)
- [Types of Parameters in Real Application Clusters](#)
- [Multiple Instance Issues for Parameters](#)
- [The Startup Process and Parameters in Real Application Clusters](#)

See Also: *Oracle9i Real Application Clusters Installation and Configuration* for information on creating and configuring the server parameter file in Real Application Clusters

Administering the Server Parameter File in Real Application Clusters Databases

This section describes administering the server parameter file and includes the following topics:

- [Backing Up the Server Parameter File](#)
- [Setting the Server Parameter File Parameter Values for Real Application Clusters](#)
- [Exporting the Server Parameter File For Backward Compatibility](#)

Backing Up the Server Parameter File

Oracle Corporation recommends that you regularly create copies of the server parameter file for recovery purposes. Do this using the `CREATE PFILE FROM SPFILE` statement.

You can also recover your database's server parameter file by starting up an **instance** using a client-side **initialization parameter file**. Then re-create the server parameter file using the `CREATE SPFILE` statement.

See Also: *Oracle9i SQL Reference* for more information about the `CREATE PFILE` and `CREATE SPFILE` statements

Setting the Server Parameter File Parameter Values for Real Application Clusters

Oracle automatically updates the values in the server parameter file for parameter settings that you change using **Oracle Enterprise Manager** or `ALTER SYSTEM SET` statements. In addition, the `ALTER SYSTEM RESET` syntax enables you to *undo* the effects of parameter settings in the server parameter file and parameters that you manually set.

For example, assume you start an instance with an the server parameter file containing:

```
proddb1.OPEN_CURSORS=1000
*.OPEN_CURSORS=500
```

Note: Sample settings only appear as text in these examples. However, the server parameter file is a binary file.

For the instance with *sid proddb1*, the parameter remains set to 1000 even though it is followed by a database-wide setting of 500. This prevents any database-wide alteration of the setting and gives the **Database Administrator (DBA)** of the instance *proddb1* complete control over parameter settings.

If another DBA executes the following:

```
ALTER SYSTEM SET OPEN_CURSORS=1500 sid='' SCOPE=MEMORY;
```

Then Oracle updates the setting on all instances except the instance with *sid proddb1*. If you later change the parameter setting by executing the following on the instance with *sid proddb1*, then the parameter begins accepting future ALTER SYSTEM values set by other instances:

```
ALTER SYSTEM RESET OPEN_CURSORS SCOPE=MEMORY sid='proddb1';
```

More precisely, if you execute the following on another instance, the instance with *sid proddb1* also assumes the new setting of 2000:

```
ALTER SYSTEM SET OPEN_CURSORS=2000 sid='' SCOPE=MEMORY;
```

If the server parameter file contains the entries:

```
proddb1.OPEN_CURSORS=1000  
*.OPEN_CURSORS=500
```

Executing:

```
ALTER SYSTEM RESET OPEN_CURSORS SCOPE=SPFILE sid='proddb1';
```

Makes Oracle disregard the first entry from the server parameter file.

To reset a parameter to its default value throughout your **cluster** database, enter the syntax:

```
ALTER SYSTEM RESET OPEN_CURSORS SCOPE=SPFILE sid='*';
```

Note: Not all ALTER SYSTEM statement changes and modifications are recorded in the server parameter file. Some ALTER SYSTEM statements only change *in-memory* parameter settings.

Exporting the Server Parameter File For Backward Compatibility

You can revert to a pre-release 1 (9.0.1) version of Oracle and convert from using the server parameter file to the traditional client-side parameter file type. Do this by using the `FROM` option of the `CREATE PFILE` statement. Export the contents of a server parameter file into a pre-release 1 (9.0.1) release parameter file using the following syntax:

```
CREATE PFILE[= 'pfile-name'] FROM SPFILE [= 'raw_device_name'];
```

This statement exports the contents of an the server parameter file into a pre-release 1 (9.0.1) release initialization parameter file named 'pfile-name'. If you do not specify a `PFILE` or an `SPFILE` file name, Oracle uses the platform-specific default `PFILE` and `SPFILE` names.

Oracle creates the `PFILE` as a text file on the server. This file contains all parameter settings of all instances. Entries for overrides appear as `sid.parameter=value`. The `PFILE` also contains any comments associated with the parameter. Comments appear in the same line as the parameter setting. You must move any `sid`-specific entries to an instance-specific parameter file and delete the `sid` specification. The `CREATE PFILE` statement requires `DBA` privileges.

You can execute the `CREATE PFILE` statement to:

- Create backups of the server parameter file. Note that **Recovery Manager (RMAN)** *does not* backup the server parameter file.
- Obtain a list of all parameter values currently used by an instance for diagnostic purposes. Do this using the `SHOW PARAMETER` command or by querying the `V$PARAMETER` table.
- Modify the server parameter file by exporting it, editing the output file, and then recreating it.

Setting Parameters within the Server Parameter File

Use the `sid` designator to set instance-specific parameter values in the server parameter file. For example, the following:

```
proddb1.OPEN_CURSORS = 1000  
proddb2.OPEN_CURSORS = 1500
```

Sets `OPEN_CURSORS` to 1000 for instance `proddb1`, and to 1500 for instance `proddb2`. These entries are recognized as entries for specific `sids` in a Real Application Clusters database. The value 1000 is applied to the parameter when

the instance is started up with *sid proddb1* and the value 1500 is applied to when the instance is started up with *sid proddb2*.

The entry:

```
DB_FILE_MULTIBLOCK_READ_COUNT=16
```

Sets the value of parameter `DB_FILE_MULTIBLOCK_READ_COUNT` to 16 for all instances. Parameter `DB_FILE_MULTIBLOCK_READ_COUNT` takes the value 16 for all instances because a *sid* of * is considered special and the value specified in the parameter setting is applied to all the *sids*.

Note that the server parameter file supports the pre-release 1 (9.0.1) syntax of:

```
parameter1 = value1
```

In addition, `parameter1` takes the value `value1` regardless of the *sid*. You can override parameters for specific *sids* as follows:

```
OPEN_CURSORS = 1000
proddb1.OPEN_CURSORS = 1500
```

In this case, `OPEN_CURSORS` takes the value 1000 for all the instances that have a *sid* other than *proddb1* and takes the value 1500 in the instance with *sid proddb1*.

Specifying Comments in the Server Parameter File

Specify comments with parameter settings on the same line with the parameter setting. For example, if `init.ora` contains the following lines:

```
# first comment
OPEN_CURSORS = value # second comment
```

The string `second comment` is associated with `OPEN_CURSORS`'s setting. Oracle displays this comment in the `V$PARAMETER` and `V$PARAMETER2` views. Oracle also displays comments such as the entry `#first comment` in the example.

Using Client-Side Parameter Files

You can use one or more client-side parameter files to manage parameter settings in Real Application Clusters. By default, if you do not specify `PFILE` in your `STARTUP` command, Oracle uses a server parameter file.

You can set global parameters within instance-specific parameter files. To do this, you must have identical parameter settings for global parameters in all of your

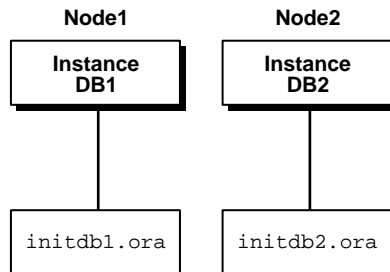
instance-specific parameter files. You can also maintain one file for global parameters and point to it with the **IFILE** parameter.

Note: If you use client-side parameter files, self-tuning parameter changes that Oracle makes are lost after shutdown. In addition, using client-side parameter files increases your parameter administration overhead.

Client-Side Parameter File Naming Conventions

Base your file names for the client-side parameter filenames on the *sid* of each instance and the **global database name**. For example, name each instance-specific parameter file **initsid.ora**, where *sid* is the system identifier of the instance. Name the common parameter file, **initdb_name.ora**, where *db_name* is the database name of your Real Application Clusters database as shown in [Figure 2-1](#).

Figure 2-1 Traditional Instance-Specific Initialization Files



The parameter file can contain both instance-specific and global parameter settings. If you include global parameter settings, the entries for these must be identical in each instance's file.

Oracle reads entries in this file beginning at the top of the file. When interpreting parameters, Oracle uses the last value for any parameters in this file that are duplicates. This is true for both instance-specific and global parameter settings.

[Table 2-1](#) describes the optional initialization parameter files:

Table 2–1 Naming Conventions and Descriptions for Optional Parameter Files

Initialization Parameter File/ Naming Convention	Description
Instance Specific File: <code>initsid.ora</code>	Each node's instance can have its own <code>initsid.ora</code> file, where <code>sid</code> is the Oracle system identifier (sid) of the instance. This file uniquely defines the instance-specific parameter settings. Use the <code>IFILE</code> parameter from within this file to call the <code>initdb_name.ora</code> file.
Global File: <code>initdb_name.ora</code>	If you do not include global parameter file settings in each instance-specific file, you must store common parameters in an <code>initdb_name.ora</code> file, where <code>db_name</code> is the database name. It lists the common database parameters shared by all instances. A copy of this file must reside on each instance in the database cluster. If you use a cluster file system (CFS), all instances that run from the same Oracle home share this file.

Purpose of `initsid.ora`

The `initsid.ora` file uses the `IFILE` parameter to point to the common file for common parameter settings. The `initsid.ora` file defines the following for each instance:

- Unique instance name
- Unique thread number and **instance number**
- Private **rollback segments** or undo tablespaces
- Local **listener** if not using default TCP/IP address for listener

The convention for deriving a `sid` is to use the value of the `DB_NAME` parameter in the `initdb_name.ora` file and the **thread ID**. For example, if the `DB_NAME` is `db`, and the first instance has a `thread ID` of 1, its `sid` is `db1`; the second instance uses the `sid` `db2` to identify its instance; and so on. This is the logic that the DBCA uses when it derives a `sid`. A `sid`, however, can have any value you choose.

[Example 2–1](#) and [Example 2–2](#) show the contents of `initsid.ora` files for two instances for each **node** numbered 1 and 2 respectively:

Example 2-1 *initdb1.ora*

```
ifile='C:\OracleSW\admin\db\pfile\initdb.ora'  
thread=1  
instance_name=db1  
instance_number=1
```

Example 2-2 *initdb2.ora*

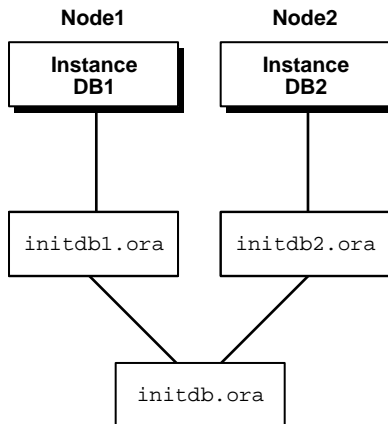
```
ifile='C:\OracleSW\admin\db\pfile\initdb.ora'  
thread=2  
instance_name=db2  
instance_number=2
```

See Also: *Oracle9i Database Reference* for complete parameter descriptions

Purpose of *initdb_name.ora*

Name the *initdb_name.ora* parameter file using the `IFILE` parameter setting in each *init_{sid}.ora* file as shown in [Figure 2-2](#).

Figure 2-2 *Common Initialization File Example*



All instances must use the same common file if you use the traditional parameter files.

Example 2-3 shows an `initdb_name.ora` file (`initdb.ora`) created for a **hybrid** or General Purpose database:

Example 2-3 *initdb.ora*

```
db_name="db"
db_domain=us.acme.com
cluster_database=true
service_names=db.us.acme.com
db_files=1024 # INITIAL
control_files=( "\\.\db_control1", "\\.\db_control2" )
open_cursors=100
db_file_multiblock_read_count=8 # INITIAL
db_block_buffers=13816 # INITIAL
shared_pool_size=19125248 # INITIAL
large_pool_size=18087936
java_pool_size=2097152
log_checkpoint_interval=10000
log_checkpoint_timeout=1800
processes=50 # INITIAL
parallel_max_servers=5 # SMALL
log_buffer=32768 # INITIAL
max_dump_file_size=10240 # limit trace file size to 5M each
global_names=true
oracle_trace_collection_name=""
background_dump_dest=C:\OracleSW\admin\db\bdump
user_dump_dest=C:\OracleSW\admin\db\udump
db_block_size=4096
remote_login_passwordfile=exclusive
os_authent_prefix=""
distributed_transactions=10
dispatchers="(protocol=TCP)(lis=listeners_db)"
compatible=9.0.1
sort_area_size=65536
sort_area_retained_size=65536
```

Note: `DB_BLOCK_BUFFERS` will be obsolete in a future Oracle release. If you set `DB_BLOCK_BUFFERS`, Oracle uses the value but also records a warning in your alert log file.

See Also: *Oracle9i Database Reference* for complete parameter descriptions

Placement and Use of IFILE Parameters in Instance-Specific Files

If you duplicate parameter entries in a parameter file, the last value specified in the file for the parameter overrides previous values. To ensure Oracle uses the correct common parameter values, place the `IFILE` parameter at the end of any instance-specific parameter files. Conversely, you can override common parameter values by placing the `IFILE` parameter before the instance-specific parameter setting.

Note: The DBCA places the `IFILE` parameter at the top of the parameter file.

Using Multiple IFILE Entries in Client-Side Parameter Files

You can specify `IFILE` more than once in your initial parameter file to include multiple global parameter files. However, do not accidentally reset a parameter value in subsequent common parameter files. Otherwise, each subsequent entry in the files specified by `IFILE` overrides previous values. For example, an instance-specific parameter file might include an `init_dbname.ora` file and separate parameter files for other parameter settings as in this example:

```
IFILE=INIT_CLUSTER.ORA
IFILE=INIT_LOG.ORA
IFILE=INIT_GC.ORA
LOG_ARCHIVE_START=FALSE
THREAD=3
UNDO_MANAGEMENT=AUTO
```

In this example, the value of `LOG_ARCHIVE_START=false` overrides any value specified in the parameter file `INIT_LOG.ORA` for this parameter. This is because the `LOG_ARCHIVE_START` parameter appears after the `IFILE` entry.

See Also:

- ["Parameters That Must Be Identical Across All Instances"](#) on page 2-12
- *Oracle9i Real Application Clusters Administration* for more information on starting and stopping instances

Parameter Settings in Real Application Clusters

As mentioned, some parameters must be identical across all instances in Real Application Clusters. Other parameters can have unique values within each instance.

Unique Identification of Instances with Parameters in the Server Parameter File

Each instance has several elements or components whose characteristics are controlled by parameter settings. Some of these are database objects and others are resource-like components that facilitate inter-instance processing. You uniquely identify these instance components using the `sid` designator in the server parameter file or by using parameter settings in `initsid.ora`.

See Also: *Oracle9i Database Reference* for more information on parameters

For example, [Table 2-2](#) shows the `sids` and `instance names` if the database name is `db` and the thread IDs for each instance are 1, 2, and 3 respectively:

Table 2-2 Example sids and Instance Names

thread id	sid	instance_name
1	db1	db1
2	db2	db2
3	db3	db3

Instance-Specific Parameter Settings Using `sid` in the Server Parameter File

Use instance-specific parameter settings to improve performance. For example, you can create System Global Areas (SGAs) of different sizes for each instance. You would do this if you had all OLTP users on one instance and all data warehouse users on another. In this case, you might decrease the value for the `SHARED_POOL_SIZE` and use parallel execution on the Data Warehouse instance. In addition, the OLTP instance would likely require a larger setting for `SHARED_POOL_SIZE`.

In the server parameter file, you would use the `sid` designator to identify these instance-specific settings. Also use the `sid` designator in the server parameter file when you create instances that specify:

- `INSTANCE_NAME`

- `THREAD`

Types of Parameters in Real Application Clusters

There are three types of initialization parameters in Real Application Clusters environments as described in this section. There are:

- [Multi-Valued Parameters](#)
- [Parameters That Must Be Identical Across All Instances](#)
- [Parameters That Must Be Unique Across All Instances](#)

See Also: *Oracle9i Database Reference* for details about other Oracle initialization parameters

Multi-Valued Parameters

You can configure some parameters to have different values for one or more instances. Parameters that can be multi-valued have a default value and a value for each instance that has modified the default setting.

Use the `ALTER SYSTEM SET` statement to set multiple values for such parameters. You can also use `ALTER SYSTEM SET` to define a global value that is effective for all instances. In addition, you can override these global values for specific instances.

Parameters That Must Be Identical Across All Instances

Certain initialization parameters that are critical at database creation or that affect certain database operations must have the same value for every instance in Real Application Clusters. Specify these parameter values in the common parameter file, or within each `init_dbname.ora` file on each instance. [Table 2-3](#) lists the parameters that must be identical on every instance.

Table 2-3 Parameters That Must Be Identical on All Instances

<code>CONTROL_FILES</code>	<code>DML_LOCKS</code> (Only if set to zero)
<code>DB_BLOCK_SIZE</code>	<code>LOG_ARCHIVE_DEST_n</code> (Optional)
<code>DB_FILES</code>	<code>MAX_COMMIT_PROPAGATION_DELAY</code>
<code>DB_NAME</code>	<code>SERVICE_NAMES</code>
<code>DB_DOMAIN</code>	<code>ACTIVE_INSTANCE_COUNT</code>
<code>ARCHIVE_LOG_TARGET</code>	<code>CLUSTER_INTERCONNECTS</code> (Solaris only)

Table 2–3 Parameters That Must Be Identical on All Instances

ROW_LOCKING	TRACE_ENABLED
GC_FILES_TO_LOCKS (Optional: Oracle automatically controls resource assignments, so ore-release 1 (9.0.1) locks are not needed)	

Parameters That Must Be Unique Across All Instances

If you use the `THREAD` or `ROLLBACK_SEGMENTS` parameters, Oracle Corporation recommends setting unique values for them using the `sid` identifier in the server parameter file. However, you *must* set a unique value for `INSTANCE_NUMBER` for each instance and you cannot use a default value.

- Oracle uses the `INSTANCE_NUMBER` parameter to distinguish among instances at startup. Oracle also uses `INSTANCE_NUMBER` to assign free space to instances using the `INSTANCE` option of the `ALLOCATE EXTENT` clause in the `ALTER TABLE` or `ALTER CLUSTER` statements.
- Specify the `THREAD` parameter so instances avoid the overhead of acquiring different thread numbers during startup and shutdown. Oracle uses the `THREAD` number to assign **redo log file** groups to specific instances. To simplify administration and avoid confusion, use the same number for the `THREAD` parameter that you used for the `INSTANCE_NUMBER` parameter.
- Specify `INSTANCE_NAME` to uniquely identify the instance. The default is the instance's `sid`. Oracle Corporation recommends that you use the `sid` for `INSTANCE_NAME`.
- Oracle acquires private rollback segments upon instance startup based on the rollback segment names you identify with the `ROLLBACK_SEGMENTS` initialization parameter. If you do not declare rollback segment names with this parameter for an instance, Oracle acquires public rollback segments for the instance.
- If you specify `UNDO_TABLESPACE` with **automatic undo management** enabled, set this parameter to a unique value for each instance.

Multiple Instance Issues for Parameters

[Table 2–4](#) alphabetically summarizes issues for parameters in Real Application Clusters.

Table 2–4 Initialization Parameter Notes for Multiple Instances

Parameter	Description and Comments
CLUSTER_ INTERCONNECTS (Solaris environments only)	<p>This parameter is only valid for Solaris environments.</p> <p>Oracle uses information from this parameter to distribute traffic among the various interfaces. You should set a value for CLUSTER_INTERCONNECTS when a single interconnect is insufficient to meet the bandwidth requirements of large Real Application Clusters databases.</p> <p>The syntax of the parameter is:</p> <pre>CLUSTER_INTERCONNECTS = <if1>:<if2>:...:<ifn></pre> <p>Where <ifn> is an IP address in standard dotted-decimal format, for example, 144.25.16.214. Subsequent platform implementations may specify interconnects with different syntaxes.</p> <p>Warning: When you set CLUSTER_INTERCONNECTS in Solaris configurations, the interconnect high availability features are not available. In other words, an interconnect failure that is normally unnoticeable would instead cause an Oracle cluster failure.</p>
CLUSTER_DATABASE	To enable a database to be started in Real Application Clusters mode, set this parameter to TRUE.
CLUSTER_DATABASE_ INSTANCES	Set this parameter to the number of instances in your Real Application Clusters environment. A proper setting for this parameter can improve memory use.
DB_NAME	If you set a value for DB_NAME in instance specific parameter files, it must be identical for all instances.

Table 2–4 Initialization Parameter Notes for Multiple Instances

Parameter	Description and Comments
DISPATCHER	<p>To enable a shared server configuration, set the DISPATCHERS parameter. The DISPATCHERS parameter may contain many attributes.</p> <p>Oracle Corporation recommends that you configure at least the PROTOCOL and LISTENER attributes. PROTOCOL specifies the network protocol for which the dispatcher generates a listening end point. LISTENER specifies an alias name for the listeners with which the PMON process registers dispatcher information. Set the alias to a name that is resolved through a naming method such as a tnsnames.ora file.</p> <p><i>Oracle Net Services Administrator's Guide</i> for complete information about configuring the DISPATCHER parameter and its attributes and for configuring the shared server</p>
DML_LOCKS	<p>Must be identical on all instances only if set to zero. The default value assumes an average of four tables referenced per transaction. For some systems, this value may not be enough. If you set the value of DML_LOCKS to 0, enqueues are disabled and performance is slightly increased. However, you cannot use DROP TABLE, CREATE INDEX, or explicit lock statements such as LOCK TABLE IN EXCLUSIVE MODE.</p>
INSTANCE_NAME	<p>If specified, this parameter must have unique values on all instances. In Real Application Clusters environments, multiple instances can be associated with a single database service. Clients can override connection load balancing by specifying a particular instance by which to connect to the database. INSTANCE_NAME specifies the unique name of this instance. Oracle Corporation recommends that you set INSTANCE_NAME equivalent to the sid.</p>

Table 2–4 Initialization Parameter Notes for Multiple Instances

Parameter	Description and Comments
LOG_ARCHIVE_FORMAT	<p>This parameter is applicable only if you are using the redo log in ARCHIVELOG mode. Use a text string and variables to specify the default filename format when archiving redo log files. The string generated from this format is appended to the string specified in the LOG_ARCHIVE_DEST_n parameter. You must include the thread number.</p> <p>The following variables can be used in the format:</p> <ul style="list-style-type: none"> ■ %s: log sequence number ■ %S: log sequence number, zero filled ■ %t: thread number ■ %T: thread number, zero filled <p>Using uppercase letters for the variables (for example, %S) causes the value to be fixed length and padded to the left with zeros. An example of specifying the archive redo log filename format is:</p> <ul style="list-style-type: none"> ■ LOG_ARCHIVE_FORMAT = "LOG%s_%t.ARC"
MAX_COMMIT_PROPAGATION_DELAY	<p>This is a Real Application Clusters-specific parameter. However, you should not change it except under a limited set of circumstances.</p> <p>This parameter specifies the maximum amount of time allowed before the system change number (SCN) held in the SGA of an instance is refreshed by the log writer process (LGWR). It determines whether the local SCN should be refreshed from the lock value when getting the snapshot SCN for a query. Units are in hundredths of seconds. Under unusual circumstances involving rapid updates and queries of the same data from different instances, the SCN might not be refreshed in a timely manner. Setting the parameter to zero causes the SCN to be refreshed immediately after a commit. The default value (700 hundredths of a second, or seven seconds) is an upper bound that allows the preferred existing high performance mechanism to remain in place.</p> <p>If you want commits to be seen immediately on remote instances, you may need to change the value of this parameter.</p>
NLS_* parameters For Oracle Globalization Support	<p>There are several Globalization Support parameters as described in <i>Oracle9i Database Reference</i> and <i>Oracle9i Globalization and National Language Support Guide</i>. You can set different values for different instances.</p>

Table 2–4 Initialization Parameter Notes for Multiple Instances

Parameter	Description and Comments
PROCESSES	Defaults for the SESSIONS and TRANSACTIONS parameters are derived directly or indirectly from the value of the PROCESSES parameter. Therefore, if you change the value of PROCESSES, you should evaluate whether to adjust the values of those derived parameters. If you do not use defaults, you may want to increase the values for some of the above parameters to allow for additional background processes.
RECOVERY_ PARALLELISM	To speed up roll forward or cache recovery processing, you may want to set this parameter to specify the number of processes to participate in instance or crash recovery. A value of zero or one indicates that recovery is to be performed serially by one process.
ROLLBACK_SEGMENTS (Use only in Rollback Managed Undo Mode) Note: Oracle Corporation strongly recommends that you use automatic undo management , not Rollback Managed Undo.	Use this parameter in manual rollback managed undo mode only to specify the private rollback segments for each instance by allocating one or more rollback segments by name to an instance. If you set this parameter, the instance acquires all of the rollback segments named in this parameter, even if the number of rollback segments exceeds the minimum number required by the instance, calculated from the ratio of: $\text{TRANSACTIONS} / \text{TRANSACTIONS_PER_ROLLBACK_SEGMENT}$
SESSIONS_PER_USER	Each instance maintains its own SESSIONS_PER_USER count. If SESSIONS_PER_USER is set to 1 for a user, the user can log on to the database more than once as long as each connection is from a different instance.
SPFILE	The value of SPFILE is the name of the current server parameter file in use. You can define the SPFILE parameter in a client-side PFILE to indicate the name of the server parameter file to use. When the server uses the default server parameter file, the server internally sets the value of SPFILE.
THREAD	If specified, this parameter must have unique values on all instances. THREAD is a Real Application Clusters parameter that specifies the number of the redo thread to be used by this instance. In Real Application Clusters, you can specify any available redo thread number, as long as that thread number is enabled and is not in use by another instance. A value of zero specifies that an instance can use any available, enabled public thread.

See Also: *Oracle9i Database Reference* for more information about these parameters and *Oracle9i Real Application Clusters Deployment and Performance* for a discussion of additional parameters for parallel execution in Real Application Clusters environments

The Startup Process and Parameters in Real Application Clusters

In Real Application Clusters, the first instance to start mounts the database. In addition, entries in the `alert.log` file of the first instance to start identifies that instance as the first one to start.

Note: To find your alert log file, use the search string `alert*.log`. You can usually find `alert.log` in the `background_dump_dest` directory.

See Also: *Oracle9i Real Application Clusters Administration* for more information about starting instances

Special Startup Considerations for Traditional Parameter File Use

If you use the traditional parameter files and a file for an instance contains a global parameter, its value must match the value set in other instances for that parameter. Otherwise, the instance cannot mount the database.

Starting Two Instances on Remote Nodes

Oracle Corporation recommends using the **Server Control (SRVCTL)** utility to start instances. You can also use SRVCTL for other administrative tasks as described under the heading "[Using SRVCTL to Administer Real Application Clusters Instances](#)" on page 4-2. The rest of this section describes using SQL*Plus to start instances.

To start multiple instances from a SQL*Plus session on one node by way of Oracle Net. For example, you can use a SQL*Plus session on a local node to start two instances on remote nodes using individual parameter files named `init_db1.ora` and `init_db2.ora`. Before connecting to the database, in SQL*Plus direct your commands to the first instance by entering:

```
SET INSTANCE DB1;
```

Connect to the first instance, start it, and disconnect from it by entering:

```
CONNECT / AS SYSDBA;
STARTUP PFILE=$ORACLE_HOME/dbs/initsid.ora
DISCONNECT;
```

Where the file *init*sid*.ora* contains an entry for an *spfile.ora* file's location on a raw device.

Re-direct commands to the second instance using the following syntax:

```
SET INSTANCE DB2;
```

Connect to and start the second instance by entering:

```
CONNECT / AS SYSDBA;
STARTUP PFILE=full pathINIT_DB2.ORA;
```

Here, DB1 and DB2 are *sids* for the two instances. These *sids* are defined with the *sid* entry in *tnsnames.ora*. An SPFILE parameter entry in the *init*sid*.ora* file specifies a location on the remote instance.

IFILE Use in Traditional Parameter File Scenarios

In the previous example, both parameter files can use the IFILE parameter to include values from an *init_dbname.ora* file.

Setting Instance Numbers

You must explicitly specify an instance number by using the `INSTANCE_NUMBER` parameter when you start it. You can do this with Real Application Clusters enabled or disabled. Set `INSTANCE_NUMBER` equal to the value of the instance's `THREAD_ID`. The SQL*Plus command:

```
SHOW PARAMETER INSTANCE_NUMBER
```

Shows the current number for each instance.

Note: All instance numbers must be unique.

If you start an instance merely to perform administrative operations with Real Application Clusters disabled, you can omit the `INSTANCE_NUMBER` parameter from the parameter file. An instance starting with Real Application Clusters disabled can also specify a thread other than 1 to use the online redo log files associated with that thread.

See Also:

- *Oracle9i Real Application Clusters Administration* for more information about managing undo space in Real Application Clusters
- *Oracle9i Real Application Clusters Deployment and Performance* for information about allocating free space for inserts and updates
- *The Oracle9i Database Administrator's Guide* for more information on starting Oracle databases

Administering Storage Components in Real Application Clusters

This chapter describes how to administer storage components in **Real Application Clusters**. It includes the following topics:

- [File Management Issues in Real Application Clusters](#)
- [Automatic Undo Management In Real Application Clusters](#)
- [Associating Instances, Users, and Resources with Free List Groups](#)
- [SQL Options for Managing Free Space](#)
- [Preallocating Extents to Free List Groups](#)

See Also: *Oracle9i Real Application Clusters Deployment and Performance* for information about optimizing the use of storage components to improve performance

File Management Issues in Real Application Clusters

This section discusses the following file management issues that are specific to Real Application Clusters:

- [Adding Data Files](#)
- [Automatic Undo Management In Real Application Clusters](#)

Adding Data Files

If you add **data file** while your Real Application Clusters database is running, Oracle automatically controls resource assignments to the new files, just as it controls resource assignments for existing files.

Automatic Undo Management In Real Application Clusters

This section explains how to use **automatic undo management** and manual rollback segment undo in Real Application Clusters. Oracle offers undo space management features in Real Application Clusters that augment the undo space management features available in single-instance environments.

You can use either automatic undo management or rollback segment undo to manage undo space. However, Oracle Corporation strongly recommends that you use the more transparent automatic undo management method. The undo space management topics in this section are:

- [Using Automatic Undo Management](#)
- [Overriding Automatic Undo Management by Using Rollback Segment Undo](#)

Note: You cannot simultaneously use automatic undo management and manual rollback segment undo mode. In other words, all instances within Real Application Cluster environments must run in the same undo mode.

Using Automatic Undo Management

To use automatic undo management, set the following parameters:

- Set the global parameter `UNDO_MANAGEMENT` to `auto` in your **server parameter file**. If you use client-side parameter files, the setting for `UNDO_MANAGEMENT` must be identical in all the files.

- Set the `UNDO_TABLESPACE` parameter to assign undo tablespaces to instances.

You must have already created any undo tablespaces you use with the `UNDO_TABLESPACE` parameters. Otherwise, the `STARTUP` command fails. If you do not set the `UNDO_TABLESPACE` parameter, each **instance** uses the first available undo tablespace. If undo tablespaces are not available, the instances use the `SYSTEM` rollback segment. Therefore, Oracle Corporation recommends that you assign an `UNDO TABLESPACE` to a specific instance to control their use.

Warning: Using the `SYSTEM` rollback segment is not recommended. When you use the `SYSTEM` rollback segment for undo, Oracle writes a message to the **alert files** to warn that your database is running without undo tablespaces.

When you are using automatic undo management, Oracle ignores settings for the `TRANSACTIONS` parameter. This is because Oracle dynamically allocates transaction objects from the System Global Area (SGA) for automatic undo management.

Switching Undo Tablespaces

You can dynamically re-direct undo tablespace use by executing the `ALTER SYSTEM SET UNDO_TABLESPACE` statement. For example, assume you have instances `db1` and `db2` accessing undo tablespaces `undotbs01` and `undotbs02` respectively. If you have an idle undo tablespace, for example, `undotbs03`, you can execute the following statement from either instance to re-direct undo processing to `undotbs03`:

```
ALTER SYSTEM SET UNDO_TABLESPACE = undotbs3;
```

Note: Each instance can only use one undo tablespace at a time. In other words, instances cannot share undo tablespaces.

User transactions proceed normally while Oracle executes this operation. In some circumstances, an instance can temporarily access two undo tablespaces at the same time. This only happens while transition processing occurs during the tablespace switching operation.

This process also does not wait for all user transactions to commit. Instead, it places the previous undo tablespace in a *pending-offline* state if there are active transactions

in that tablespace. This means that the pending offline tablespace may be unavailable for other instances until all transactions against that tablespace are committed.

In the last code example, the previously-used undo tablespaces, `undotbs01` or `undotbs02` remain owned by the instance until the instance's last active transaction has committed.

Each undo tablespace can only be used by one instance for undo at any one time. However, all instances can read undo blocks for consistent read purposes at any time. Also, any instance is allowed to update any undo tablespace during transaction recovery, as long as that undo tablespace is not currently used by another instance for undo generation or transaction recovery.

See Also: Refer to the *Oracle9i Database Administrator's Guide* for more information about the `ALTER SYSTEM SET UNDO_TABLESPACE` statement

System Rollback Segment

When you use automatic undo management, the only external rollback segment Oracle uses is the `SYSTEM` rollback segment. There is only one `SYSTEM` rollback segment for each database. The `SYSTEM` rollback segment resides in the `SYSTEM` tablespace, and Oracle automatically creates it at `CREATE DATABASE` time.

In Real Application Cluster environments, all instances use the same `SYSTEM` rollback segment. Under normal circumstances, the `SYSTEM` rollback segment is only used for performing system transactions, such as the creation of transaction tables. You normally do not have to perform any operations to manage the `SYSTEM` rollback segment.

See Also: The *Oracle9i Database Administrator's Guide* for information about the remaining administrative operations you can perform on undo tablespaces, such as setting the undo retention period and dropping undo tablespaces

Overriding Automatic Undo Management by Using Rollback Segment Undo

Although not recommended, you can override the default automatic undo management by setting the `UNDO_MANAGEMENT` parameter to *manual*. In addition, follow the recommendations in the rest of this section. If you do not specify the `UNDO_MANAGEMENT` parameter, Oracle starts the instance in automatic undo management mode.

Manually Creating Rollback Segments

Real Application Clusters databases need at least as many **rollback segments** as the maximum number of concurrent instances plus one; the extra one is for the `SYSTEM` rollback segment. An instance cannot start up shared without exclusive access to at least one rollback segment, whether it is public or private.

You can create new rollback segments in any tablespace except for the temporary tablespace. To reduce **contention** between rollback data and table data, partition your rollback segments in a separate tablespace. This also facilitates taking tablespaces offline because a tablespace cannot be taken offline if it contains active rollback segments.

In general, make all rollback segment **extent** the same size by specifying identical values for the storage parameters `INITIAL` and `NEXT`. To ensure you have correctly created the rollback segments, examine the data dictionary view `DBA_ROLLBACK_SEGS`. This view shows each rollback segment's name, segment ID number, and owner (`PUBLIC` or other).

See Also: *Oracle9i Database Administrator's Guide* for information about contention for a rollback segment and for information on the performance implications of adding rollback segments

Public and Private Rollback Segments

Public and private rollback segments do not have performance differences. However, private rollback segments provide more control over the matching of instances with rollback segments. This enables you to locate the rollback segments for different instances on different disks to improve performance. Therefore, use private rollback segments to reduce disk contention in high-performance systems.

Public rollback segments form a pool of rollback segments that can be acquired by any instance needing an additional rollback segment. Using public rollback segments can be disadvantageous, however, when instances are shut down and started up at the same time. For example, instance X shuts down and releases public rollback segments. Instance Y starts up and acquires the released rollback segments. Finally, instance X starts up and cannot acquire its original rollback segments. Instances are forced to acquire public rollback segments at startup if you do not properly set the `TRANSACTIONS` and `TRANSACTIONS_PER_ROLLBACK_SEGMENTS` parameters.

You can use public rollback segments to improve space utilization. If you create only one large public rollback segment for long-running transactions that run on different instances each month, you can take the rollback segment offline and bring

it back online or *move* it from one instance to another to better serve instances with the heavier workloads.

By default, a rollback segment is private and is used by the instance specifying it in the parameter file. Specify private rollback segments using the parameter `ROLLBACK_SEGMENTS`.

In addition, the following rules also apply:

- Once a public rollback segment is acquired by an instance, it is then used exclusively by that instance.
- Once created, both public and private rollback segments can be brought online using the `ALTER ROLLBACK SEGMENT` command.
- If an instance has more than the system rollback segment online, then you need more than one rollback segment. If an instance has only the system rollback segment online, you only need one more rollback segment to start the instance.

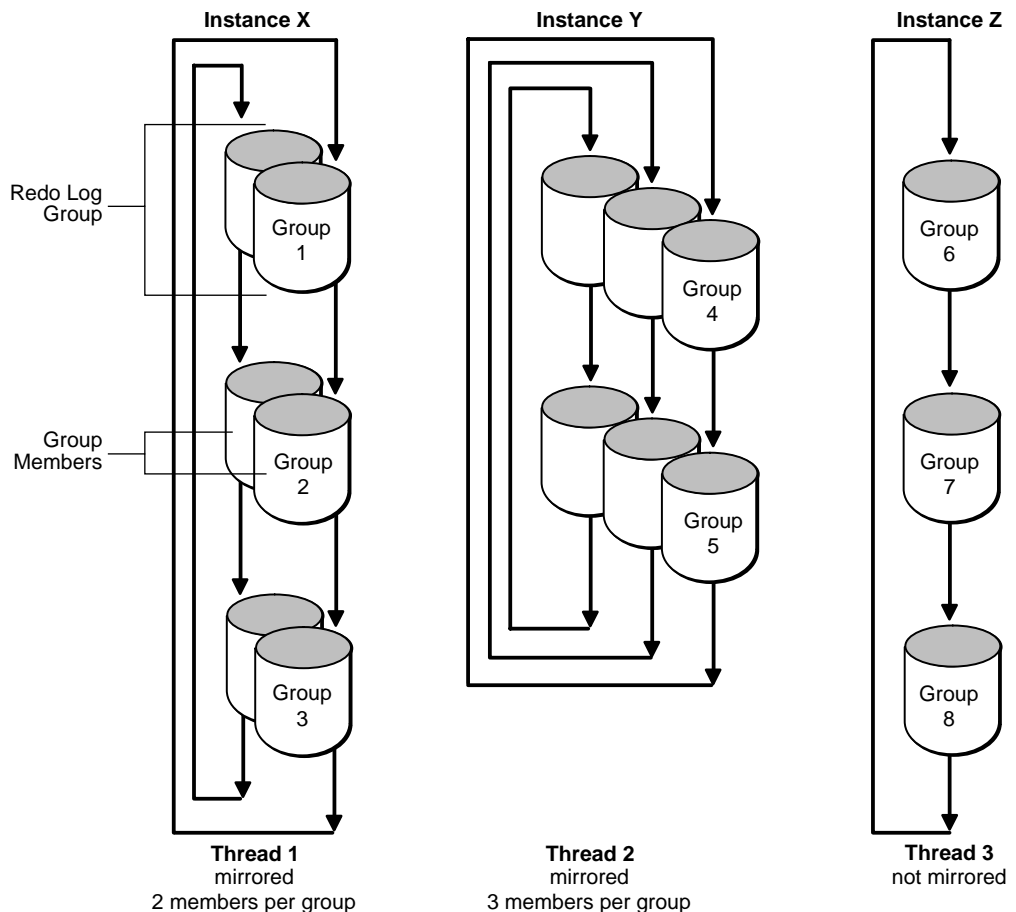
See Also:

- *Oracle9i Database Administrator's Guide* for more information about how instances acquire rollback segments and how to monitor them
- *Oracle9i Database Reference* for a description of `DBA_ROLLBACK_SEGS` and `DBA_SEGMENTS` and for information about other dynamic performance views

Using Redo Log Files

Each instance has its own online redo log groups which are called an instance's *thread* of online redo. Create these online redo log groups and establish group members as described in the *Oracle9i Database Administrator's Guide*. [Figure 3-1](#) shows the threads of redo for three Real Application Clusters instances.

Figure 3-1 Threads of Redo



- Instance X uses thread 1 that contains three *groups* of online **redo log files**. Thread 1 is multiplexed, that is, each group has two copies, or *members*, of the redo log file.

- Instance Y uses thread 2 that contains two groups of online redo log files. Thread 2 is multiplexed and each group of redo logs has three members.
- Instance Z uses thread 3 that contains three groups of online redo log files, but these redo log files are not multiplexed; there is only one member for each group.

Note: The **Oracle Database Configuration Assistant (DBCA)** creates two log files, one member for each group and one thread for each instance.

Group numbers must be unique within the database. However, the order of assigning groups to threads, and threads to instances, is arbitrary.

For example, although in [Figure 3–1](#) thread 1 contains groups 1, 2, and 3 while thread 2 contains groups 4 and 5, you could instead assign groups 2, 4, and 5 to thread 1 while assigning groups 1 and 3 to thread 2. The `V$LOGFILE` view displays the group number associated with each redo log file.

Although it is possible to have different numbers of groups and members for each thread, Oracle Corporation recommends that you configure all threads using a standard that facilitates administration. That is, if possible configure all of your threads like those shown for thread X or Y. Oracle Corporation recommends against using non-mirrored redo log groups as shown for thread Z. Non-standard redo log configurations can be useful, however, for performance reasons.

Different degrees of mirroring may be required for some instances that perform better with less mirroring overhead. For example, one instance could have three groups with two members for each group, a second instance could have four non-multiplexed log files, and a third instance could have two groups with four members for each group.

In Real Application Clusters, each instance must have at least two groups of online redo log files. When the current group fills, an instance begins writing to the next log file group. At a **redo log switch**, Oracle writes information to the **control file** that identifies the filled group and its thread number after it has been archived.

Note: `MAXLOGHISTORY` is useful for sites with very demanding availability requirements. This option can help you administer recovery, especially when there are many instances and many log files.

See Also: *Oracle9i Database Administrator's Guide* for a full description of multiplexed redo log files

Managing Trace Files and the Alert Files

Oracle records information about important events that occur in your Real Application Clusters environment in **trace files** and **alert files**. The trace files and alert files for Real Application Clusters are the same as those in single instance Oracle.

Monitor these files frequently and regularly make copies of them for all instances of your Real Application Clusters environment. This preserves their content and avoids accidentally overwriting the files.

See Also: "[Using Trace Files](#)" on page A-1 for more information about trace files and alert logs

Associating Instances, Users, and Resources with Free List Groups

This section describes:

- [Associating Instances with Free Lists](#)
- [Associating User Processes with Free Lists](#)

In general, all tables should have the same number of **free list groups**, but the number of free lists within a group may vary, depending on the type and amount of activity of each table.

Partitioning free space can particularly improve the performance of applications that have a high volume of concurrent inserts, or updates requiring new space, from multiple instances. Performance improvements also depend, of course, on your operating system, hardware, data block size, and so on.

Note: In multi-instance environments, Oracle does not preserve information about multiple free lists and free list groups upon import. If you use Export and Import to back up and restore your data, the imported data is not partitioned. Oracle recommends that you maintain scripts to pre-create all objects before import to preserve your free list configuration and then import with `IGNORE = Y`.

Associating Instances with Free Lists

Although it is not required for Real Application Clusters, sometimes data partitioning can reduce contention for data blocks. The **Cache Fusion** resources that often cover blocks in one free list group tend to be held primarily by the instance using that free list group. This is because an instance that modifies data is usually more likely to reuse that data than other instances. However, if multiple instances take free space from the same extent, they are more likely to contend for blocks in that extent if they subsequently modify the data that they inserted.

Assignment of New Instances to Existing Free List Groups

If `MAXINSTANCES` is greater than the number of free list groups in the table or cluster, then an **instance number** maps to the free list group associated with:

$$\text{instance_number modulo number_of_free_list_groups}$$

Modulo (or *rem* for *remainder*) is a formula for determining which free list group should be used by calculating a remainder value. In the following example there are 2 free list groups and 10 instances. To determine which free list group instance 6 will use, the formula would read `6 modulo 2 = 0`. Six divided by 2 is 3 with zero remainder, so instance 6 will use free list group 0. Similarly, instance 5 would use free list group 1 because `5 modulo 2 = 1`. Five is divisible by 2 with a remainder of 1.

If there are more free list groups than `MAXINSTANCES`, then a different hashing mechanism is used. If multiple instances share one free list group, they share access to every extent specifically allocated to any instance sharing that free list group.

FREELIST GROUPS and MAXINSTANCES

In a system with relatively few **nodes**, the `FREELIST_GROUPS` option for a table should generally have the same value as the `MAXINSTANCES` option of `CREATE DATABASE`, which limits the number of instances that can access a database concurrently. In a Massively Parallel Processing system, however, `MAXINSTANCES` could be many times larger than `FREELIST_GROUPS` so that many instances share one group of free lists.

See Also: *Oracle9i Real Application Clusters Deployment and Performance* for more information on associating instances and users with free list groups

Associating User Processes with Free Lists

User processes associate with free lists based on their Oracle process IDs. Each user process has access to only one free list in the free list group for the instance on which it is running. Every user process also has access to the **master free list** of free blocks. If a table has multiple free lists but does not have multiple free list groups, or has fewer free list groups than the number of instances, then free lists and free list groups would be shared among user processes from different instances.

Using the statement `ALTER SESSION INSTANCE_NUMBER`, you can increase the instance number value beyond the value of `MAXINSTANCES`. You can also dynamically alter a table's free list assignment with the `ALTER TABLE` statement. However, you cannot use this statement for free list groups.

SQL Options for Managing Free Space

Several SQL options enable you to create free lists and free list groups for tables, clusters, and indexes. You can explicitly specify that new space for an object be taken from a specific data file. You can also associate free space with particular free list groups that you can then associate with particular instances.

The SQL statements include:

```
CREATE [TABLE | CLUSTER | INDEX]
    STORAGE
    FREELISTS
    FREELIST GROUPS
ALTER [TABLE | CLUSTER | INDEX]
    ALLOCATE EXTENT
    SIZE
    DATAFILE
    INSTANCE
```

You can use these SQL options with the `INSTANCE_NUMBER` parameter to associate data blocks with instances.

See Also: *Oracle9i SQL Reference* for complete syntax of these statements

Preallocating Extents to Free List Groups

Preallocating extents is a static approach to the problem of preventing automatic allocation of extents by Oracle. You can preallocate extents to tables that have free list groups. This means that all free blocks are formatted into free lists that reside in

the free list group of the instance to which you are preallocating the extent. This approach is useful if you need to accommodate objects that you expect will grow in size.

The Sequence Number Generator

Real Application Clusters allows users on multiple instances to generate unique sequence numbers with minimal synchronization. The sequence number generator allows multiple instances to access and increment a sequence without contention among instances for sequence numbers and without waiting for transactions to commit.

Each instance can have its own sequence cache for faster access to sequence numbers. Oracle uses **Global Cache Service (GCS) resources** to coordinate sequences across instances in Real Application Clusters.

This section describes the `CREATE SEQUENCE` statement and its options.

- [The CREATE SEQUENCE Statement](#)
- [The CACHE Option](#)
- [The ORDER Option](#)

The CREATE SEQUENCE Statement

The SQL statement `CREATE SEQUENCE` establishes a database object from which multiple users can generate unique integers without waiting for other users to commit transactions to access the same sequence number generator.

Real Application Clusters allows users on multiple instances to generate unique sequence numbers with minimal cooperation or contention among instances.

Sequence numbers are always unique, unless you use the `CYCLE` option. However, you can assign sequence numbers out of order if you use the `CACHE` option without the `ORDER` option, as described in the following section.

See Also: *Oracle9i SQL Reference* for more information about the `CREATE SEQUENCE` and `CYCLE` options

The CACHE Option

The `CACHE` option of `CREATE SEQUENCE` preallocates sequence numbers and retains them in an instance's SGA for faster access. You can specify the number of sequence numbers cached as an argument to the `CACHE` option. The default value is 20.

Caching sequence numbers significantly improves performance but can cause the loss of some numbers in the sequence. In other words, the sequence numbers will not be in chronological order. Losing sequence numbers is unimportant in some applications, such as when sequences are used to generate unique numbers for primary keys.

A cache for a given sequence is populated at the first request for a number from that sequence. After the last number in that cached set of numbers is assigned, the cache is repopulated with another set of numbers.

Each instance keeps its own cache of sequence numbers in memory. When an instance shuts down, cached sequence values that have not been used in committed DML statements can be lost. The potential number of lost values can be as great as the value of the `CACHE` option multiplied by the number of instances shutting down. Cached sequence numbers can be lost even when an instance shuts down normally.

The `ORDER` Option

The `ORDER` option of `CREATE SEQUENCE` guarantees that sequence numbers are generated in the order of the requests. You can use the `ORDER` option for time-stamp numbers and other sequences that must indicate the request order across multiple processes and instances.

If you do not need Oracle to issue sequence numbers in order, the `NOORDER` option of `CREATE SEQUENCE` can significantly reduce overhead in a Real Application Clusters environment.

Note: Real Application Clusters databases do not support the `CACHE` option with the `ORDER` option of `CREATE SEQUENCE` when the database is mounted in cluster mode. Oracle cannot guarantee an order if each instance has some sequence values cached. Therefore, if you should create sequences with both the `CACHE` and `ORDER` options, they will be ordered but not cached.

Administering Real Application Clusters Databases with Server Control, SQL, and SQL*Plus

This chapter explains how to administer instances and databases in **Real Application Clusters** environments using **Server Control (SRVCTL)**, SQL, and SQL*Plus. The topics in this chapter include:

- [Using SRVCTL to Administer Real Application Clusters Instances](#)
- [Global Services Daemon \(GSD\)](#)
- [Administering Real Application Clusters Databases Using SQL and SQL*Plus](#)
- [Shutting Down Real Application Clusters Instances](#)
- [Quiescing A Real Application Clusters Database](#)
- [How SQL and SQL*Plus Commands Affect Instances](#)

See Also: [Chapter 5, "Administering Real Application Clusters Databases with Oracle Enterprise Manager"](#) for information on using **Oracle Enterprise Manager** to administer Real Application Clusters

Using SRVCTL to Administer Real Application Clusters Instances

Oracle Corporation recommends that you use SRVCTL to administer your Real Application Clusters database environment. SRVCTL manages configuration information that is used by several Oracle tools. For example, Oracle Enterprise Manager and the **Intelligent Agent** use the configuration information that SRVCTL generates to discover and monitor **nodes** in your **cluster**.

Before using SRVCTL, ensure that your **Global Services Daemon (GSD)** is running after you configure your database as described in *Oracle9i Real Application Clusters Installation and Configuration*. Do this by executing the appropriate command to run the GSD based on your platform. Oracle should respond with a message stating that the GSD is already running.

For more information on using the Global Services Daemon, refer to "**Global Services Daemon (GSD)**" on page 4-9.

Note: To use SRVCTL, you must have already created the configuration information for the database that you want to administer. You must have done this either by using the **Oracle Database Configuration Assistant (DBCA)**, or by using the `srvctl add` command as described in the chapter on manually configuring your database in *Oracle9i Real Application Clusters Installation and Configuration*.

Administrative Tasks That You Can Accomplish with SRVCTL

This section describes how to use the SRVCTL utility to administer instances in Real Application Clusters. You can use SRVCTL to perform two types of administrative tasks as listed under the following sub-headings:

- [SRVCTL Cluster Database Tasks](#)
- [SRVCTL Cluster Database Configuration Tasks](#)

SRVCTL Cluster Database Tasks

- Start and stop cluster databases
- Start and stop cluster database instances
- Start and stop listeners associated with a cluster database
- Start and stop listeners associated with a cluster database instance
- Obtain the status of a cluster database instance

- Obtain the statuses of listeners associated with a cluster database

SRVCTL Cluster Database Configuration Tasks

- Add and delete cluster database configuration information
- Add an **instance** to and delete an instance from a cluster database configuration
- Rename **instance name** within a cluster database configuration
- Move instances in a cluster database configuration
- Set and unset the environment for an instance in a cluster database configuration
- Set and unset the environment for an entire cluster database in a cluster database configuration

See Also: For procedures on adding nodes, refer to [Chapter 9, "Adding Nodes and Instances and Deleting Instances in Real Application Clusters"](#)

SRVCTL Command Syntax

To see the online command syntax and options for each SRVCTL command, enter:

```
srvctl command option -h
```

Where *command option* is one of the valid options such as *start*, *stop*, or *status*. To see a list of available command options, enter:

```
srvctl
```

The remainder of this section describes the SRVCTL commands.

srvctl start

Use the `srvctl start` command to start all instances or a subset of instances in your Real Application Clusters database. For example, to start all the instances use the syntax:

```
srvctl start -p db_name
```

Or you can start specific instances using the syntax:

```
srvctl start -p db_name -i instance_name
```

This syntax starts the specific instance that you name. Using `srvctl start` also starts all listeners associated with an instance.

Use the `-s` option (lower-case *s*) as in the following syntax where *stage* is either *inst* or *lsnr*:

```
srvctl start -p db_name -s stage
```

Use this syntax to separately start instances and listeners.

srvctl stop

Use the `srvctl stop` command to stop all instances or a subset of instances in your Real Application Clusters database. For example, to stop all instances use the syntax:

```
srvctl stop -p db_name
```

Or you can stop specific instances using:

```
srvctl stop -p db_name -i instance_name
```

Using `srvctl stop` also stops all listeners associated with an instance.

Use the `-s` option (lower-case *s*) as in the following syntax where *stage* is either *inst* or *lsnr*:

```
srvctl stop -p db_name -s stage
```

Use this syntax to separately stop instances and listeners.

srvctl status

Use the `srvctl status` command to determine what instances are running. For example, use the output from the following syntax to identify which instances are running:

```
srvctl status -p db_name
```

Use the `-s` option (lower-case *s*) as in the following syntax where *stage* is either *inst* or *lsnr*:

```
srvctl status -p db_name -s stage
```

Use this syntax to obtain the status of an instance or a listener.

srvctl config

Use the `srvctl config` command to identify the existing Real Application Clusters databases. You can use two syntaxes for `srvctl config`. For example, the following syntax lists all the Real Application Clusters databases in your environment:

```
srvctl config
```

The following syntax lists the instances for the Real Application Clusters database name that you provide:

```
srvctl config -p db_name
```

The Oracle Enterprise Manager **auto-discovery** process also uses output from this command to discover the configurations for databases in your Real Application Clusters.

srvctl get env

Use the `srvctl get env` command to obtain environment information for either a specific instance or for an entire Real Application Clusters database. For example, the output from the following syntax displays environment information for the entire Real Application Clusters database identified by the name you provide:

```
srvctl get env -p db_name
```

The following syntax displays environment information for a specific instance:

```
srvctl get env -p db_name -i instance_name
```

Updating Your Configuration Information with SRVCTL

The remaining SRVCTL commands update your configuration information. If you use these commands, you must also use SQL to add or remove database objects as needed so that your instance or database can operate. For example, you must add redo log groups and **thread IDs** for instances if you use the `srvctl add instance` command.

Note: Oracle Corporation recommends that you use the Instance Management feature of the Oracle Database Configuration Assistant to add and delete instances.

srvctl add db

The `srvctl add db` command creates the configuration for the Real Application Clusters database. For example, the following syntax adds the configuration information for a Real Application Clusters database to the configuration repository.

```
srvctl add db -p db_name -o oracle_home
```

This database is identified by the name that you provide in the command syntax. You must then execute the `srvctl add instance` command to add instance configurations to the database.

srvctl add instance

Use the `srvctl add instance` command to add static configuration information for an instance. For example, the following syntax adds an instance using the instance name that you provide:

```
srvctl add instance -p db_name -i instance_name -n node_name
```

As mentioned, this command only updates the configuration; it does not create the database. Execute the `srvctl add db` command before using the `srvctl add instance` command.

srvctl delete instance

Use the `srvctl delete instance` command to delete static configuration information for a Real Application Clusters instance. For example, the following syntax deletes the configuration for the instance identified by the database name that you provide:

```
srvctl delete instance -p db_name -i instance_name
```

srvctl delete db

Use the `srvctl delete db` command to delete the static configuration for a Real Application Clusters database. For example, the following syntax deletes the Real Application Clusters database identified by the name that you provide:

```
srvctl delete db -p db_name
```

srvctl rename instance

Use the `srvctl rename instance` command to rename a Real Application Clusters database instance. For example, the following syntax renames the Real Application Clusters instance identified by the name that you provide:

```
srvctl rename instance -p db_name -i old_name -e new_name
```

srvctl move instance

Use the `srvctl move instance` command to move a Real Application Clusters instance from one node to another in the static configuration. For example, the following syntax moves the Real Application Clusters instance to the node identified by the node name that you provide:

```
srvctl move instance -p db_name -i instance_name -n new_node
```

srvctl set env

Use the `srvctl set env` command to set the environment that Oracle uses when you start instances in a Real Application Clusters database that you name. For example, the following syntax sets the environment for an entire Real Application Clusters database:

```
srvctl set env -p db_name -t name=value
```

You can also set the environment for specific instances. For example, the following syntax sets the environment variable for the Real Application Clusters database instance that you name:

```
srvctl set env -p db_name -t name=value -i instance_name
```

An example of this command is:

```
srvctl set env -p proddb -t LANG=en
```

srvctl unset env

Use the `srvctl unset env` command to unset the environment that Oracle uses when you start instances in the Real Application Clusters database that you name. For example, the following syntax unsets the environment variable for an entire Real Application Clusters database:

```
srvctl unset env -p db_name -t environment_variable
```

An example of this command is:

```
srvctl unset env -p proddb -t LANG
```

You can also unset the environment for specific instances. For example, the following syntax unsets the environment variable for the Real Application Clusters database instance that you name:

```
srvctl unset env -p db_name -t environment_variable -i instance_name
```

An example of this command is:

```
srvctl unset env -p proddb -t LANG -i proddb1
```

Importing and Exporting Raw Device Configuration with SRVCONFIG

Use SRVCONFIG to import and export raw device configuration information. For example, the following syntax exports the contents of the configuration information to the text file that you name:

```
srvconfig -exp file name
```

While the following example imports configuration information from the text file you name to the configuration repository for the Real Application Clusters environment in which you execute the command:

```
srvconfig -imp file name
```

Migrating Oracle8i Configurations to Oracle9i

If you are upgrading from Oracle8i to Oracle9i, migrate your configuration information using the following post-installation procedure. Do this for each Real Application Clusters database:

1. Stop all Global Services Daemons (GSD).
2. Execute the following command from the node in a UNIX cluster where the *db_name.conf* file is located:

```
srvconfig -conv $Oracle_Home/ops/db_name.conf
```

Global Services Daemon (GSD)

Clients of the Global Services Daemon (GSD), such as SRVCTL, the DBCA, and Oracle Enterprise Manager, interact with the GSD to perform various manageability operations on the nodes in your cluster. You must start the GSD on all the nodes in your Real Applications Clusters database so that the manageability features and tools operate properly.

For example, if you start an instance using Oracle Enterprise Manager, the Intelligent Agent launches a script that contains SRVCTL commands. The GSD executes these commands which correspond to the requested operation.

Note: Only run one GSD service on each node regardless of how many Real Application Clusters database installations you have.

UNIX GSD Implementations

The name of the Global Services Daemon on UNIX platforms is *gsd*. The GSD is located in the `$ORACLE_HOME/bin` directory. The GSD records information such as connection requests from SRVCTL and stores these records in the `gsdaemon.log` file in the `$ORACLE_HOME/srvm/log` directory.

Windows GSD Implementations

The name of the GSD service on Windows NT and Windows 2000 platforms is *OracleGSD*. The GSD service is located in the `%ORACLE_HOME%\bin` directory. The GSD service records information such as connection requests from SRVCTL and stores these records in the `gsdservice.log` file in the `%ORACLE_HOME%\srvm\log` directory.

Administering the GSD on Windows Platforms

You can use *gsdservice* commands to install the GSD service in the Windows Services panel. You can also use this command to start and stop the GSD service. The options for *gsdservice* are:

- `gsdservice -install` — To install the GSD service in the Services panel
- `gsdservice -start` — To start the GSD service
- `gsdservice -remove` — To stop and delete the GSD service

Administering Real Application Clusters Databases Using SQL and SQL*Plus

Although Oracle Corporation recommends that you use SRVCTL to administer your Real Application Clusters database environment, you can also use SQL and SQL*Plus. Prior to performing the tasks with SQL and SQL*Plus as described in this section, ensure your **Cluster Manager (CM)** component is started on each node.

The SQL and SQL*Plus procedures you use to start your cluster database depends on your platform as explained under the following sub-headings:

- [Starting Databases in Cluster Mode on UNIX](#)
- [Starting Databases in Cluster Mode on Windows NT and Windows 2000](#)

Starting Databases in Cluster Mode on UNIX

To start a Real Application Clusters database in cluster mode:

1. Ensure your Cluster Manager software is running. Instructions on Cluster Manager software administration appear in your operating-system specific documentation. If the Cluster Manager is not available or if Oracle cannot communicate with this component, Oracle displays the error ORA-29701: "Unable to connect to Cluster Manager".
2. Start any required operating system specific processes. For more information about these processes, see your operating system-specific documentation.
3. If the **listener** is not started, start it on each of the nodes. Enter:

```
LSNRCTL
LSNRCTL> start [listener_name]
```

Where *listener_name* is the name of the listener defined in the **listener.ora** file. It is not necessary to identify the listener if you are using the default listener named LISTENER.

LSNRCTL displays a status message indicating that the listener started successfully. You can check that all expected services for that listener are listed in the services summary in the status message. You can also check the status of the listener with the LSNRCTL STATUS command.

4. Start the database on one of the nodes by starting SQL*Plus. Then enter:

```
CONNECT SYS/password as SYSDBA
STARTUP PFILE=init$ORACLE_sid.ora
```

The first instance to start in cluster mode determines the values of any global parameters for the other instances. When another instance attempts to start in cluster mode, the Real Application Clusters database compares the values of any global parameters in its parameter file with those already in use and issues messages if any values are incompatible. An instance cannot mount the database unless it has the correct values for its global parameters.

5. On the remaining nodes, start the database:

```
CONNECT SYS/password as SYSDBA
STARTUP PFILE=$ORACLE_sid.ora;
```

Starting Databases in Cluster Mode on Windows NT and Windows 2000

To start the Real Application Clusters database in cluster mode on Windows platforms:

1. Start OracleServicesid instance on *each* node.
 - From the MS-DOS command line enter:


```
C:\> net start OracleServicesid
```
 - From the Control Panel's Services window, select OracleServicesid, then click Start.
2. If the listener is not started, start it on each of the nodes. Enter:

```
LSNRCTL
LSNRCTL> start [listener_name]
```

Where *listener_name* is the name of the listener defined in the `listener.ora` file. You do not have to identify the listener if you are using the default listener named `LISTENER`.

`LSNRCTL` displays a status message indicating that the listener started successfully. You can check that all expected services for that listener appear in the services summary in the status message. You can also check the status of the listener with the `LSNRCTL STATUS` command.

3. Start the database on one of the nodes by starting SQL*Plus. Then enter:

```
CONNECT SYS/password
STARTUP PFILE=%ORACLE_HOME%\database\initsid.ora;
```

The first instance to start in cluster mode determines the values of any global parameters for the other instances. When another instance attempts to start in

cluster mode, the Real Application Clusters database compares the values of any global parameters in its parameter file with those already in use and issues messages if any values are incompatible. The instance cannot mount the database unless it has the correct values for its global parameters.

4. On the remaining nodes, start the database:

```
CONNECT SYS\password
STARTUP PFILE=%ORACLE_HOME%\database\initsid.ora;
```

Using RETRY to Mount a Database in Cluster Mode

If you attempt to start an instance and mount a database in cluster mode while another instance is recovering the same database, your current instance cannot mount the database until the recovery is complete. Rather than repeatedly attempting to start the instance, use the `STARTUP RETRY` statement. This causes the new instance to retry mounting the database every five seconds until it succeeds or has reached the retry limit. Use the syntax:

```
STARTUP OPEN database_name RETRY
```

To set the maximum number of times the instance attempts to mount the database, use the SQL*Plus `SET` command with the `RETRY` option. You can specify either an integer such as 10, or the keyword `INFINITE`.

If the database can only be opened by being recovered by another instance, then using the `RETRY` does not repeat connection attempts. For example, if the database was mounted in exclusive mode by one instance, then trying the `STARTUP RETRY` command in cluster mode does not work for another instance.

Note: Because an instance startup does not affect the **data files**, you can start an instance without mounting the data files.

Setting and Connecting to Instances

Before setting instances and connecting to them, you must install and configure Oracle Net for the Real Application Clusters nodes and any clients that access these nodes. This establishes remote connections from the clients to the nodes.

See Also:

- *Oracle9i Real Application Clusters Installation and Configuration*
- *Oracle Net Services Administrator's Guide*

SQL*Plus commands operate on the current instance with some exceptions as noted under the next heading, "[The SET INSTANCE and SHOW INSTANCE Commands](#)" on page 4-14.

The current instance can be either the local default instance on which you initiated your SQL*Plus session, or it can be a remote instance. Because the SQL*Plus prompt does not show which instance is the current instance, be sure you direct your commands to the correct instance.

Starting a SQL*Plus session and connecting to the database without specifying an instance directs all SQL*Plus commands to the local instance. In this case, the default instance is also the current instance.

To switch the current instance from the local instance to a remote instance, do one of the following:

- Re-execute the `CONNECT` command specifying a remote instance **net service name** as in this example:

```
CONNECT SYSTEM/MANAGER@net_service_name
```

- Disconnect from the database and execute a `SET INSTANCE` command as in this example:

```
SET INSTANCE net_service_name
```

Issue another `CONNECT` command with only your user ID and password. Specifying a remote instance with the `CONNECT` command while connected to the database by way of an instance enables you to switch from one instance to another without disconnecting.

See Also:

- *The Oracle Net Services Administrator's Guide* for information on configuring net service names
- Your operating system-specific Oracle documentation for more information about the exact format required for the connect string used in the `SET INSTANCE` and `CONNECT` commands

The `SET INSTANCE` and `SHOW INSTANCE` Commands

When using `SET INSTANCE` to specify an instance on a remote node for the `STARTUP` command, the parameter file for the remote instance must be accessible by the local node.

The `SHOW INSTANCE` command displays the net service name for the current instance. `SHOW INSTANCE` returns the value `local` if you have not used `SET INSTANCE` during the SQL*Plus session.

To reset to the default instance, use `SET INSTANCE` without specifying a net service name or specify `local`. Do not follow the `SET INSTANCE` command with the word `default`; this syntax specifies a connect string for an instance named `default`.

The `CONNECT` Command

Connecting as **`SYSOPER`** or **`SYSDBA`** enables you to perform privileged operations, such as instance startup and shutdown. Multiple SQL*Plus sessions can connect to the same instance at the same time. SQL*Plus automatically disconnects you from the first instance whenever you connect to another one.

See Also:

- *The Oracle Net Services Administrator's Guide* or the proper specification of `net_service_name`
- *Oracle9i Database Administrator's Guide* for information on connecting to the database using `SYSDBA` or `SYSOPER` privileges

Verifying That Instances are Running

To verify that instances are running:

1. On any node, enter:

```
CONNECT SYS/password
SELECT * FROM V$ACTIVE_INSTANCES;
```

Oracle returns output similar to the following:

```
INST_NUMBER INST_NAME
-----
1 db1-sun:db1
2 db2-sun:db2
3 db3-sun:db3
```

Where the output columns from this `SELECT` statement are as described in [Table 4-1](#):

Table 4-1 Descriptions of `V$ACTIVE_INSTANCES` Columns

Column	Description
<code>INSTANCE_NUMBER</code>	Identifies the instance number.
<code>INST_NAME</code>	Identifies the host name and instance name.

Shutting Down Real Application Clusters Instances

Shutting down Real Application Clusters instances is procedurally identical to shutting down instances in single instance environments with these exceptions:

- In Real Application Clusters, shutting down one instance does not interfere with the operation of other running instances.
- To shut down a database mounted in shared mode, shut down every instance in the Real Application Clusters environment.
- When you shut down an instance abnormally, Oracle forces all user processes running in that instance to log off the database. If a user process is currently accessing the database, Oracle terminates that access and displays the message "ORA-1092: Oracle instance terminated. Disconnection forced". If a user process is not currently accessing the database when the instance shuts down, Oracle displays the message "ORA-1012: Not logged on" upon the next call or request made to Oracle.

- After a `NORMAL` or `IMMEDIATE` shutdown, instance recovery is not required. Recovery is required, however, after you issue the `SHUTDOWN ABORT` command or after an instance terminates abnormally. The `SMON` process of an instance that is still running performs instance recovery for the instance that shut down. If no other instances are running, the next instance to open the database performs instance recovery for any instances needing it.
- The `SHUTDOWN TRANSACTIONAL` command with the `LOCAL` option is useful to shutdown an instance after all active transactions on the instance have either committed or rolled back. This is in addition to what this command does for `SHUTDOWN IMMEDIATE`. Transactions on other instances do not block this operation. If you omit the `LOCAL` option, then this operation waits until transactions on all other instances that started before the shutdown was issued either commit or rollback.
- If multiple `SQL*Plus` sessions are connected to the same instance simultaneously, all but one must disconnect before the instance can shut down normally. You can use the `IMMEDIATE` or `ABORT` option of the `SHUTDOWN` command to shut down an instance when multiple `SQL*Plus` sessions (or any other sessions) are connected to it.

See Also: *Oracle9i Database Administrator's Guide* for more information on shutting down Oracle databases

Quiescing A Real Application Clusters Database

Quiescing a Real Application Clusters database is procedurally identical to quiescing a single-instance database except as described in this section. For example, you cannot open the database on one instance if the database is *being quiesced*. In other words, if you issued the `ALTER SYSTEM QUIESCE RESTRICTED` statement but Oracle has not finished processing it, you cannot open the database. Nor can you open the database if it is already in a quiesced state. In addition, the `ALTER SYSTEM QUIESCE RESTRICTED` and `ALTER SYSTEM UNQUIESCE` statements affect all instances in a Real Application Clusters environment, not just the instance that issues the command.

See Also: The *Oracle9i Database Administrator's Guide* for details on the quiesce database feature and the *Oracle9i SQL Reference* for more information about the `ALTER SYSTEM QUIESCE RESTRICTED` syntax

How SQL and SQL*Plus Commands Affect Instances

Most SQL statements apply to the current instance. For example, the statement `ALTER SYSTEM SET CHECKPOINT LOCAL` only applies to the instance to which you are currently connected, rather than the default instance or all instances.

`ALTER SYSTEM CHECKPOINT LOCAL` also applies to the current instance. By contrast, `ALTER SYSTEM CHECKPOINT` or `ALTER SYSTEM CHECKPOINT GLOBAL` applies to *all* instances.

`ALTER SYSTEM SWITCH LOGFILE` applies only to the current instance. To force a global log switch, use the `ALTER SYSTEM ARCHIVE LOG CURRENT` statement. The `THREAD` option of `ALTER SYSTEM ARCHIVE LOG` enables you to archive online redo log files for a specific instance.

Table 4–2 describes how common SQL*Plus commands affect instances.

Table 4–2 How SQL*Plus Commands Apply to Instances

SQL*Plus Command	Associated Instance
ARCHIVE LOG	Always applies to the current instance.
CONNECT	Applies the default instance if no instance is specified in the <code>CONNECT</code> command.
HOST	Applies to the node running the SQL*Plus session, regardless of the location of the current and default instances.
RECOVER	Does not apply to any particular instance, but rather to the database.
SHOW INSTANCE	Displays information about the current instance, which can be different from the default local instance if you have redirected your commands to a remote instance.
SHOW PARAMETER and SHOW SGA	Display parameter and SGA information from the current instance.
STARTUP and SHUTDOWN	Always apply to the current instance. These are privileged SQL*Plus commands.

Note: The security mechanism that Oracle uses when you execute privileged SQL*Plus commands depends on your operating system. Most operating systems have a secure authentication mechanism when logging onto the operating system. On these systems, your default operating system privileges usually determine whether you can use `STARTUP` and `SHUTDOWN`. For more information, refer to your operating system-specific documentation.

Part II

Using Oracle Enterprise Manager to Administer Real Application Clusters

Part Two contains information about using the Server Management (SRVM) component of Oracle Enterprise Manager. The chapter in Part Two is:

- [Chapter 5, "Administering Real Application Clusters Databases with Oracle Enterprise Manager"](#)

Administering Real Application Clusters Databases with Oracle Enterprise Manager

This chapter describes how **Oracle Enterprise Manager** manages multiple instances. The **Server Management (SRVM)** component of **Real Application Clusters** can perform a variety of **cluster** database management tasks. Many of these tasks can be initiated from the Oracle Enterprise Manager console.

This section is meant to only describe Oracle Enterprise Manager administration for Real Application Clusters. Use this section as a supplement to information contained in the *Oracle Enterprise Manager Administrator's Guide* and other Oracle Enterprise Manager documentation.

The topics in this chapter are:

- [Overview of Oracle Enterprise Manager Administration](#)
- [Starting the Console](#)
- [Displaying Objects in the Navigator Pane](#)
- [Using the Cluster Database Right-Mouse Menu](#)
- [Starting a Cluster Database](#)
- [Shutting Down a Cluster Database](#)
- [Viewing the Cluster Database Operation Results](#)
- [Viewing Cluster Database Status](#)
- [Creating a Job for a Cluster Database or Instance](#)
- [Specifying Job Details](#)
- [Registering Cluster Database Events](#)

Overview of Oracle Enterprise Manager Administration

The Oracle Enterprise Manager **Console** provides a central point of control for the Oracle environment through an intuitive graphical user interface (GUI). The Oracle Enterprise Manager Console can initiate a variety of cluster database management tasks at the managed cluster **nodes** with the Server Management (SRVM) component.

After nodes have been discovered, you can use the Oracle Enterprise Manager Console to start, stop, and monitor services as well as to schedule jobs or register events. You also perform these tasks simultaneously on multiple nodes. You can also use the Console to manage schemas, security, and the storage features of cluster databases.

See Also: *Oracle9i Real Application Clusters Installation and Configuration* for configuration information. The Oracle Enterprise Manager online Help system may also answer many frequently asked questions.

Starting the Console

To use the Oracle Enterprise Manager Console, start the following components:

1. An **Oracle Intelligent Agent** on each of the nodes
2. **Management Server**
3. Console

See Also: *Oracle9i Real Application Clusters Installation and Configuration* for Real Application Clusters-specific information on installing Oracle Enterprise Manager

Displaying Objects in the Navigator Pane

From the Navigator pane, you can view and manage both single- and multiple-**instance** databases. After nodes are discovered, the Navigator displays cluster databases, their instances, and other related services, such as a **listener**. The information available for cluster databases is essentially the same as for single-instance databases. Just as in single instance databases, cluster databases and all of their related elements can be administered using master/detail views and Navigator menus.

The Navigator displays all the network objects and their relationships to other objects including a direct view of objects such as user-defined groups, nodes, listeners, servers, databases, and database objects.

In the Navigator tree, cluster databases are listed under the Databases folder as siblings to single-instance databases. Just as in single instance databases, each cluster database folder contains the instances and sub-folders for Instance, Schema, Security, and Storage, as shown in [Figure 5-1](#).

Note: If you attempt to expand a cluster database and you have not yet set preferred credentials for the database as described in *Oracle9i Real Application Clusters Installation and Configuration*, the Database Connect Information dialog prompts you to enter database connect information.

By selecting objects within a Cluster Database subfolder, you can access property sheets to inspect and modify properties of these objects. (Just as you can for single-instance databases.) For example, by expanding the Storage folder and then right-clicking Redo Log Groups subfolder, and then choosing Create, you can add a new redo log group. All discovered instances will be displayed, under the Cluster Database Instances folder.

When accessing cluster databases, only the Instance folder has subfolder contents that differ from those seen in single instance databases. Within the Instance folder, the instance database subfolders are split into two functional parts:

- [Database-Specific File Structure](#)
- [Instance-Specific File Structure](#)

Database-Specific File Structure

All of the database-specific functionality is seen directly under the Instance subfolder for the cluster database. The available database-specific functionality includes:

- **In-doubt Transactions.** This provides details on distributed database in-doubt transactions.
- **Resource Consumer Groups.** This lets you view, create, and modify resource consumer groups.

Instance-Specific File Structure

All instance-specific functionality appears beneath the individual instance icons within the Cluster Database Instances subfolder. The instance-specific functionality includes:

Configuration

Here you can view instance states, view and edit initialization parameters, toggle the archive mode, and view performance statistics of active resource plans.

Stored Configurations

Here you can create, edit, and store multiple startup configurations for instances. This eliminates the need to track `initsid.ora` parameter files.

Sessions

Here you can list the status of connected users, view the latest SQL for sessions, and terminate sessions.

Locks

Here you can view list details for currently held User type and System type locks.

Resource Plans

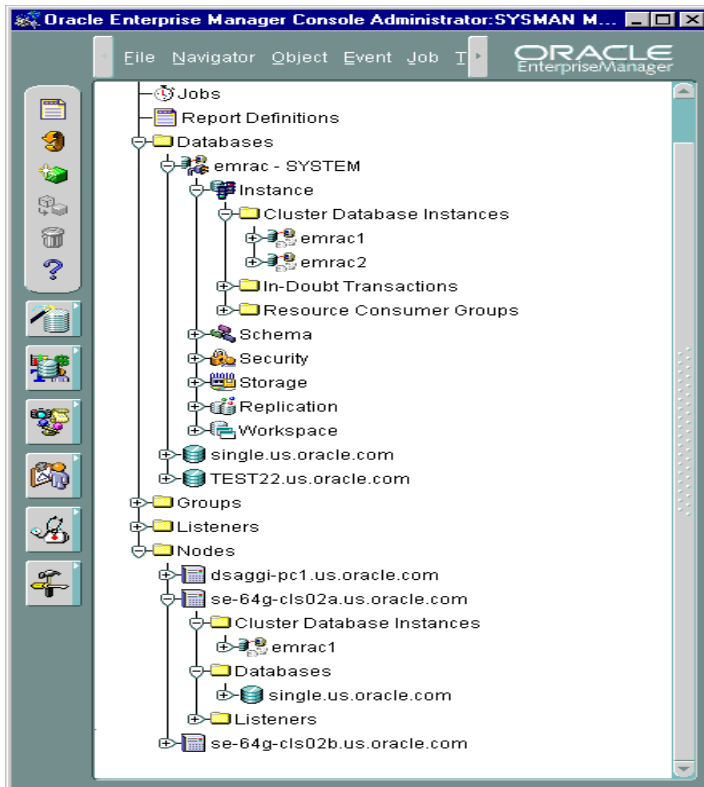
Here you can define and modify the resource plans for the cluster database. You can also activate a specific resource plan for an individual instance.

Resource Plan Schedule

Here you can schedule activation of a resource plan.

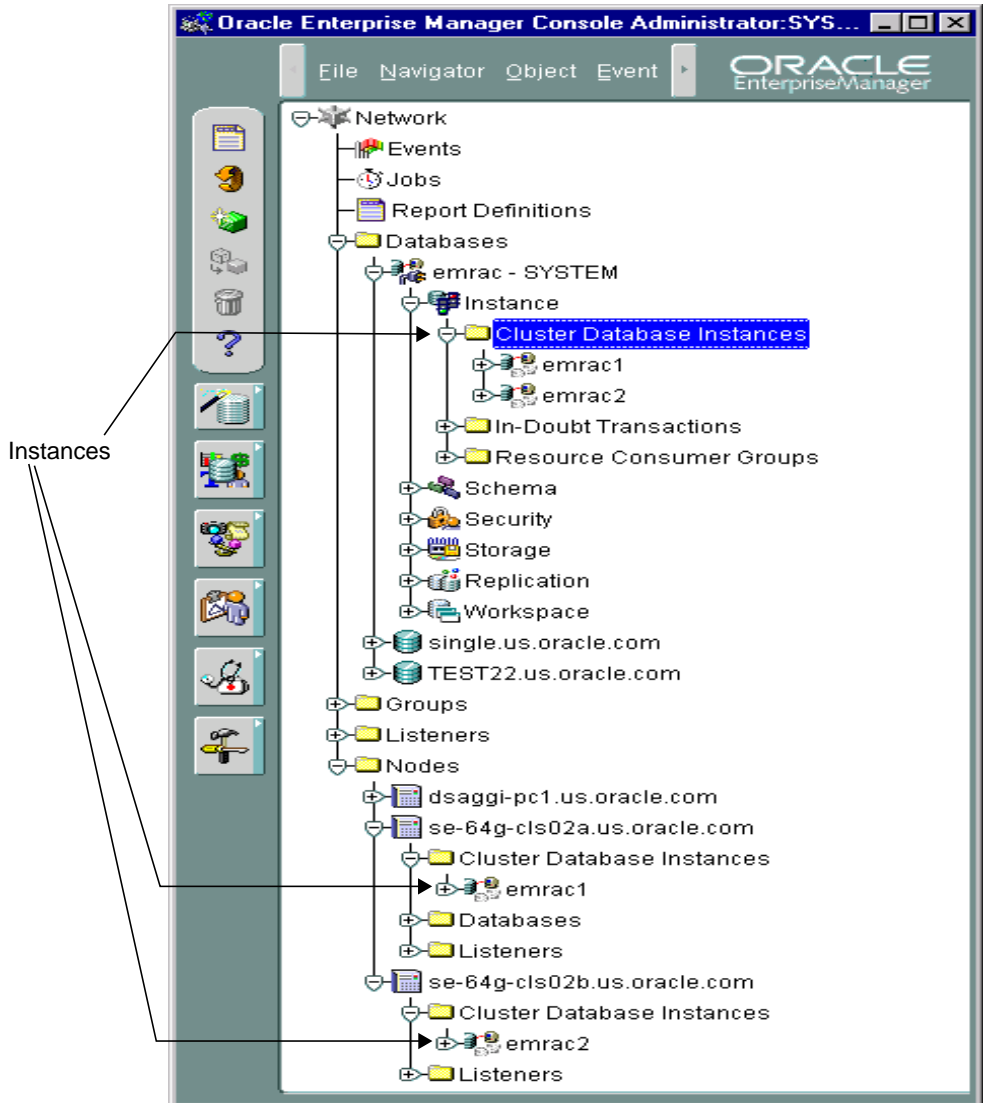
The contents of a typical database folder are shown in [Figure 5-1](#).

Figure 5-1 Database Subfolders



You can expand the Cluster Database Instances folder to display the instances belonging to each discovered database or node, as shown in [Figure 5-2](#).

Figure 5-2 Cluster Database Instances Folder



Note: The instances have the same right-mouse menu as the right-mouse menu that is used in single instance databases.

Using the Cluster Database Right-Mouse Menu

Right-clicking on a cluster database displays the specialized cluster database right-mouse menu as shown in [Figure 5-3](#).

Figure 5-3 *Right Mouse Menu*



The menu contains entries for the specialized functions listed in [Table 5-1](#). (All others are identical to single instance Oracle Enterprise Manager.) When selected, you will see dialogs that differ from single instance databases. Note that in addition to the tools listed in [Table 5-1](#) you will also see the single instance tools, which also work with Real Application Clusters.

Table 5-1 *Cluster Database Right-Mouse Menu Functions Specific to Real Application Clusters*

Option	Description
Startup	Starts the database See Also: "Starting a Cluster Database"
Shutdown	Shuts down the database See Also: "Shutting Down a Cluster Database" on page 5-9

Option	Description
Results	Displays startup and shutdown results See Also: " Viewing the Cluster Database Operation Results " on page 5-11.
View Edit Details	Enables inspection of the state of the cluster database, including which instances are active. See Also: " Viewing Cluster Database Status " on page 5-14
Related Tools	Contains access to other tools which have been enabled for a cluster database.

Starting a Cluster Database

The Console enables you to start an entire cluster database or selected instances within the cluster database.

To start up a cluster database or its instances:

1. In the Navigator pane, expand Databases.
2. Right-click on a **Cluster Database**.
A menu appears with options for the database.
3. Choose **Startup** from the menu.
The Cluster Database Startup dialog box appears.
4. Select a startup type as shown in [Table 5-2](#).

Table 5-2 Cluster Database Startup Types

Option	Description
No Mount	Does not mount the database upon instance startup
Mount	Mounts a database but does not open it
Open	(<i>default</i>) Opens the database
Force the instance(s) to start	Shuts down the currently running Oracle instances with the SHUTDOWN mode, ABORT, before restarting them. If the instances are running and FORCE is not specified, an error results. Note: You should <i>not</i> use the FORCE mode under normal circumstances. Use the FORCE mode only while debugging or under abnormal circumstances.

Option	Description
Restrict access to the database	Makes the started instances accessible only to users with the <code>RESTRICTED SESSION</code> system privilege. Users already connected are not affected.

5. To start up all instances, click **Startup**. To start up only selected instances, follow these steps:
 - a. Select **Instances**.
The Select Instances to Start dialog box appears.
 - b. Select the instances to start up in the Available list, then click **Add**.
 - c. Click **OK** to close the Select Instances to Start dialog box.
 - d. Click **Startup** from the Cluster Database Startup dialog box.

The Cluster Database Startup Results dialog box displays the progress of the startup operation, as described in "[Viewing the Cluster Database Operation Results](#)" on page 5-11.

If the instances were started successfully, the Cluster Database Started message box appears with a successful message.

Click **OK** in the Cluster Database Started message to acknowledge the message, then click **Close** in the Cluster Database Startup Results.

If the startup fails, the Cluster Database Started message box appears with a failure message. Click **View Details** to view more information in the Cluster Database Startup Results dialog box about why the startup failed, then click **Close**.

Shutting Down a Cluster Database

The Console enables you to shut down an entire cluster database or selected instances within a cluster database. Once all instances are shut down, the database is also considered shut down.

Note: A cluster database can occasionally be completely down, but some of its services, such as the database listener, might remain running.

To shut down a cluster database or its instances:

1. In the Navigator pane, expand Databases.
2. Right-click on a cluster database.
A menu appears with options for the database.
3. Choose **Shutdown** from the menu.
The Cluster Database Shutdown dialog box appears.
4. Select a shutdown type, as shown in [Table 5–3](#).

Table 5–3 Shutdown Types

Option	Description
Normal	Waits for the currently connected users to disconnect from the database, prohibits further connects, and closes and dismounts the database before shutting down the instance. Instance recovery is not required on next startup.
Immediate	<i>(default)</i> Does not wait for current calls to complete, prohibits further connects, and closes and dismounts the database. The instance is immediately shut down. Connected users are not required to disconnect and instance recovery is not required on next startup.
Abort	Proceeds with the fastest possible shutdown. Connected users are not required to disconnect. The database is not closed or dismantled, but the instances are shut down. Instance recovery is required on next startup. Note: You must use this option if a background process terminates abnormally.
Transactional	Waits for transactions to complete before shutting down
Shutdown Database Only	<i>(default)</i> Shuts down the database only. The services required for an instance, such as the listener, remain up and available.
Shutdown Database And Other Services	Shuts down the database and associated services, such as the listener

5. To shut down all instances, click **Shutdown**.
To shut down only selected instances, follow these additional steps:
 - a. Select **Instances**.

The Select Instances to Stop dialog box appears.

- b. Select the instances to stop in the Available list, then click **Add**.
- c. Click **OK** to close the Select Instances to Stop dialog box.
- d. Click **Shutdown** from the Cluster Database Shutdown dialog box.

The Cluster Database Shutdown Progress dialog box displays the progress of the shutdown operation.

See Also: ["Viewing the Cluster Database Operation Results"](#) on page 5-11

If the instances were shut down successfully, the Cluster Database Stopped message box appears with a successful message.

Click **OK** in the Cluster Database Stopped message to acknowledge the message, then click **Close** in the Cluster Database Shutdown Results.

If the shutdown fails, the Cluster Database Stopped message box appears with a failure message. Click **View Details** to view more information in the Cluster Database Shutdown Progress dialog box about why the shutdown failed, then click **Close**.

Viewing the Cluster Database Operation Results

The Cluster Database Startup/Shutdown Results dialog displays information about the progress of the instance startup or shutdown operation you selected:

The operation results are presented in two views:

- [Status Details Tab](#)
- [Output Tab](#)

The Cluster Database Startup/Shutdown Results dialog box automatically displays during a startup or shutdown operation. You can also initiate it with the following steps:

1. In the Navigator pane, expand Databases.
2. Right-click on a cluster database.
A menu appears with options for the database.
3. Choose Results from the menu.

Status Details Tab

While a startup or shutdown operation is running against a cluster, the Status Details tab progress display is shown and updated dynamically as the operation progresses.

A successful startup operation for a three-node cluster looks like the following in the Status Details tab, as shown in [Figure 5-4](#).

Figure 5-4 Status Details Tab



A successful shutdown operation for a three-node cluster looks like the Status Details tab shown in [Figure 5-5](#).

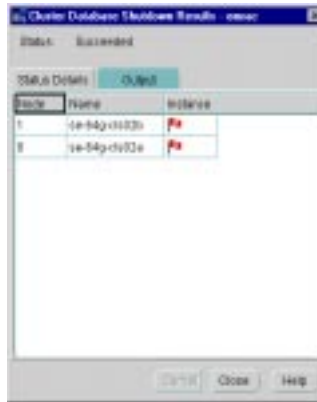
Figure 5–5 Successful Shutdown Results

Table 5–5 shows the possible states that each component might experience.

Table 5–4 Possible Component States

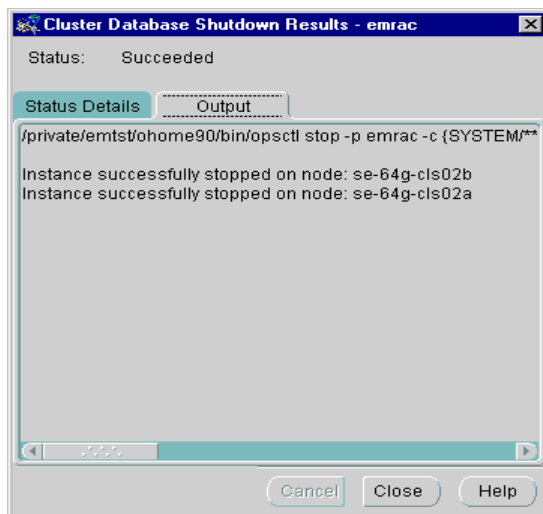
State	Description
Up (green flag)	The component is running.
Down (red flag)	The component is not running.
In Progress (timer)	Oracle Enterprise Manager cannot determine the state of the component. This state occurs typically when the component startup or shutdown operation has not completed.
Component does not exist on this node (blank background)	The component was not configured on the node. Not all components (listener, instance) are required on every node.

Output Tab

The Output tab displays the commands executed by the node and any associated error messages in textual format.

A successful shutdown looks like the following in the Output tab, as shown in [Figure 5-6](#).

Figure 5-6 *Successful Shutdown Results*



Oracle Enterprise Manager displays a Results tab that is similar to the Shutdown tab in [Figure 5-6](#).

Viewing Cluster Database Status

The Edit Cluster Database dialog box displays status information about the database, such as instances available in the cluster and the status of cluster components.

Note: Because this dialog box requires a connection to a database, this dialog box will not appear if the cluster database is down.

To view status information about a database:

1. In the Navigator pane, expand Databases > *database_name*.
2. Right-click on a cluster database under the Databases folder in the Navigator pane.

A menu appears with options for the database.

3. Choose **Edit** from the menu.

The Edit Cluster Database dialog box appears.

The operation results are presented in two views:

- [General Tab](#)
- [Status Details Tab](#)

General Tab

The General tab displays information about the currently running instances by querying `V$ACTIVE_INSTANCES` table, as shown in [Figure 5-7](#). Oracle Enterprise Manager makes a connection to the cluster database. Therefore, this tab will not appear if the database is down.

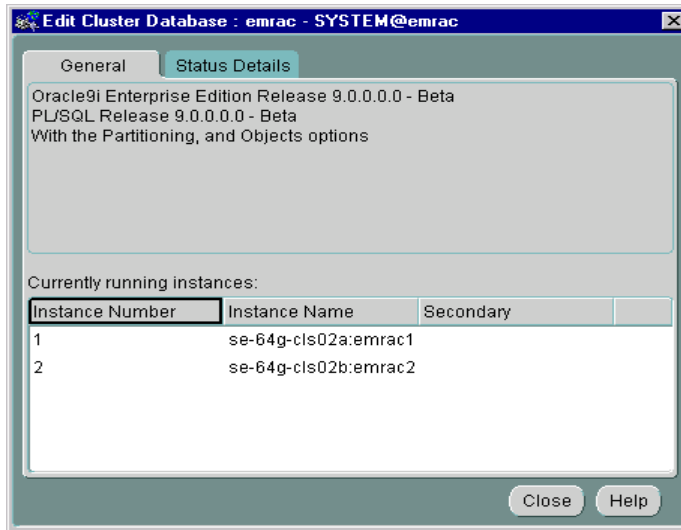
Figure 5–7 General Tab

Table 5–5 describes the fields in the Output Tab Form.

Table 5–5 Fields in Output Tab Form

Column Name	Description
Instance Number	Identifies the instance number
Instance Name	The name specified for the instance and the node it is running on. This name has the following format: <i>node:instance_name</i> .
Secondary	Indicates if the node is an secondary instance in a primary/secondary instance configuration

Status Details Tab

Note: This tab is not available for a Windows NT or Windows 2000 cluster database because SRVCTL on Windows NT and Windows 2000 does not generate status details.

The Status Details Tab displays an overall view of the state of the cluster and related components, as shown in [Figure 5-8](#). This tab displays the status of the various components, such as listeners and instances, for all nodes.

Figure 5-8 Status Details Tab



[Table 5-6](#) shows the possible states that each component might experience.

Table 5-6 Component States

State	Description
Up (green flag)	The component is running.
Down (red flag)	The component is not running.
In Progress (timer)	Oracle Enterprise Manager cannot determine the state of the component. This state occurs typically when the component startup or shutdown operation has not completed.
Component does not exist on this node (blank background)	The component was not configured on the node. Not all components (listener, instance) are required to exist on every node.

Creating a Job for a Cluster Database or Instance

The job scheduling system provides a highly reliable and flexible mechanism for you to schedule and automate repetitive jobs on both the cluster database and instances.

The Console contains a full-featured scheduling tool that enables you to develop a customized schedule. This provides you with actual “lights out” management capability so that you can focus on other tasks. A rich selection of jobs is provided for cluster databases.

You can create a job with a cluster database or instance as the destination. To create a new job, follow these steps:

1. Choose Job > Create Job.
2. Complete the tabs of the Create Job property sheet.
3. When you are satisfied with your job settings, click **Submit** to submit the job to Oracle Intelligent Agent. The job appears in the Active Jobs window.
4. Click **Save** to save the job. The job appears in the Job Library window. You can modify or submit a saved job at a later time.

Note: There is usually a slight delay between submitting the job and notification by Oracle Intelligent Agent.

Specifying Job Details

From the Create Job property sheet, you can specify the details of a new job. The Create Job property sheet contains the tabs listed in [Table 5-7](#).

Table 5-7 *Tabs Contained in the Create Job Property Sheet*

Tab	Description
General	Specify the job name, description, destination type, and destination.
Tasks	Choose the task(s) for the job.
Parameters	Set the run-time parameters for the tasks. The parameters that appear on this tab depend on which task(s) you chose on the Task list box.

Tab	Description
Schedule	Schedule the time and frequency for Oracle Enterprise Manager to run the job.
Permissions	Specify the administrator to perform the job.

The following tabs contain options specific to cluster databases:

- [General Tab](#)
- [Tasks Tab](#)
- [Parameters Tab](#)

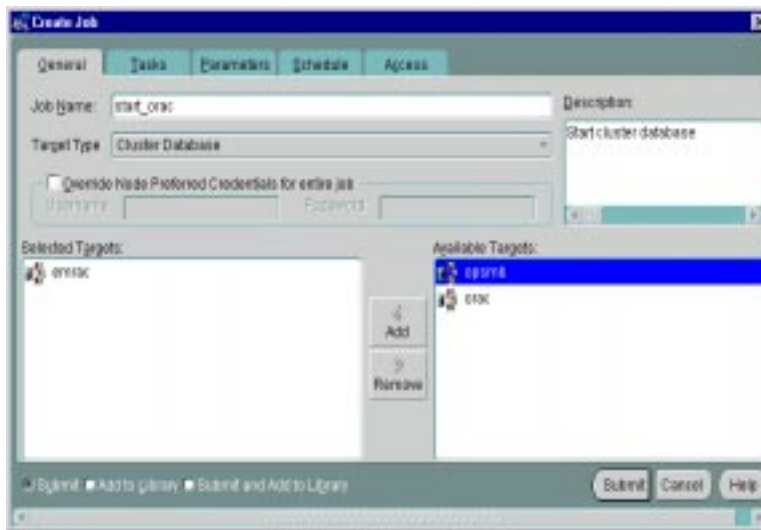
See Also: *Oracle Enterprise Manager Administrator's Guide* for general job scheduling information

General Tab

As shown in [Figure 5-9](#), from the General tab you can specify:

- Job Name
- Destination Type (either Cluster Database or Instance)
- Description
- Destinations

Figure 5–9 General Tab



The General tab contains the options in [Table 5–8](#).

Table 5–8 General Tab Options

Option	Description
Job Name	Enter the name of the new job.
Description	Enter a description of the job.
Destination Type	Select a destination type from the drop-down list box. You can select from the following options: Cluster Database, instance, database, listener, or node.
Available Destinations	<p>The destinations are determined by your selection of the Destination Type. The destinations include Cluster Databases, instances, databases, listeners, and nodes.</p> <p>Click the destinations of the job in the Available Destinations list, then click Add to move the destination to the Selected Destinations list. To remove a destination from a job, click the destination in the Selected Destinations list, then click Remove.</p>

Option	Description
Fixit Job	Select this check box to use this job as the fixit job for an event occurrence. The job cannot be scheduled.

Tasks Tab

From the Tasks tab, choose the task(s) for the job to perform. The list of tasks that appear is different depending on whether you select a cluster database or instance as your Destination Type from the General tab.

Move the tasks between the Available Tasks and Selected Tasks lists with the Add and Remove buttons.

Tasks for Cluster Database Destinations

If your Destination Type is a cluster database, you can select from these tasks that are specific to Real Application Clusters:

- Shutdown Cluster Database
- Startup Cluster Database

See Also: *Oracle Enterprise Manager Administrator's Guide* for a description of these tasks and the parameters to set

Parameters Tab

From the Parameters tab, you can specify parameter settings for the job tasks you selected on the Tasks tab. The parameters that display vary according to the job task. Parameters specific to cluster database startup and shutdown tasks are described below.

See Also: *Oracle Enterprise Manager Administrator's Guide* for a description of parameters to set for instance tasks

Cluster Database Startup Task

When you select the **Startup Cluster Database** task on the Tasks tab, the following display appears.

Complete the parameters on the tab and click **Submit** to run the cluster database startup task.

The Parameters tab for Startup contains the options listed in [Table 5–9](#).

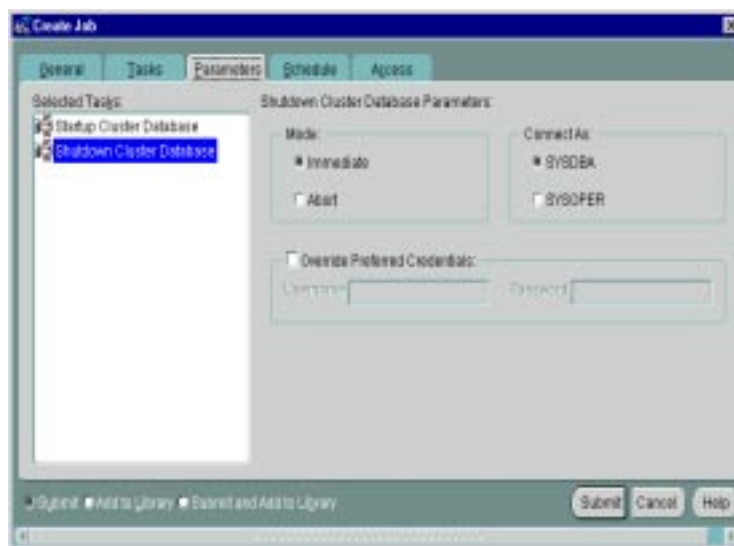
Table 5–9 Parameters Tab for Startup

Parameters	Description
Startup	Select the startup modes for the job from the drop-down list box.
Connect As	<p>Specify the role.</p> <p>Only Normal is allowed for Oracle7. For Oracle8i and subsequent releases, SYSOPER and SYSDBA roles allow you the maximum database administration privileges. You require SYSDBA or SYSOPER privileges to run job tasks such as shutdown or startup on the database.</p> <p>See Also: <i>Oracle9i Database Administrator's Guide</i> for more information about SYSOPER and SYSDBA roles</p> <p>If you attempt to connect as SYSDBA and do not have SYSDBA privileges, an error message states that an invalid user name or password was entered.</p>
Override Preferred Credentials	You can use the preferred credentials that have been set up for the database, or you can enter another database user name and password.

Parameters for the Cluster Database Shutdown Task

When you select **Shutdown Cluster Database** on the Tasks tab, the following display in [Figure 5–10](#) appears.

Figure 5–10 Shutdown Cluster Database Parameters



Complete the parameter entries on the tab and click **Submit** to run the cluster database shutdown task.

The Parameters tab for Shutdown contains the options listed in [Table 5–10](#).

Table 5–10 Parameters Tab for Shutdown

Options	Description
Mode	Click Immediate (<i>default</i>) or the Abort.
Connect As	Click SYSDBA (<i>default</i>) or SYSOPER.
Override Preferred Credentials	You can use the preferred credentials that have been set up for the database, or you can use another database user name and password.

Registering Cluster Database Events

Oracle Enterprise Manager includes an Event Management feature. This feature enables you to monitor tests on cluster database instances. All of the tests that are available for single instance databases can also be run on cluster databases. In addition to these standard event tests, there are specialized event tests available only for cluster database instances. These tests include: global cache converts, pings, timeouts, consistent read requests, freelist waits, and so forth. Data from the **Oracle Data Gatherer** is used to create these Event Management reports.

To access the Event Management feature, from the Event menu, select **Create Event**. The Create Event window opens. There, you can select targets, set parameters, and set notification preferences, using the following sub-pages:

- The General Page is used to select the cluster database and cluster database instance target types.
- The Test Page is used select the type of specialized cluster databases tests that you want to schedule.
- The Parameters Page, is used to select the parameter for your test event.
- The Schedule Page is used for scheduling the execution of a test event.
- The Access Page is used for setting notification options. By default, notifications are sent directly to the SYSADMIN console. Depending on your hardware configuration, you can also select notification through a telephone pager and/or e-mail.

Typically, you will set parameters and notifications so that the system will inform you when a given parameter limit has been reached. The Event Management Tests submenu is shown in [Figure 5-11](#).

Figure 5-11 *Event Management Tests Menu*



Part III

Backup and Recovery in Real Application Clusters

Part Three discusses implementing backup and recovery in Real Application Clusters environments. Part Three also contains information about configuring RMAN for Real Application Clusters. The chapters in Part Three are:

- [Chapter 6, "Configuring RMAN for Real Application Clusters"](#)
- [Chapter 7, "Backing Up Real Application Clusters Databases"](#)
- [Chapter 8, "Database Recovery in Real Application Clusters"](#)

Configuring RMAN for Real Application Clusters

This chapter explains how to configure **Recovery Manager (RMAN)** for **Real Application Clusters** environments. The topics in this chapter include:

- [Configuring RMAN in Real Application Clusters: Overview](#)
- [Selecting An Archived Log Configuration Scheme](#)
- [Shared Read Local Archive Logging](#)
- [Implementation Considerations of Archive Logging Schemes](#)

Configuring RMAN in Real Application Clusters: Overview

Before using Recovery Manager in Real Application Clusters, review the following documentation.

- *Oracle9i Recovery Manager User's Guide*
- *Oracle9i Recovery Manager Reference*
- *Oracle Enterprise Manager Administrator's Guide* for information about using **Oracle Enterprise Manager** for recovery

See Also: The RMAN online documentation roadmap and the Quick Start and Configuration chapters in the *Oracle9i Recovery Manager User's Guide*

After installing the **Oracle Enterprise Edition** with the **Oracle9i Real Application Clusters** license, you can configure RMAN for use after following the configuration procedures in this chapter. Before configuring RMAN, however, decide whether to use a recovery catalog as described in the RMAN documentation. You also can use RMAN without a catalog and add one later. If you use a catalog, be sure to register your Real Application Clusters database in the RMAN recovery catalog.

The first configuration step is to specify the default location of the **snapshot control file**. RMAN creates a snapshot control file as described in this chapter to obtain a consistent picture of the target control file before making a backup, and RMAN must create the snapshot control file on the **node** that makes the backup. Also, you can enable the control file autobackup feature, which causes RMAN to automatically make backups of the control file after BACKUP and COPY commands.

The aspect of RMAN use that requires the careful planning is the configuration of the archived logs. When Oracle generates an archived log, it records the filename of the log both in the control file of the target database and in the recovery catalog. RMAN accesses the archived log files by this filename regardless of which node runs the backup. Configure the RMAN environment by sharing the archived log directories in Real Application Clusters environments as explained in this chapter.

Configuring RMAN Control File Snapshots and Autobackups

The CONFIGURE command creates persistent settings that are in effect across RMAN sessions. Thus, after running these commands, you do not need to run them again unless you want to change the default configuration.

This section contains these topics:

- [Configuring the Snapshot Control File Location in RMAN](#)
- [Configuring the Control File Autobackup Feature in RMAN](#)

Configuring the Snapshot Control File Location in RMAN

In Real Application Clusters, configure the location for your snapshot control files before performing backups using RMAN. The snapshot control file must exist on the node from which you perform backups. Therefore, on all nodes used for backups, ensure that the same destination directory for a snapshot control file exists.

For example, on UNIX specify that the snapshot control file should be written to the file `/oracle/db_files/snapshot/snap_prod.cf` as follows:

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/db_files/snapshot/snap_prod.cf';
```

This RMAN configuration is globally set across all nodes in the **cluster**. Therefore, ensure that the directory `/oracle/db_files/snapshot` exists on all nodes that make backups.

You can also specify a raw device destination for a snapshot control file. This raw device destination is shared across all nodes in the cluster just like other **data files** in Real Application Clusters.

See Also: *Oracle9i Recovery Manager User's Guide* for more information on configuring the RMAN environment

Configuring the Control File Autobackup Feature in RMAN

If you set `CONFIGURE CONTROLFILE AUTOBACKUP` to `ON`, then RMAN makes an automatic control file backup after you execute the `BACKUP` or `COPY` commands. RMAN gives this backup a default name. However, you can change this name with the `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT` command. RMAN can restore this control file and even autobackup after the loss of both the recovery catalog and the current control file.

RMAN performs the control file autobackup on the first allocated channel. Thus, when you allocate multiple channels with different parameters, especially when you allocate a channel with the `CONNECT` command, you must determine which channel will perform the `CONTROLFILE AUTOBACKUP`. Always allocate the channel for this node first.

See Also: *Oracle9i Recovery Manager User's Guide* for more information on configuring the RMAN environment

The rest of this chapter describes various archived log configuration schemes for implementing RMAN in Real Application Clusters databases. This chapter also explains each schemes' advantages and disadvantages and provides configuration details.

Selecting An Archived Log Configuration Scheme

Carefully select the archived log configuration scheme that best meets your requirements. The archivelog scheme you use is critical for RMAN. With RMAN, you can distribute backup operations among all the nodes in the cluster, but only one node performs recovery operations. The node performing recovery must have access to all of the logs created by all nodes.

The source for archived log information is the target database control file. If you use a recovery catalog, then the log information in the control file is imported by RMAN into the catalog. Depending on your configuration, RMAN selects the logs to backup or recover by referring to the control file or the RMAN catalog to determine the log filenames generated by the nodes.

Each node in the configuration reads the common control file to obtain log filenames, but each node also looks for log filenames on its own file system. Therefore, the node performing the backup or recovery must be able to read the filename for each log listed in the repository to use the log. If every node in the configuration must be able to perform backup and recovery, then every node must be able to read all log filenames.

A problem can occur if you configure each node to write to a local directory that has the same path name as the local archiving directory on the other nodes in your Real Application Clusters database. For example, assume that each node archives to its local directory called `/arc_dest`. If you use identical directory names, the locations of the RMAN logs appear in the control file as in the following example:

```
/arc_dest/log_1_62  
/arc_dest/log_2_100  
/arc_dest/log_2_101  
/arc_dest/log_3_70  
/arc_dest/log_1_63
```

If node 1 attempts to access these logs for backup or recovery, then node 1 searches its local `arc_dest` directory for the required logs as they were recorded in the

control file. Using this information, node 1 only finds the logs that it archived in its local directory. In this example, node 1 can find log 62 and 63, but it cannot locate the remaining logs. The various configuration schemes described in this chapter solve this problem so that RMAN can access the logs that it needs for backup and recovery.

RMAN Archive Log Configuration Schemes

The most common archive logging schemes belong one of the following six categories:

- [Non-Shared Local Archive Logging](#)
- [Shared Read Local Archive Logging](#)
- [One Remote Archive Logging](#)
- [Cross Shared Cross Writing with One Local and N Remote Archive Logging](#)
- [Central NFS Directory for High Availability Archive Logging](#)
- [Hybrid Archive Logging](#)

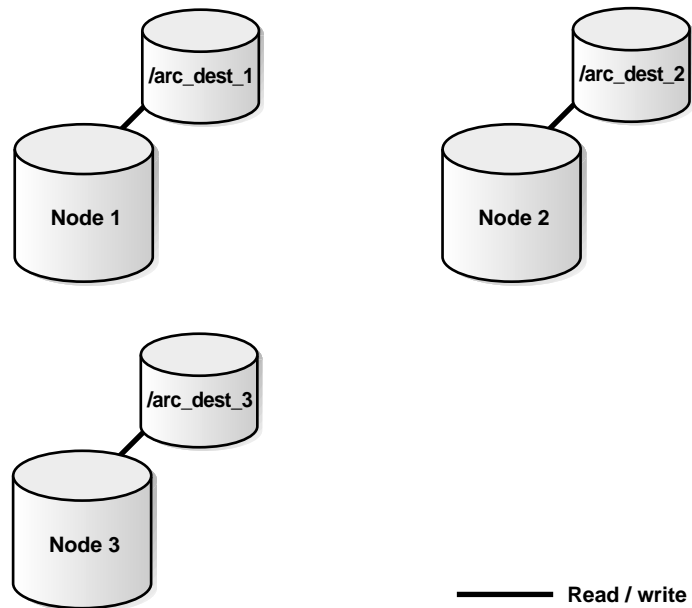
The following sections describe each of these schemes in ascending order of reliability from least reliable to most reliable. The sample schemes assume that you have a three-node Real Application Clusters database.

Note: Oracle Corporation recommends using multiple archiving destinations, with at least one destination on a remote host.

Non-Shared Local Archive Logging

In the non-shared scheme, each node writes to a local archived log file as shown in [Figure 6-1](#).

Figure 6-1 Non-Shared Local Archiving Configuration Scheme



Each archive destination requires a unique name so that RMAN can distinguish the archived logs for backups and restores. This configuration is useful if every node has a local tape device. Otherwise, although the reading is local on every node, writing to tape must go through the network.

If one node fails, transactions that were logged after the last full backup are lost. Therefore, the node that fails becomes a single point of failure. For this reason, Oracle Corporation does *not* recommend that you implement non-shared local configurations for RMAN in Real Application Clusters.

In the scenario shown in [Figure 6-1](#), the archived log records may look something like the following:

```
/arc_dest_1/log_1_62
/arc_dest_2/log_2_100
/arc_dest_2/log_2_101
/arc_dest_3/log_3_70
/arc_dest_1/log_1_63
```

Because each node archives to a different directory, the path names of the logs in the repository are different. Thus, if you run RMAN on node 1 and you do not use [network file system \(NFS\)](#) to map local directories to directories on the other nodes, then the archivelog backup step in the following script fails because RMAN cannot access the archived redo logs from node 2:

```
% RMAN TARGET / CATALOG RMANUSER/RMANPASS@RMANCAT

RUN
{
  ALLOCATE CHANNEL dbltape1 DEVICE TYPE sbt;
  BACKUP DATABASE;
  BACKUP ARCHIVELOG ALL;
}
```

The configuration scheme in this section prevents this type of failure. To configure this scheme, perform the following one-time configuration so that one channel is configured for each node in the cluster. Do this at the RMAN prompt by entering:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'user1/password1@node1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'user2/password2@node2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT 'user3/password3@node3';
```

Similarly, you can set up this configuration for a device type of DISK.

Note: As mentioned, this is a one-time configuration step; you do not need to do this at every backup.

Configuring Initialization Parameters for Non-Shared Archiving Destinations

To configure initialization parameters for non-shared archiving destinations, create the local archiving destination on each node. You do not need to use NFS or shared drives for backups. However, during recovery, the node performing recovery must

be able to read all the archived logs from all nodes in the cluster. In this case, you may still need use NFS or shared drives for recovery.

In UNIX, for example, make the following initialization parameter settings so that each node is archiving to a path that is distinct from the other archiving paths:

```
db1.LOG_ARCHIVE_DEST_1="LOCATION=?/admin/db_name/arc_dest1"  
db2.LOG_ARCHIVE_DEST_1="LOCATION=?/admin/db_name/arc_dest2"  
db3.LOG_ARCHIVE_DEST_1="LOCATION=?/admin/db_name/arc_dest3"
```

Note: If Oracle reconstructs the names of archived **redo log files**, the format that LOG_ARCHIVE_FORMAT specifies for the **instance** performing recovery must be the same as the format specified for the instances that archived the files. In Real Application Clusters, all instances should use the same value of LOG_ARCHIVE_FORMAT. Moreover, the instance performing recovery should also use that value. You can specify a different value of LOG_ARCHIVE_DEST_n during recovery if the archived redo log files are not at their original archive destinations.

Backups Using Non-Shared Archive Logging

You should develop a production backup script for whole database backups that you can run from any node. In the script, make sure that the channel allocated on each node only backs up the logs that are located on the node. The following example uses automatic channels to make a database and archived log backup:

```
BACKUP DATABASE  
  (ARCHIVELOG LIKE '%arc_dest_1%' CHANNEL ORA_SBT_TAPE_1)  
  (ARCHIVELOG LIKE '/arc_dest_2/' CHANNEL ORA_SBT_TAPE_2)  
  (ARCHIVELOG LIKE '/arc_dest_3/' CHANNEL ORA_SBT_TAPE_3)
```

The following script uses automatic channels to back up only archived logs:

```
BACKUP  
  (ARCHIVELOG LIKE '/arc_dest_1/' CHANNEL ORA_SBT_TAPE_1)  
  (ARCHIVELOG LIKE '/arc_dest_2/' CHANNEL ORA_SBT_TAPE_2)  
  (ARCHIVELOG LIKE '/arc_dest_3/' CHANNEL ORA_SBT_TAPE_3)
```

Restore and Recovery Using Non-Shared Archive Logging

To restore and recover the whole database, make all the archived logs that were not backed up accessible by the node from which you are performing the restore and recovery. Do this by using one of the following strategies:

- Using NFS or configuring shared drives on Windows platforms and making the logs available to the recovery node with the same name as the remote name
- Restoring the archived logs onto a disk accessible by the node performing the recovery

For example, if node 3 is performing the recovery, then you can use the following script, where *directory* is a directory on node 3 into which you will restore the archived logs generated by node 1 and node 2. This example assumes that the channels are set up as in the previous example, where one channel will be allocated each node:

```
RUN
{
SET AUTOLOCATE ON;
RESTORE DATABASE;
# set the archiving destination to the directory
# where RMAN will restore the logs it needs for recovery. Note that this
# directory must be read-write accessible by all nodes in the cluster.
SET ARCHIVELOG DESTINATION TO directory;
RECOVER DATABASE;
}
```

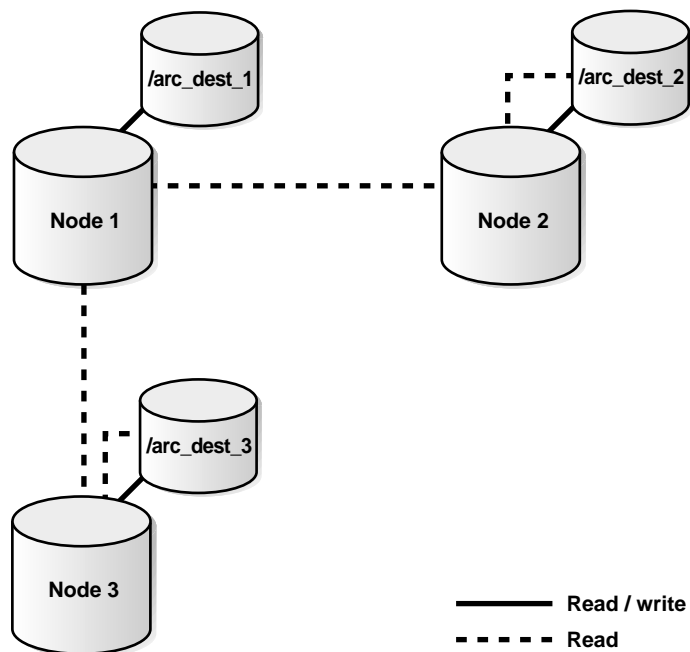
To only restore the archived logs (not the database) in this archiving scheme, you can use the following script:

```
RUN
{
SET AUTOLOCATE ON;
# set the archiving destination to the directory containing all needed logs
SET ARCHIVELOG DESTINATION TO directory;
RESTORE ARCHIVELOG ALL;
}
```

Shared Read Local Archive Logging

In the shared read local archiving scheme, each node writes to a local archived log file and can read the archived log files of the other nodes as shown in [Figure 6-2](#). Read access is commonly achieved using NFS on UNIX or shared drives on Windows platforms.

Figure 6-2 Shared Read Local Archiving Configuration Scheme



This configuration is useful only if a few nodes have tape devices and if you perform the backups from those nodes.

Note: For simplification in this and the remaining configuration examples, only Node 1 is shown with the read/write access to the archived logs. When you configure a scenario, you must configure access for all nodes as described in the remaining examples.

Consider a UNIX example in which you are using the following configuration for the scheme in [Figure 6-2](#):

- Node 1 archives to `/arc_dest_1`
- Node 2 archives to `/arc_dest_2`
- Node 3 archives to `/arc_dest_3`

Node 1 has the following archived log configuration:

```
/arc_dest_2 is a symbolic link to /arc_dest_2 on node 2  
/arc_dest_3 is a symbolic link to /arc_dest_3 on node 3
```

Node 2 has the following archived log configuration:

```
/arc_dest_1 is a symbolic link to /arc_dest_1 on node 1  
/arc_dest_3 is a symbolic link to /arc_dest_3 on node 3
```

Node 3 has the following symbolic links:

```
/arc_dest_1 is a symbolic link to /arc_dest_1 on node 1  
/arc_dest_2 is a symbolic link to /arc_dest_2 on node 2
```

For this scheme, the control file has the following entries:

```
/arc_dest_1/log_1_62  
/arc_dest_2/log_2_100  
/arc_dest_2/log_2_101  
/arc_dest_3/log_3_70  
/arc_dest_1/log_1_63
```

In this case, any node performing the recovery can read the control file and access the logs on the other nodes by means of the symbolic links. For example, if node 2 is performing recovery and needs to access `/arc_dest_1/log_1_62`, then it is going to look for it in the `/arc_dest_1` directory on its local file system. Because `/arc_dest_1` on node 2 is a symbolic link pointing to `/arc_dest_1` on node 1, it can access the log remotely.

For this scheme and the remaining schemes that use sharing, to enable RMAN to back up and recover a Real Application Clusters database in one step, all archived logs must have uniquely identifiable names throughout the Real Application Clusters database. In addition, you must configure every node so that it can identify and access all the archived logs throughout your cluster using these same unique names.

Note: For UNIX environments only: The archived log files cannot be on the shared disk because UNIX shared disks are raw devices that you cannot easily partition for use with archived logs.

Although this scheme offers simplified manageability over non-shared local configurations, shared read local archiving configurations also have single points of failure. If one node fails after the last complete backup, the archived logs located on that node are unavailable to the surviving nodes for recovery.

Note: You do not have to use a particular configuration for this scheme. However, to distribute the backup processing onto multiple nodes, the easiest method is to configure channels. For more information about this, refer to the RMAN documentation.

The production script for whole database backups from any node is:

```
BACKUP DATABASE PLUS ARCHIVELOG;
```

To back up just the archived logs and to delete them after the backup, use the following script:

```
BACKUP ARCHIVELOG ALL DELETE INPUT;
```

To restore and recover the whole database from any node, use the script:

```
RESTORE DATABASE;  
RECOVER DATABASE;
```

To only restore the archived logs if the remote destinations are writable, use the script:

```
RESTORE ARCHIVELOG ALL;
```

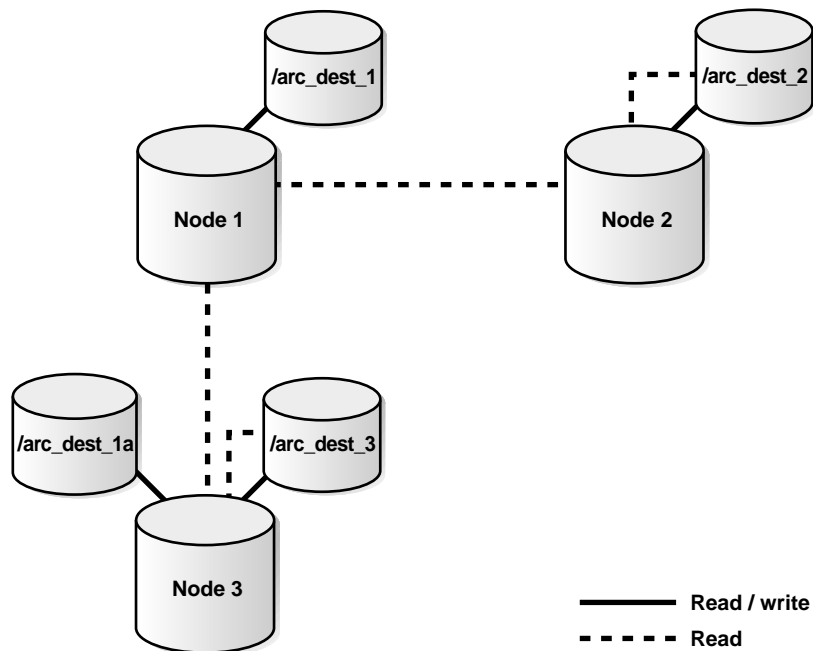
If the remote archived log destinations are not writable, then enter the following, where *directory* is the directory into which you want to restore the archived logs:

```
RUN  
{  
  SET ARCHIVELOG DESTINATION TO directory;  
  RESTORE ARCHIVELOG ALL;  
}
```

One Remote Archive Logging

Figure 6–3 shows a one remote archive logging scheme in which each node writes to both a local archived log destination and to a remote archived log destination on one of the other nodes.

Figure 6–3 One Remote Archive Logging Scheme



In this example, the following occurs:

- Node 1 archives locally and also remotely to a directory on node 2
- Node 2 archives locally and also remotely to a directory on node 3
- Node 3 archives locally and also remotely to a directory on node 1

The one remote archive logging scheme is more reliable than the previous two schemes. This is because each node writes to both a local archived log file as well as to remote log files. As with shared read local archiving, each archived log within the Real Application Clusters environment must have a unique path name.

For the remote destination disk, Oracle Corporation recommends that you logically arrange your nodes in a circular sequence. Do this to allow the first node to write to second node, the second node to write to the third node, and so on. The last node must write to the first node. This way, each node writes to a local archived log file as well as to a remote archived log.

Recovery does not have a single point of failure in this scheme unless you are backing up to tape and only one node can access the tape device. Multiple archived log destinations also avoid single-points-of-failure by making the archived logs for a failed node available to other nodes for recovery processing.

If you use NFS to link the remote logs to each node, then you may experience some performance degradation. This occurs because each node must write to its remote archived log files by way of the network. Moreover, if you make the NFS directory mandatory and the network fails, your database also stops. Avoid this by making the NFS directory optional. However, in this case you risk losing synchronization among the archived logs.

You must also decide whether to use remote reading so that each node can read a directory on every other node. This scheme is known as cross-mounting, and is illustrated in the following example:

- Node 1 can read directories on both node 2 and node 3
- Node 2 can read directories on both node 1 and node 3
- Node 3 can read directories on both node 1 and node 2

You can achieve cross-mounting using either NFS on UNIX or shared drives on Windows platforms.

Configuring Initialization Parameters for One Remote Archive Logging

To configure parameters for one remote archive logging, create a local and a remote archived log directory. For example on UNIX, on node 1 create the following directory structure:

```
$ORACLE_HOME/admin/db1/arc_dest1  
$ORACLE_HOME/admin/db1/arc_dest2 # remote NFS destination on node 2
```

Create the following directory structure on node 2:

```
$ORACLE_HOME/admin/db2/arc_dest2  
$ORACLE_HOME/admin/db2/arc_dest3 # remote NFS destination on node 3
```

Create the following directory structure on node 3:

```
$ORACLE_HOME/admin/db3/arc_dest3
$ORACLE_HOME/admin/db3/arc_dest1 # remote NFS destination on node 1
```

Continue by creating a similar directory structure for every node. Then use the following **initialization parameter file** settings:

```
db1.LOG_ARCHIVE_DEST_1="LOCATION=?/admin/db_name//arc_dest1"
db1.LOG_ARCHIVE_DEST_2="LOCATION=?/admin/db_name//arc_dest2"
db2.LOG_ARCHIVE_DEST_1="LOCATION=?/admin/db_name//arc_dest2"
db2.LOG_ARCHIVE_DEST_2="LOCATION=?/admin/db_name//arc_dest3"
db3.LOG_ARCHIVE_DEST_1="LOCATION=?/admin/db_name//arc_dest3"
db3.LOG_ARCHIVE_DEST_2="LOCATION=?/admin/db_name//arc_dest1"
```

On Windows using shared drives, create the following shared drives on node 1:

```
J:\
K:\ # maps to K: drive on node 2
```

Create the following shared drives on Node 2:

```
K:\
L:\ # maps to L: drive on node 3
```

Create the following shared drives on Node 3:

```
L:\
J:\ # maps to J: drive on node 1
```

Then use the following initialization parameter file settings:

```
db1.LOG_ARCHIVE_DEST_1="LOCATION=J:\archivelogs"
db1.LOG_ARCHIVE_DEST_2="LOCATION=K:\archivelogs"
db2.LOG_ARCHIVE_DEST_1="LOCATION=K:\archivelogs"
db2.LOG_ARCHIVE_DEST_2="LOCATION=L:\archivelogs"
db3.LOG_ARCHIVE_DEST_1="LOCATION=L:\archivelogs"
db3.LOG_ARCHIVE_DEST_2="LOCATION=J:\archivelogs"
```

Backups for One Remote Archive Logging

Using cross-mounting simplifies your backup scripts for one remote archive logging. However your system may incur network processing overhead. If you are backing up to tape, cross-mounting is preferable if only a few of the nodes have local tape devices.

If you do not use cross-mounting, your scripts must be more complex. Not using cross-mounting is only beneficial if all of the nodes have local tape devices.

Regardless of whether you use cross-mounting, you can optimize your backup scripts so that each node locally reads two archived log destinations. To do this, use customized scripts on each node; in this case you cannot use one generalized script for all the nodes.

No Cross-Mounting Archive Logging

To configure the no cross-mounting scheme, perform the following one-time configuration. To do this, at the RMAN prompt enter:

```
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'user1/password1@node1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'user2/password2@node2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT 'user3/password3@node3';
```

Backups for No Cross-Mounting

If you use this configuration, then your production backup script for whole database backups is:

```
BACKUP DATABASE
  (ARCHIVELOG LIKE '/arc_dest1' CHANNEL ORA_SBT_1)
  (ARCHIVELOG LIKE '/arc_dest2' CHANNEL ORA_SBT_2)
  (ARCHIVELOG LIKE '/arc_dest3' CHANNEL ORA_SBT_3);
```

You can run this script from any node. To only backup the archived logs, use this script:

```
BACKUP
  (ARCHIVELOG LIKE '/arc_dest1' CHANNEL ORA_SBT_1)
  (ARCHIVELOG LIKE '/arc_dest2' CHANNEL ORA_SBT_2)
  (ARCHIVELOG LIKE '/arc_dest3' CHANNEL ORA_SBT_3);
```

Otherwise, if you do not want to perform the one-time configuration, then your production backup script for a whole database backup is:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt CONNECT 'node1';
  ALLOCATE CHANNEL c2 DEVICE TYPE sbt CONNECT 'node2';
  ALLOCATE CHANNEL c3 DEVICE TYPE sbt CONNECT 'node3';
BACKUP DATABASE
  (ARCHIVELOG LIKE '/arc_dest1' CHANNEL ORA_SBT_1)
  (ARCHIVELOG LIKE '/arc_dest2' CHANNEL ORA_SBT_2)
```

```
(ARCHIVELOG LIKE '/arc_dest3' CHANNEL ORA_SBT_3);
}
```

Restore and Recovery for No Cross-Mounting

To restore and recover the whole database, make all the archived logs that were not backed up accessible by the node from which you are performing the restore and recovery operation. Even though you did not use cross-mounting for the backups, you could use it for the restore and recover operations. In this case, you must use *writable* cross-mounting. Or you can copy the archived logs onto a disk accessible by the node you are using. To copy the logs, use the following script:

```
RUN
{
SET AUTOLOCATE ON;
RESTORE DATABASE;
SET ARCHIVELOG DESTINATION TO directory;
RECOVER DATABASE;
}
```

Where *directory* is the directory into which you copied the archived logs. To restore only the archived logs in this scheme, use the following script:

```
RUN
{
SET AUTOLOCATE ON;
SET ARCHIVELOG DESTINATION TO directory;
RESTORE ARCHIVELOG ALL;
}
```

Cross-Mounting for Writing Archive Logging

You do not have to use a particular configuration to use a cross-mounting scheme. However, for more information see the RMAN documentation.

Configuring Initialization Parameters for Cross-Mounting for Writing

To configure initialization parameters for cross-mounting for writing, create the same directory structure for the archived logs on every instance. For a three-node cluster, one of the entries is the local archived log destination, and the other two entries are for the remote archived logs.

For example, on UNIX, create the following directory structure on each node.

```
$ORACLE_HOME/admin/db_name//arc_dest1  
$ORACLE_HOME/admin/db_name//arc_dest2  
$ORACLE_HOME/admin/db_name//arc_dest3
```

Then use the following initialization parameter settings:

```
db1.LOG_ARCHIVE_DEST_1="LOCATION=?/admin/db_name//arc_dest1"  
db1.LOG_ARCHIVE_DEST_2="LOCATION=?/admin/db_name//arc_dest2"  
db1.LOG_ARCHIVE_DEST_3="LOCATION=?/admin/db_name//arc_dest3"  
db2.LOG_ARCHIVE_DEST_1="LOCATION=?/admin/db_name//arc_dest2"  
db2.LOG_ARCHIVE_DEST_2="LOCATION=?/admin/db_name//arc_dest1"  
db2.LOG_ARCHIVE_DEST_3="LOCATION=?/admin/db_name//arc_dest3"  
db3.LOG_ARCHIVE_DEST_1="LOCATION=?/admin/db_name//arc_dest3"  
db3.LOG_ARCHIVE_DEST_2="LOCATION=?/admin/db_name//arc_dest2"  
db3.LOG_ARCHIVE_DEST_3="LOCATION=?/admin/db_name//arc_dest1"
```

On Windows using shared drives, create the following shared drives on each node:

```
J:\  
K:\  
L:\
```

Use the following initialization parameter settings:

```
db1.LOG_ARCHIVE_DEST_1="LOCATION=J:\archivelogs"  
db1.LOG_ARCHIVE_DEST_2="LOCATION=K:\archivelogs"  
db1.LOG_ARCHIVE_DEST_3="LOCATION=L:\archivelogs"  
db2.LOG_ARCHIVE_DEST_1="LOCATION=K:\archivelogs"  
db2.LOG_ARCHIVE_DEST_2="LOCATION=L:\archivelogs"  
db2.LOG_ARCHIVE_DEST_3="LOCATION=J:\archivelogs"  
db3.LOG_ARCHIVE_DEST_1="LOCATION=L:\archivelogs"  
db3.LOG_ARCHIVE_DEST_2="LOCATION=J:\archivelogs"  
db3.LOG_ARCHIVE_DEST_3="LOCATION=K:\archivelogs"
```

Each instance writes archived logs to its local archive directory and to the remote directories.

- For the first node, the entry for `/arc_dest1` is the local archived log destination for the instance. The other entries for `/arc_dest2` and `/arc_dest3` are the mount points for the archived logs for the instances on the second and third nodes.
- For the second node, the entry for `/arc_dest1` is a mount point for the remote archived log on the first node, the second entry for `/arc_dest2` is the local archived log directory, and the third entry for `/arc_dest3` is a mount point for the remote archived log on the third node.

- For the third node, the entry for `/arc_dest1` is a mount point for the remote archived log on the first node, the second entry for `/arc_dest2` is a mount point for the remote archived log on the second node, and the third entry for `/arc_dest3` is a mount point for the local archived log directory.

Note: You can use the `LOG_ARCHIVE_DEST_n` parameter to configure up to five archived log destinations.

See Also: *Oracle9i Database Reference* for more information about the `LOG_ARCHIVE_DEST_n` parameter

Backups for Cross-Mounting for Writing

The production script for whole database backups from any node is:

```
BACKUP DATABASE PLUS ARCHIVELOG;
```

To only backup the archived logs and to delete them after the backup, use the following script:

```
BACKUP ARCHIVELOG ALL DELETE INPUT;
```

Restore and Recovery for Cross-Mounting for Writing

To restore and recover the whole database from any node, use the following script:

```
RESTORE DATABASE;
RECOVER DATABASE;
```

To only restore the archived logs if the remote archived log destinations are writable, enter:

```
RESTORE ARCHIVELOG ALL;
```

If the remote archived log destinations are not writable, then enter:

```
RUN
{
SET ARCHIVELOG DESTINATION TO directory;
RESTORE ARCHIVELOG ALL;
}
```

Where *directory* is the directory into which you copied the archived logs.

Optimizing the Reading of Local Archivelog Destinations

To configure RMAN to read two local archived log destinations, manually partition the archived logs. For example, if you do not use cross-mounting and you are backing up from Node 3, use a script similar to the following:

```
BACKUP
  (ARCHIVELOG LIKE '/arc_dest3%')
  (ARCHIVELOG LIKE '/arc_dest1a%')
  (ARCHIVELOG LIKE '/arc_dest2%' CHANNEL Node 2);
```

In this case, the archived log destinations `/arc_dest3` and `/arc_dest1a` are local to Node 3. Archivelog `/arc_dest2` is backed up locally to Node 2. Change this script if you perform the backup from Node 2 to:

```
BACKUP
  (ARCHIVELOG LIKE '/arc_dest2%')
  (ARCHIVELOG LIKE '/arc_dest3a%')
  (ARCHIVELOG LIKE '/arc_dest1%' CHANNEL Node 1);
```

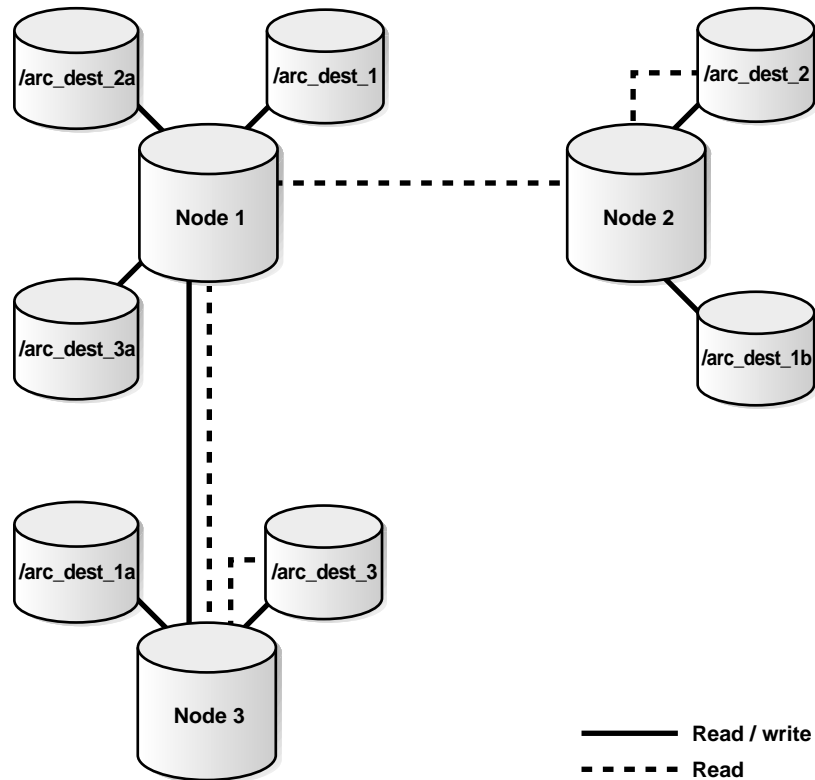
Use the following script if you use cross-mounting:

```
BACKUP
  (ARCHIVELOG LIKE '/arc_dest2%')
  (ARCHIVELOG LIKE '/arc_dest3%')
  (ARCHIVELOG LIKE '/arc_dest1%');
```

Cross Shared Cross Writing with One Local and N Remote Archive Logging

The cross shared cross writing with one local and n remote archiving scheme provides even greater reliability than the previous scheme. This is because each node writes to a local archived log file as well as to multiple remote log files as shown in Figure 6-4

Figure 6-4 Cross Shared Cross Writing, One Local and N Remote Archive Logging



For the example in [Figure 6-4](#), the archiving destinations are as shown in [Table 6-1](#):

Table 6-1 Cross Shared, Cross Writing, One Local N Remote Archiving Destinations Example

Node	Archives to...	Mounted on...
1	/arc_dest_1 (local)	-
1	/arc_dest_2 (mount point)	/arc_dest_2 on node 2
1	/arc_dest_3 (mount point)	/arc_dest_3 on node 3
2	/arc_dest_1	-
2	/arc_dest_2 (mount point)	/arc_dest_2 on node 1
2	/arc_dest_3 (mount point)	/arc_dest_3 on node 3
3	/arc_dest_1	-
3	/arc_dest_2 (mount point)	/arc_dest_2 on node 2
3	/arc_dest_3 (mount point)	/arc_dest_3 on node 3

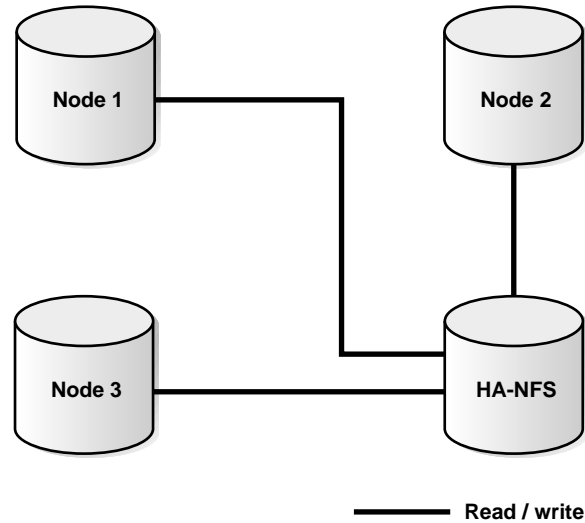
In this configuration, none of the nodes are single points of failure. As long as at least one node survives, complete recovery is possible. However, this scheme is limited for use on Real Application Clusters databases with no more than five nodes.

The configuration, backup, and restore scripts for this example are similar to the one remote archive logging scheme with cross-mounting, with added reliability.

Central NFS Directory for High Availability Archive Logging

Configuring a central **high availability** NFS (HA-NFS) directory requires that each node mount the same destination device for archive logging as shown in [Figure 6-5](#).

Figure 6-5 Central NFS Directory Configuration Scheme



In this example, the archiving destinations are as shown in [Table 6-2](#):

Table 6-2 Central NFS Directory Archive Log Example

Node	Archives to...	Mounted on...
1	/arc_dest_1 (mount point)	/arc_dest_1 on HA-NFS server
2	/arc_dest_1 (mount point)	/arc_dest_1 on HA-NFS server
3	/arc_dest_1 (mount point)	/arc_dest_1 on HA-NFS server

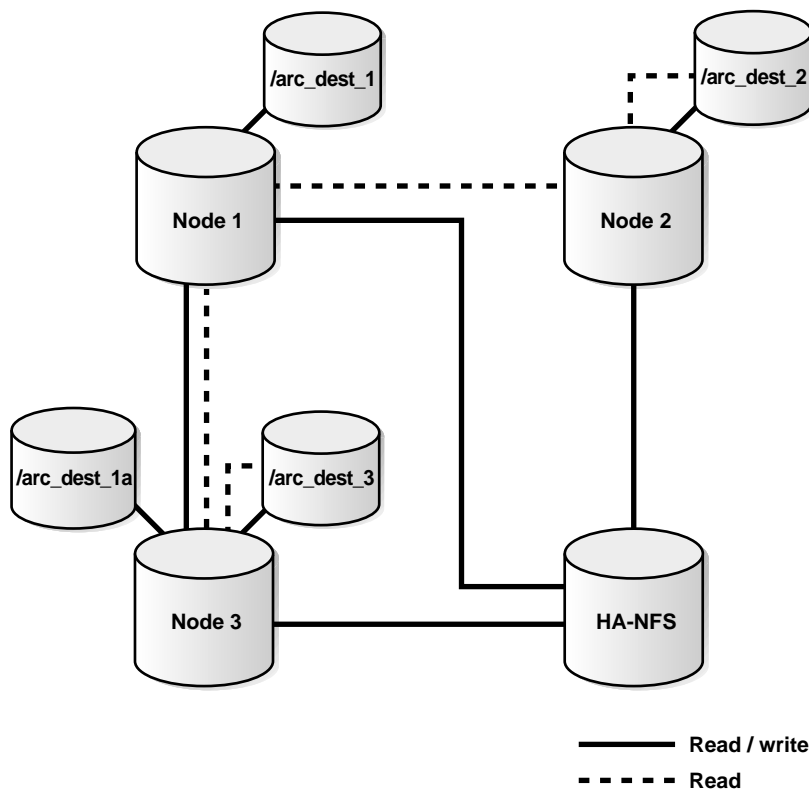
The only point of failure in this configuration is the HA-NFS node. If one or all of the other nodes fail, you can perform a complete recovery using the HA-NFS node.

The configuration, backup, and restore scripts for this example are similar to the one remote archive logging with cross-mounting scheme. You can distribute the archived log backup processing among the nodes, or perform it from one node.

Hybrid Archive Logging

For the greatest reliability, use a **hybrid** scheme that combines the advantages of the previous schemes. This enables you to reduce the number of or completely eliminate single points of failure. For example, you could implement HA-NFS and one remote logging with cross-mounting as shown in [Figure 6-6](#).

Figure 6-6 Hybrid Scheme Example



Your configuration, backup, and restore scripts will vary according to your implementation.

Implementation Considerations of Archive Logging Schemes

To support cross-mounting reading and cross-mounting writing, you must rely on your operating system's capabilities. These capabilities vary as described in the following topics:

- [Using NFS on UNIX Platforms](#)
- [Using Shared Drives on Windows Platforms](#)

Using NFS on UNIX Platforms

To use NFS, create one non-raw partition for each node and assign it to one node. Then create NFS mounts for all the the nodes so they can access all of the archived logs.

Exercise caution, however, when using NFS in Real Application Clusters environments. *Hard* NFS, which is the default, can block the entire cluster if the remote directories become inaccessible. This might occur as a result of a hardware failure. For this reason, Oracle Corporation strongly recommends that you use NFS implemented for high availability or soft-mounted NFS directories.

See Also: Contact your vendor to determine whether your hardware supports NFS and to determine the proper settings for the NFS mount

Soft-Mounted NFS Directories

Soft mounting means that a process attempting to access the mounted directory is not blocked until the directory becomes available after a failure. Consult your vendor documentation because the commands to configure this are operating system-dependent.

On Solaris, for example, create a soft mounted directory using the following commands:

```
mount -F NFS -o soft,rw,retry=10,timeo=30 node1:  
/ORACLE_HOME/admin/db_name//arc_dest1  
/ORACLE_HOME/admin/db_name//arc_dest1
```

NFS Implemented for High Availability

As mentioned, the optimal solution is to use NFS implemented for high availability. This solution uses the exported NFS directory stored on the shared disks of a

cluster. One node is the primary node that is used to provide access to the archived log files. If this node fails, a **failover** process changes the access path to a backup node that also has access to the shared disks. If your hardware supports NFS for high availability, consult your vendor documentation to configure this feature.

Using Shared Drives on Windows Platforms

To access remote archived log directories from your database, configure the OracleServicesid to start with a Windows NT and Windows 2000 account that has permission to write to this directory. Otherwise, attempts to write to the directory produce this message:

```
ORA-9291: sksachk: invalid device specified for archive destination.
```

Then use the following procedures to configure shared archived logs on Windows NT and Windows 2000:

1. Assign an unused drive letter to each node in your cluster. For example, if you have a cluster comprising three nodes named db1, db2, and db3, and if drive letters J, K, and L are unused, assign these letters to the nodes as shown in [Table 6-3](#):

Table 6-3 Example Letter-to-Node Assignment on Windows

Node Name	Drive Letter
db1	J:
db2	K:
db3	L:

2. Use the Windows NT or Windows 2000 Disk Administrator application to create logical partitions containing the Windows NT File System (NTFS) or Windows 2000 file system.

Each partition is a local archived log destination for the instance running on that node. To configure this, assign the drive letter owned by that node to the partition. Continuing with the example in Step 1 on db1, create a partition named "J:", on db2, create a new partition named "K:", and so on. When you

create each partition, also create a directory hierarchy called `\archivedlogs` as shown in [Table 6-4](#):

Table 6-4 Example Directory Hierarchy on Windows

Node Name	Command
db1	<code>mkdir J:\archivedlogs</code>
db2	<code>mkdir K:\archivedlogs</code>
db3	<code>mkdir L:\archivedlogs</code>

3. On each node, share the new NTFS partition with the other nodes using the following command syntax:

```
net share db_name_logs=drive_letter:\
```

Using the variables `db_name` and `drive_letter` in the example commands for each node as shown in [Table 6-5](#) where the database name is `db`:

Table 6-5 Example Netshare Command Syntax

Node Name	Command
db1	<code>net share db1_logs=J:\</code>
db2	<code>net share db2_logs=K:\</code>
db3	<code>net share db3_logs=L:\</code>

4. Use Windows NT and Windows 2000 Explorer to set permissions on these shared drives.
5. Map the shared drives from the remote nodes in the cluster using the same drive letters with the command:

```
net use \\node_name\db_name_logs drive_letter:
```

For this example, use the variables `node_name`, `db_name`, and `drive_letter` as in the following entries:

On db1, that has local drive J., enter

```
net use \\db2\DB_logs K:
net use \\db3\DB_logs L:
```

On db2, that has local drive K:, enter:

```
net use \\db1\DB_logs J:  
net use \\db3\DB_logs L:
```

On db3, that has local drive L:, enter:

```
net use \\db1\DB_logs J:  
net use \\db2\DB_logs K:
```

Backing Up Real Application Clusters Databases

This chapter discusses creating backups in **Real Application Clusters** environments. The topics in this chapter are:

- [Backups in Real Application Clusters](#)
- [Choosing a Backup Method and Backup Type](#)
- [Archiving Considerations for Real Application Clusters Environments](#)
- [Checkpoints and Log Switches](#)
- [RMAN Backups in Real Application Clusters](#)
- [Online Backups and Real Application Clusters](#)
- [Backup Examples for Real Application Clusters Using Shared Directories](#)

See Also:

- *Oracle9i User-Managed Backup and Recovery Guide* for general information about backup and recovery
- *Oracle9i Recovery Manager User's Guide* for information about using Recovery Manager
- *Oracle9i Recovery Manager Reference* for Recovery Manager reference information
- *Oracle Real Application Clusters Guard Administration and Reference Guide*

Backups in Real Application Clusters

To protect your data, you should archive the online **redo log files** and periodically back up the **data files**.

Choosing a Backup Method and Backup Type

You can use two methods for backups:

- Using Recovery Manager
- Using the operating system

Part of this chapter describes how to use Recovery Manager for backups in Real Application Clusters. For information about using operating system utilities, refer to the *Oracle9i Recovery Manager User's Guide*.

See Also: *Oracle Enterprise Manager Administrator's Guide* for information about using the **Oracle Enterprise Manager Backup Wizard**

There are two types of backups:

- **Open Database Backups**
- **Closed Database Backups**

This chapter uses the terms *open* and *closed* to indicate whether a database is available or unavailable during a backup. The term *whole backup* or *database backup* indicates that all data files and control files have been backed up. *Full* and *incremental* backups refer only to particular types of backups provided by **Recovery Manager (RMAN)**.

Open Database Backups

An open backup includes copies of one or more data files and the current control file. Open backups allow you to back up all or part of the database while it is running. Users can update data in any part of the database during an open backup. Subsequent archived redo log files or incremental backups are also necessary to allow recovery up to the time of a media failure.

Depending on your configuration, you can perform backup operations from any **node** of a Real Application Clusters database. With Real Application Clusters, you can also make open backups of multiple tablespaces simultaneously from different nodes.

Closed Database Backups

When using operating system utilities for closed backups, the backups are done while the database is closed. When you use RMAN for closed backups, an **instance** must be started and mounted but not open.

Before performing a closed backup, shut down *all* instances of your Real Application Clusters database. Then start up mount one instance. While the database is closed, you can back up its files in parallel from different nodes (to back up from multiple nodes concurrently, an instance must be started, and mounted, on each node). A closed, whole database backup includes copies of all data files and the current control file.

In ARCHIVELOG mode, both open and closed backups are permitted. In addition, in the event of media failure, you can recover all transactions that were successfully committed to the database. In NOARCHIVELOG mode, however, only whole, closed backups are permitted. Also in NOARCHIVELOG mode, in the event of media failure you can only recover transactions that were committed to the database prior to the time that it was last shut down for backup.

Note: Do *not* use operating system utilities to back up the control file in ARCHIVELOG mode. Instead, Oracle Corporation recommends that you use the ALTER DATABASE BACKUP CONTROLFILE command.

Never erase, reuse, or destroy archived redo log files until completing at least one or preferably multiple backups *after* the archivelogs were created. You can execute these backups in either open or closed mode.

See Also:

- *Oracle9i User-Managed Backup and Recovery Guide*
- *Oracle9i Database Concepts*

Online Backups and Real Application Clusters

Online backups in Real Application Clusters are efficient because they do not use the cache. Because backups primarily use I/O resources, you can use the less busy instances for other purposes. However, you should monitor disk usage to ensure that the I/O is not being saturated by the backup. If the I/O is saturated by the backup, it may adversely affect the online users.

Note: Using the `ALTER TABLESPACE ... BEGIN BACKUP` statement generates extra redo logs.

Archiving Considerations for Real Application Clusters Environments

This section explains how to archive the redo log files for each instance:

- [Changing the Archiving Mode in Real Application Clusters](#)
- [Archive File Format and Destinations in Real Application Clusters](#)
- [Backing Up the Archive Logs](#)

Changing the Archiving Mode in Real Application Clusters

To enable archive logging in Real Application Clusters environments, the database must be mounted but not open. Then start Real Application Clusters in a disabled state. To do this:

1. Shut down all instances.
2. Reset the `CLUSTER_DATABASE` parameter to `false` on one instance. If you are using the **server parameter file**, make a *sid*-specific entry for this.
3. Add settings for the `LOG_ARCHIVE_DEST_n`, `LOG_ARCHIVE_FORMAT`, and `LOG_ARCHIVE_START` parameters to the parameter file. You can multiplex the destination to up to ten locations, and the `LOG_ARCHIVE_FORMAT` parameter should contain the `%t` parameter to include the thread number in the archived log file name.
4. Start up the instance on which you have set `CLUSTER_DATABASE` to `false`.
5. Enter the following statement:

```
ALTER DATABASE ARCHIVELOG
```

6. Shut down the instance.

7. Change the value of the `CLUSTER_DATABASE` parameter to `true`.
8. Restart your instances.

To disable archive logging, follow the same steps but use the `NOARCHIVELOG` clause of the `ALTER DATABASE` statement.

Monitoring the Archiving Process in Real Application Clusters

Use the `GV$ARCHIVE_PROCESSES` and `V$ARCHIVE_PROCESSES` views to find information about the state of the ARCH processes. These views display this information for all database instances, or just the instance you are connected to, respectively.

See Also: *Oracle9i Database Reference* for more information about these views

Archive File Format and Destinations in Real Application Clusters

Archived redo logs are uniquely named as specified by the `LOG_ARCHIVE_FORMAT` parameter. This operating system-specific format can include text strings, one or more variables, and a filename extension. `LOG_ARCHIVE_FORMAT` can have variables as shown in [Table 7-1](#). Examples in [Table 7-1](#) assume that `LOG_ARCHIVE_FORMAT= arch%parameter`, and the upper bound for all parameters is 10 characters.

Table 7-1 Archived Redo Log Filename Format Parameters

Parameter	Description	Example
<code>%T</code>	Thread number, left-zero-padded	<code>arch0000000001</code>
<code>%t</code>	Thread number, not padded	<code>arch1</code>
<code>%S</code>	Log sequence number, left-zero-padded	<code>arch0000000251</code>
<code>%s</code>	Log sequence number, not padded	<code>arch251</code>

The thread parameters `%t` or `%T` are mandatory for Real Application Clusters. For example, if the instance associated with redo thread number 7 sets `LOG_ARCHIVE_FORMAT` to `LOG_%s_T%t.ARC`, then its archived redo log files are named:

```
LOG_1_T7.ARC
LOG_2_T7.ARC
LOG_3_T7.ARC
```

...

See Also:

- *Oracle9i Database Administrator's Guide* for information about specifying the archived redo log filename format and destination
- Oracle system-specific documentation for information about the default log archive format and destination

Backing Up the Archive Logs

Archive logs are generally accessible only by the node on which they were created. In Real Application Clusters you have three backup options:

- Share the location of all archive log destinations with all nodes, for example, using a [network file system \(NFS\)](#)
- Have each node back up its own archive log, for example, non-shared archived logging
- Move the archive logs to one node and back them up, for example, using `rmp`

You can use RMAN to implement the first and second solutions as described in [Chapter 6, "Configuring RMAN for Real Application Clusters"](#).

Checkpoints and Log Switches

This section discusses the use of checkpoints and log switches in Real Application Clusters environments. The topics in this section are:

- [Forcing a Checkpoint](#)
- [Forcing a Log Switch](#)
- [Forcing a Log Switch on a Closed Thread](#)

Forcing a Checkpoint

The SQL statement `ALTER SYSTEM CHECKPOINT` explicitly forces Oracle to perform a checkpoint for either the current instance or all instances. Forcing a checkpoint ensures that all changes to the database buffers are written to the data files on disk.

The `GLOBAL` clause of `ALTER SYSTEM CHECKPOINT` is the default. It forces all instances that have opened the database to perform a checkpoint. The `LOCAL` option forces a checkpoint on the current instance.

A global checkpoint is not finished until all instances complete the checkpoint. If any instance fails during the global checkpoint, however, the checkpoint might complete before that instance has been recovered.

To force a checkpoint on an instance running on a remote node, change the current instance with the `CONNECT` statement.

Note: You need the `ALTER SYSTEM` privilege to force a checkpoint.

See Also: ["Setting and Connecting to Instances"](#) on page 4-13 for information on specifying a remote node

Forcing a Log Switch

A Real Application Clusters database can force a log switch for any instance that fails to archive its online redo log files for some period of time. This can be done either because the instance has not generated many redo entries or because the instance has shut down. This prevents an instance's redo log from remaining unarchived for an extended period of time. If media recovery is necessary, the redo entries used for recovery are always recent.

For example, after an instance has shut down, another instance can force a log switch for that instance so its current redo log file can be archived. The SQL statement `ALTER SYSTEM SWITCH LOGFILE` forces the current instance to begin writing to a new redo log file, regardless of whether the current redo log file is full.

When all instances perform forced log switches, it is known as a global log switch. To force all instances to perform a global log switch, use the SQL statement `ALTER SYSTEM ARCHIVE LOG CURRENT` omitting the `THREAD` keyword. After issuing this statement, Oracle waits until all online redo log files are archived before returning control to you. Specify the `THREAD` keyword to use this statement to force a single instance to perform a log switch and archive its online redo log files.

Use the `INSTANCE FORCE LOG SWITCH` clause for each instance. You may want to force a log switch to archive, drop, or rename the current redo log file.

Note: You need the `ALTER SYSTEM` privilege to force a log switch.

Forcing a Log Switch on a Closed Thread

You can force a closed thread to complete a log switch while the database is open. This is useful if you want to drop the current log of the thread. This procedure does not work on an open thread, including the current thread, even if the instance that had the thread open is shut down. For example, if an instance aborted while the thread was open, you could not force the thread's log to switch.

To force a log switch on a closed thread, manually archive the thread using the SQL statement `ALTER SYSTEM` with the `ARCHIVE LOG` clause. For example:

```
ALTER SYSTEM ARCHIVE LOG GROUP 2;
```

To archive a closed redo log group manually that will force it to log switch, you must connect with `SYSOPER` or `SYSDBA` privileges.

See Also: The *Oracle9i Database Administrator's Guide* for information on connecting with [SYSDBA](#) or [SYSOPER](#) privileges

RMAN Backups in Real Application Clusters

This section describes the following RMAN backup issues:

- [Node Affinity Awareness](#)
- [Performing an Open Backup Using RMAN](#)

Node Affinity Awareness

On some [cluster](#) platforms, certain nodes of the cluster have faster access to some data files than to other data files. RMAN automatically detects this affinity. When deciding which channel to use to back up a particular data file, RMAN gives preference to channels allocated at nodes with affinity to that data file. To use this feature, configure RMAN channels at the various nodes of the cluster that have affinity to the data files being backed up. For example:

```
CONFIGURE CHANNEL 1 DEVICE TYPE SBT CONNECT '@INST1';  
CONFIGURE CHANNEL 2 DEVICE TYPE SBT CONNECT '@INST2';
```

See Also: *Oracle9i Recovery Manager User's Guide and Reference* for more information about the `CONNECT` clause of the `ALLOCATE` statement

Performing an Open Backup Using RMAN

If you are also backing up archive logs, then issue an `ALTER SYSTEM ARCHIVE LOG CURRENT` statement after the backup is complete. This ensures that you have all redo data to make the files in your backup consistent.

The following sample script distributes data file and archive log backups across two instances in a Real Application Clusters environment. It assumes:

- There are more than 20 files in the database
- 4 tape drives available, two on each node
- The archive log files produced by thread 2 are readable by node1

The `BACKUP . . . PLUS ARCHIVELOG` command switches out of and archives the current online log and backs up archived logs that have not been backed up. Also, At the beginning of a backup of archived logs, RMAN automatically switches out of and archives the current online redo log if needed.

The sample script is as follows:

```
RUN
{
  ALLOCATE CHANNEL NODE1_T1 TYPE SBT CONNECT 'SYS/KNL@NODE1' ;
  ALLOCATE CHANNEL NODE1_T2 TYPE SBT CONNECT 'SYS/KNL@NODE1' ;
  ALLOCATE CHANNEL NODE2_T3 TYPE SBT CONNECT 'SYS/KNL@NODE2' ;
  ALLOCATE CHANNEL NODE2_T4 TYPE SBT CONNECT 'SYS/KNL@NODE2' ;
  BACKUP
  FILESPERSET 6
  FORMAT 'DF_%T_%S_%P'
  DATABASE PLUS ARCHIVELOG;
```

See Also: *Oracle9i Recovery Manager User's Guide and Reference* for complete information on open backups using RMAN

Backup Examples for Real Application Clusters Using Shared Directories

This section contains backup examples using shared directories, some of which use RMAN, for Real Application Clusters. The topics in this section are:

- [Using RMAN to Backup to Multiple Nodes with Several Channels](#)
- [Avoiding the Thread Statement for Backing Up Archivelogs](#)
- [Making Backups to Shared Archive Log Destinations](#)

Using RMAN to Backup to Multiple Nodes with Several Channels

Although you can perform a concurrent backup of more than one node of a cluster using the `CONNECT` clause in the `ALLOCATE CHANNEL` command, you cannot recover from such a backup unless you have installed Oracle version 8.1.6 or greater.

RMAN stores the local path of the backup file in its recovery catalog. If you allocate multiple channels on recovery, RMAN expects to find the backup file in any of these channels. If RMAN cannot locate the required file in the channel, recovery fails.

Use the `SET AUTOLOCATE ON` command to make RMAN search all channels for each backup that is needed for a restore operation.

Avoiding the Thread Statement for Backing Up Archivelogs

Do not use the `THREAD` statement of the `BACKUP ARCHIVED LOG` clause when running RMAN first on one instance and then on the second one. The command syntax for this is:

```
RMAN TARGET SYS/SYSPASS@DB1 CATALOG RMANUSER/RMANPASS@RMANCAT
RUN
{
...
  BACKUP ARCHIVELOG THREAD 1;
...
}
RMAN TARGET connect /as sys@db2 CATALOG RMANUSER/RMANPASS@RMANCAT
{
...
  BACKUP ARCHIVELOG THREAD 2;
...
}
```

This procedure might not work because it is possible that archive logs for other threads are archived on different nodes. If node 1, for example, goes down for a longer period of time, the surviving node performs log switches for the inactive instance to maintain the system change number (SCN). The archiver of node 2 also archives the redo logs from DB1 into its log archive destination. This procedure is called a **redo log switch**.

Making Backups to Shared Archive Log Destinations

To create backups by sharing all archive logs with all nodes of a cluster, use NetBios, NFS or HA-NFS. Doing this enables you to execute the backups from any node because every node can read all the logs. In the following example, node 1 backs up all redo logs of all nodes. Make sure that the directories are configured for sharing as described in the previous sections.

The `ALTER SYSTEM ARCHIVE LOG CURRENT` statement forces all nodes to back up their current log files.

```

RUN
{
  ALLOCATE CHANNEL node_1 DEVICE TYPE sbt CONNECT 'sys/sys_pwd@node_1';
  ALLOCATE CHANNEL node_2 DEVICE TYPE sbt CONNECT 'sys/sys_pwd@node_2';
  BACKUP FILESPERSET 1
    (TABLESPACE SYSTEM, rbs, data1, data2
     CHANNEL node_1)
    (TABLESPACE temp, reccat, data3, data4
     CHANNEL node_2);
  BACKUP FILESPERSET 20 ARCHIVELOG ALL DELETE ALL INPUT;
}

```

Backing Up Local Files from Each Node Using Non-Shared Archive Log Destinations

This section contains examples of making backups using non-shared archive log destinations. The topics in this chapter are:

- [Backing Up All Files from One Node](#)
- [Recovering Archive Logs from One Node](#)
- [Restoring and Recovering Archive Logs from All Local Nodes Using Oracle Release 1 \(9.0.1\)](#)

- [Restoring and Recovering Archive Logs from All Local Nodes Using Oracle 8.1.5 Or Earlier](#)

If you do not share all archive logs, you can back up the logs locally on every node. For recovery, however, you need to have access from the node on which you begin recovery to all the archive logs created by the database. For this reason Oracle Corporation recommends using a media management system that supports archiving over the network or shared directory services to simplify restoring log files.

To do this, adapt the expression in the `LIKE` clause to your own configuration, for example, `J : %` if you use Windows NT and Windows 2000.

Backing Up All Files from One Node

You can back up all the archive logs from one node into one backup instead of archiving them from each node separately. This makes it easier to find all backups when you recover. If you do not use shared directories to back up and restore archive logs, copy or move them using operating system tools, while retaining their exact names. You can easily create scripts to do this before backing up or restoring the logs. If you are using Oracle Enterprise Manager, you can schedule the execution of this script before the backup.

To copy all archive logs to the local directories on node 1, use a script similar to the following:

```
#!/bin/sh
sqlplus system/manager@node1 @switchlog.sql
rcp 'node2:/u01/app/oracle/product/900/admin/db/arch2/*' /u01/app/oracle/
product/900/admin/db/arch2
```

Ensure that you have adequate space on your destination device before executing this script.

Recovering Archive Logs from One Node

If you moved all archive logs to one node to back them up, recovery is as easy as recovery using shared directories. You do not need to use `autolocate` to restore archive log backups if they were all backed up from one node, and if you are restoring from that same node.

Restoring and Recovering Archive Logs from All Local Nodes Using Oracle Release 1 (9.0.1)

If you backed the archive logs from each node using a media management system, you can use several channels for recovery. RMAN examines every channel for the required file if RMAN does not find it in the first one. This is the autolocate feature that recovers a database using the local tape drive on the remote nodes as shown in the following example:

```
RMAN TARGET sys/syspass CATALOG rmanuser/rmanpass@rmancat
RUN
{
  ALLOCATE CHANNEL t1 DEVICE TYPE sbt PARS 'ENV=(NSR_CLIENT=node1)';
  ALLOCATE CHANNEL t2 DEVICE TYPE sbt PARS 'ENV=(NSR_CLIENT=node2)';
  SET AUTOLOCATE ON;
  RECOVER TABLESPACE users;
  SQL 'alter tablespace users online';
}
```

Restoring and Recovering Archive Logs from All Local Nodes Using Oracle 8.1.5 Or Earlier

If you backed up the logs from each node, restore all the log files from remote nodes before performing recovery. Perform recovery in three steps:

1. Restore the data files.
2. Restore the archive logs from remote nodes.
3. Recover the database.

Restore the files locally and move them to the recovering node. Because you have to restore the archive logs before RMAN can begin recovery, specify which log files to restore as in the following example:

```
RMAN TARGET sys/syspass@node1 catalog rmanuser/rmanpass@rmancat
RUN
{
  ALLOCATE CHANNEL t1 TYPE 'sbt_tape';
  RESTORE TABLESPACE users;
  RELEASE CHANNEL t1;
}
RUN
{
  ALLOCATE CHANNEL t1 TYPE 'sbt_tape' PARS 'ENV=(NSR_CLIENT=node2)';
  # this next line is optional, they will be restored to the first log_archive
```

```
dest at the restoring node
  SET ARCHIVELOG DESTINATION TO '/u01/app/oracle/product/900/admin/db/arch2';
  RESTORE ARCHIVELOG
  # this next line is optional, if you don't want to restore ALL archive logs:
from time "to_date('05.09.1999 00:00:00','DD.MM.YYYY HH24:Mi:SS')"
  # the like operand should be used as each archive dest may contain copies
from each thread
  # (this is due to redo log switches)
  LIKE '%/2_%';
  RELEASE CHANNEL t1;
}
RUN
{
  ALLOCATE CHANNEL d1 TYPE DISK;
  RECOVER TABLESPACE users;
SQL 'alter tablespace users online';
}
```

Database Recovery in Real Application Clusters

This chapter describes how to perform recovery in **Real Application Clusters** environments. This chapter contains the following topics:

- [Recovery from Instance Failures](#)
- [Recovery from Media Failures](#)
- [Parallel Recovery](#)
- [RMAN Recovery Examples for Real Application Clusters](#)

See Also:

- *Oracle9i User-Managed Backup and Recovery Guide* for general information about Oracle recovery
- *Oracle9i Recovery Manager User's Guide*
- *Oracle9i Recovery Manager Reference* for more information about Recovery Manager

Recovery from Instance Failures

An **instance** failure occurs when software or hardware problems disable an instance. After instance failure, Oracle uses the online **redo log files** to perform database recovery. When instances accessing a database in shared mode fail, Oracle automatically performs online instance recovery. Instances that continue running on other **nodes** are not affected if they are reading from the buffer cache. If instances attempt to write, Oracle stops their transactions; Oracle suspends all operations to the database until cache recovery of the failed instance completes.

The following sections describe the recovery process as it is performed after an instance's failure to access a database in shared mode.

- [Single Node Failure](#)
- [Multiple-Node Failures](#)
- [Fast-Start Checkpointing and Fast-Start On-Demand Rollback](#)
- [Access to Data Files for Instance Recovery](#)
- [Steps of Oracle Instance Recovery](#)

See Also: The *Oracle9i User-Managed Backup and Recovery Guide*

Single Node Failure

Instances in Real Application Clusters perform recovery through the SMON processes of the other running instances. If one instance fails, another instance's SMON process performs instance recovery for the failed instance.

Instance recovery does not include restarting the failed instance or the recovery of applications that were running on that instance. Applications that were running can continue by using **failover** as described in *Oracle9i Real Application Clusters Installation and Configuration*.

When one instance performs recovery for another instance, the surviving instance reads redo log entries generated by the failed instance and uses that information to ensure that committed transactions are reflected in the database. Data from committed transactions is not lost. The instance performing recovery rolls back transactions that were active at the time of the failure and releases resources used by those transactions.

Note: All online redo logs must be accessible for recovery. Therefore, Oracle Corporation recommends that you mirror your online logs.

See Also: *Oracle9i Real Application Clusters Concepts* for conceptual information about application failover and [high availability](#)

Multiple-Node Failures

After multiple node failures, as long as one instance continues running, its SMON process performs instance recovery for any other instances that fail. If all instances of a Real Application Clusters database fail, instance recovery is performed automatically the next time an instance opens the database.

The instance performing recovery does not have to be one of the instances that failed. In addition, the instance performing recovery can mount the database in either shared or exclusive mode from any node of a Real Application Clusters database. This recovery procedure is the same for Oracle running in shared mode as it is for Oracle in exclusive mode, except that one instance performs instance recovery for all failed instances.

Fast-Start Checkpointing and Fast-Start On-Demand Rollback

Fast-start checkpointing is the basis for Fast-start fault recovery in Oracle. Fast-start checkpointing occurs continuously, advancing the checkpoint while Oracle writes blocks to disk. You can limit the duration of the roll forward phase of Fast-start checkpointing. Oracle automatically adjusts the rate of checkpoint writes to meet the specified roll-forward limit while issuing a minimum number of writes.

Using the fast-start on-demand rollback feature, Oracle automatically allows new transactions to begin immediately after the roll forward phase of recovery completes.

Note: Oracle invokes this feature automatically. You do not need to set parameters or issue statements to use Fast-start checkpointing and Fast-start on-demand rollback.

See Also: *The Oracle9i Database Performance Guide and Reference* for details on these two features

Access to Data Files for Instance Recovery

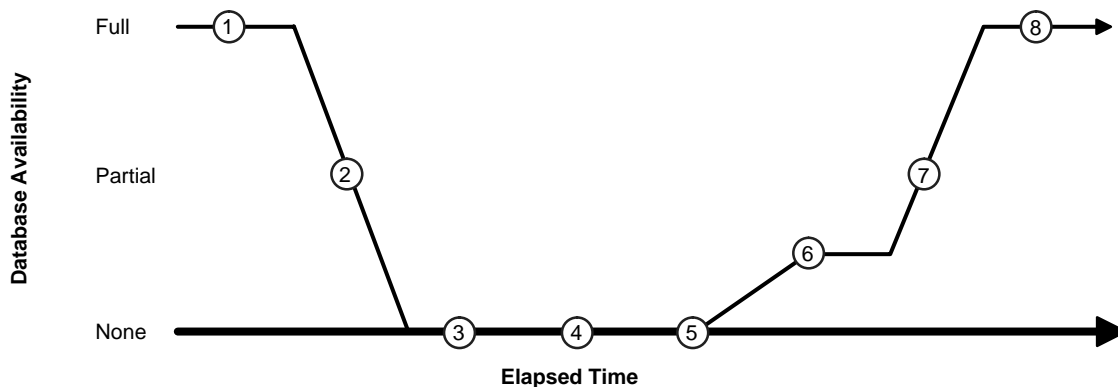
An instance performing recovery for another instance must access all online data files that the failed instance accessed. When instance recovery fails because a **data file** fails verification, the instance that attempted to perform recovery does not fail but Oracle writes a message to the alert log file. In this case, the recovering instance needs access to the online archive log files.

After you correct the problem that prevented access to the database files, use the SQL statement `ALTER SYSTEM CHECK DATAFILES` to make the files available to the instance.

Steps of Oracle Instance Recovery

Figure 8-1 and the narrated steps following illustrate the degree of database availability during each step of Oracle instance recovery.

Figure 8-1 Steps of Oracle Instance Recovery



The steps in recovery are:

1. Real Application Clusters is running on multiple nodes.
2. Node failure is detected.
3. The **Global Cache Service (GCS)** is reconfigured; resource management is redistributed onto the surviving nodes.
4. SMON reads the redo log of the failed instance to identify the database blocks that it needs to recover.

5. SMON issues requests to obtain all of the database blocks it needs for recovery. After the requests are complete, all other blocks are accessible.

Note: The **Global Cache Service Processes (LMSn)** only re-master resources that lose their masters.

6. Oracle performs roll forward: redo logs of the dead threads are applied to the database.
7. Oracle performs roll back: **rollback segments** are applied to the database for all uncommitted transactions.
8. Instance recovery is complete and all data is accessible.

During step 6, forward application of the redo logs, database access is limited by the transitional state of the buffer cache. The following data access restrictions exist for all user data in all data files:

- No writes to surviving buffer caches can succeed while access is limited
- No disk I/O of any type by way of the buffer cache and direct path can be done from any of the surviving instances
- No resource requests are made to the Global Cache Service for user data

Oracle can read buffers already in the cache with the correct global resource because this does not involve any I/O or resource operations.

The transitional state of the buffer cache begins at the conclusion of the initial resource scan step when instance recovery is first started by scanning for dead redo threads. Subsequent resource scans are made if new *dead* threads are discovered. This state lasts while the redo log is applied (cache recovery) and ends when the redo logs have been applied and the file headers have been updated. Cache recovery operations conclude with validation of the invalid resources, which occurs after the buffer cache state is normalized.

Recovery from Media Failures

Media failures occur when Oracle file storage mediums are damaged. This usually prevents Oracle from reading or writing data resulting in the loss of one or more database files. In these situations, you should restore backups of the data files to recover the database.

If you are using **Recovery Manager (RMAN)**, you may also need to apply incremental backups and archived redo log files, and use a backup of the control file. If you are using operating system utilities, you might need to apply archived redo log files to the database and use a backup of the control file.

The media recovery topics discussed in this section are:

- [Complete Media Recovery](#)
- [Incomplete Media Recovery](#)
- [Restoring and Recovering Redo Log Files](#)
- [Disaster Recovery](#)

See Also: *Oracle9i User-Managed Backup and Recovery Guide* for procedures to recover from various types of media failure

Complete Media Recovery

You can perform complete media recovery in either exclusive or shared mode. You can also recover multiple data files or tablespaces on multiple instances simultaneously. [Table 8-1](#) shows the status of the database that is required to recover particular database objects.

Table 8-1 Database Status for Media Recovery

To Recover	Database Status
An entire database or the <code>SYSTEM</code> tablespace.	The database must be mounted but not opened by any instance.
A tablespace other than the <code>SYSTEM</code> tablespace.	The database must be opened by the instance performing the recovery and the tablespace must be offline.
A data file.	The database can be open with the data file offline, or the database can be mounted but not opened by any instance. (For a data file in the <code>SYSTEM</code> tablespace, the database must be mounted but not open.)
A data block.	The database can be open with the data file online or offline or mounted (but not opened). The block must be marked as <i>media corrupt</i> on the disk.

Block Media Recovery

Block media recovery is an RMAN feature in which Oracle applies archives from multiple instances to complete the recovery process. For online backups, however, you also have the option of using block media recovery.

Block media recovery reduces your system's **mean time to recover (MTTR)** and increases system availability. Block media recovery is intended primarily for data-specific data blocks cited in Oracle error messages. You cannot use Oracle block media recovery to recover complete files.

See Also: The *Oracle9i User-Managed Backup and Recovery Guide* for more information about block media recovery

Incomplete Media Recovery

You can perform incomplete media recovery of the entire database while the database is mounted in shared or exclusive mode if the database is not opened by an instance. Do this using the following database recovery options:

With RMAN use one of the following clauses with the `SET` statement before restoring and recovering:

- `UNTIL CHANGE integer`
- `UNTIL TIME date`
- `UNTIL LOGSEQ integer THREAD integer`

With operating system utilities, restore your backups and then use one of the following clauses with the `RECOVER DATABASE` statement:

- `UNTIL CANCEL`
- `UNTIL CHANGE integer`
- `UNTIL TIME date`

Incomplete media recovery of a tablespace is called Tablespace Point in Time Recovery (SPITR).

Note: You cannot use block media recovery for incomplete media recovery.

See Also: *Oracle9i User-Managed Backup and Recovery Guide* for more information on the use of the `SET` and `RECOVER DATABASE` statements

Restoring and Recovering Redo Log Files

Media recovery of a database accessed by Real Application Clusters may require the simultaneous opening of multiple archived log files. Because each instance writes redo log data to a separate redo thread, recovery may require at least one archived log file for each thread. However, if a thread's online redo log contains enough recovery information, restoring archived log files for that thread is unnecessary.

Recovery Using RMAN

RMAN automatically restores and applies the required archive logs when processing the RMAN `RECOVER` command. By default, RMAN restores archive logs to the first local archive log destination of the instance(s) that restore the archived logs. If you are using multiple nodes to restore and recover, this means that the archive logs may be restored to any node performing the restore/recover operation.

The node that reads the restored logs and performs the roll forward recovery is the target node to which the connection was initially made. You must ensure that the logs are readable from that node using the following platform-specific methods.

Recovery Using Operating System Utilities

During recovery, Oracle displays messages that supply information about the files it requires and then prompts you for their complete filenames. For example, if the log history is enabled and the filename format is `LOG_T%t_SEQ%s`, where `%t` is the thread and `%s` is the log sequence number, you might receive messages similar to these to begin recovery with SCN 9523 in thread 8:

```
ORA-00279: Change 9523 generated at 27/09/91 11:42:54 needed for thread 8
ORA-00289: Suggestion : LOG_T8_SEQ438
ORA-00280: Change 9523 for thread 8 is in sequence 438
Specify log: {RET = suggested | filename | AUTO | FROM | CANCEL}
```

If you use the `ALTER DATABASE` statement with the `RECOVER` clause, you receive these messages but not the prompt. Redo log files may be required for each enabled thread in Real Application Clusters. Oracle issues a message when a log file is no longer needed. The next log file for that thread is then requested, unless the thread was disabled or recovery is complete.

If recovery reaches a time when an additional thread was enabled, Oracle requests the archived log file for that thread. Whenever an instance enables a thread, it writes a redo entry that records the change; therefore, all necessary information about threads is available from the control file during recovery.

If you are using a backup control file, when all archive log files are exhausted you may need to redirect the recovery process to the online redo log files to complete recovery. If recovery reaches a time when a thread was disabled, Oracle informs you that the log file for that thread is no longer needed and that Oracle does not need further log files for the thread.

Note: If Oracle reconstructs the names of archived redo log files, the format that `LOG_ARCHIVE_FORMAT` specifies for the instance doing recovery must be the same as the format specified for the instances that archived the files. The instance performing recovery should also use that value. You can specify a different value of `LOG_ARCHIVE_DEST` during recovery if the archived redo log files are not at their original archive destinations.

Disaster Recovery

This section describes disaster recovery using RMAN and operating system utilities. Implement disaster recovery, for example, when a failure makes an entire site unavailable. In this case, you can recover at an alternate site using open or closed database backups.

Note: To recover up to the latest point in time, all logs must be available at a remote site; otherwise some committed transactions may be lost.

Disaster Recovery Using RMAN

This section describes a recovery scenario using RMAN. The scenario assumes:

- You have lost the entire database, the control file, and the online redo log
- You will be distributing your recovery process over two nodes
- There are four tape drives (two on each node)
- You are using a recovery catalog
- You performed distributed backups

Note: It is highly advisable to back up the database immediately after opening the database reset logs, because all previous backups are invalidated. This step is not shown in this example.

Preparing the Recovery Before beginning the database recovery, you must:

- Restore your initialization parameter file from your most recent backup; note that you must have manually made copies of your **initialization parameter file** if you are using the **server parameter file**.
- Catalog archive logs, data file copies, or backup sets that are on disk but are not registered in the recovery catalog.

If you are using a recovery catalog, the archive logs up to the logseq number being restored must be cataloged in the recovery catalog or RMAN will not know where to find them.

If you resynchronize the recovery catalog frequently there should not be many archive logs that need cataloging.

What the Sample Script Does The following script restores and recovers the database to the most recently available archived log, which is log 124 thread 1. It does the following:

- Starts the database `NOMOUNT` and restricts connections to DBA-only users.
- Restores the control file to the locations specified by the `CONTROL_FILES` initialization parameter.
- Mounts the database.
- Catalogs any archive logs not in the recovery catalog.
- Restores the database files (to their original locations).

If you cannot restore a data file to its original location due to a failed volume, use the `SET NEWNAME` command before the restore.

- Recovers the data files either by using a combination of incremental backups and redo, or just redo.

RMAN completes the recovery when it reaches the log sequence number specified.

- Opens the database resetlogs.

Note: Only complete the following step if you are certain there are no other archived logs to apply.

- Oracle Corporation recommends that you back up your database after the resetlogs. This is not shown in the example.

Restore/Recover Sample Script: Start SQL*Plus and connect to the database. Then enter the following STARTUP command:

```
STARTUP NOMOUNT RESTRICT
```

Start RMAN and run the following script.

Note: The user specified in the target parameter must have **SYSDBA** privilege.

```
RMAN TARGET user/password@node1 RCVCAT RMAN/RMAN@RCAT
RUN
{
  SET AUTOLOCATE ON;
  SET UNTIL LOGSEQ 124 THREAD 1;
  ALLOCATE CHANNEL t1 TYPE 'SBT_TAPE' CONNECT 'user/password@node1';
  ALLOCATE CHANNEL t2 TYPE 'SBT_TAPE' CONNECT 'user/knl@node1';
  ALLOCATE CHANNEL t3 TYPE 'SBT_TAPE' CONNECT 'user/knl@node2';
  ALLOCATE CHANNEL t4 TYPE 'SBT_TAPE' CONNECT 'user/knl@node2';
  ALLOCATE CHANNEL d1 TYPE DISK;
  RESTORE CONTROLFILE;
  ALTER DATABASE MOUNT;
  CATALOG ARCHIVELOG '/oracle/db_files/node1/arch/arch_1_123.rdo';
  CATALOG ARCHIVELOG '/oracle/db_files/node1/arch/arch_1_124.rdo';
  RESTORE DATABASE;
  RECOVER DATABASE;
  ALTER DATABASE OPEN RESETLOGS;
}
```

Parallel Recovery in Real Application Clusters

Parallel recovery uses multiple CPUs and I/O parallelism to reduce the time required to perform thread or media recovery. Parallel recovery is most effective at reducing recovery time while concurrently recovering several data files on several

disks. You can use parallel instance and crash recovery as well as parallel media recovery in Real Application Clusters.

See Also: *Oracle9i User-Managed Backup and Recovery Guide* for more information on these topics

Parallel Recovery in Real Application Clusters

You can use two types of utilities for parallel recovery in Real Application Clusters:

- [Parallel Recovery Using RMAN](#)
- [Parallel Recovery Using Operating System Utilities](#)

Parallel Recovery Using RMAN

With RMAN's `RESTORE` and `RECOVER` statements, Oracle automatically parallelizes all three stages of recovery. These stages are:

Restoring Data Files When restoring data files, the number of channels you allocate in the RMAN recover script effectively sets the parallelism RMAN uses. For example, if you allocate 5 channels, you can have up to 5 parallel streams restoring data files.

Applying Incremental Backups Similarly, when you are applying incremental backups, the number of channels you allocate determines the potential parallelism.

Applying Redo Logs Oracle applies redo logs using a specific number of parallel processes as determined by the setting for the `RECOVERY_PARALLELISM` parameter. The `RECOVERY_PARALLELISM` initialization parameter specifies the number of redo application server processes participating in instance or media recovery.

During parallel recovery, one process reads the log files sequentially and dispatches redo information to several recovery processes that apply the changes from the log files to the data files. A value of 0 or 1 for `RECOVERY_PARALLELISM` indicates that Oracle performs recovery serially. The value of this parameter cannot exceed the value of the `PARALLEL_MAX_SERVERS` parameter. RMAN parallelizes the restore and application of incremental backups using channel allocation.

See Also: *The Oracle9i Recovery Manager User's Guide and Reference* for more information about using RMAN

Parallel Recovery Using Operating System Utilities

You can parallelize instance and media recovery using either of the following methods:

- [Setting the RECOVERY_PARALLELISM Parameter](#)
- [Specifying RECOVER Statement Options](#)

Real Application Clusters can use one process to read the log files sequentially and dispatch redo information to several recovery processes to apply the changes from the log files to the data files. Oracle automatically starts the recovery processes, so you do not need to use more than one session to perform recovery.

Setting the RECOVERY_PARALLELISM Parameter

The `RECOVERY_PARALLELISM` initialization parameter specifies the number of redo application server processes participating in instance or media recovery. One process reads the log files sequentially and dispatches redo information to several recovery processes. The recovery processes then apply the changes from the log files to the data files. A value of 0 or 1 indicates that Oracle reforms recovery serially by one process. The value of this parameter cannot exceed the value of the `PARALLEL_MAX_SERVERS` parameter.

Specifying RECOVER Statement Options

When you use the `RECOVER` statement to parallelize instance and media recovery, the allocation of recovery processes to instances is operating system specific. The `DEGREE` keyword of the `PARALLEL` clause can either signify the number of processes on each instance of a Real Application Clusters database or the number of processes to spread across all instances.

See Also:

- *Oracle9i Database Performance Guide and Reference* for more information on Fast-start parallel rollback
- Oracle system-specific documentation for more information on the allocation of recovery processes to instances

Fast-Start Parallel Rollback in Real Application Clusters

In Real Application Clusters, multiple parallel recovery processes are owned by and operated only within the instance that generated them. To determine an accurate setting for `FAST_START_PARALLEL_ROLLBACK`, examine the contents of `V$FAST_START_SERVERS` and `V$FAST_START_TRANSACTIONS`.

Fast-start parallel rollback does not perform cross-instance rollback. However, it can improve the processing of rollback segments for a single database with multiple instances since each instance can spawn its own group of recovery processes.

RMAN Recovery Examples for Real Application Clusters

The following are RMAN recovery examples for use in Real Application Clusters.

Recovery to Shared Archive Log Destinations

If you share archive log directories, you can change the destination of the automatic restoration of archive logs with the `SET` clause to restore the files to a local directory of the node from where you start recovery. This step does not apply if you use HA-NFS because you have only one shared directory which is mounted from all nodes.

To restore the `USERS` tablespace from node 1, use an RMAN command similar to the following:

```

RMAN TARGET connect /as sys@node1 CATALOG rmanuser/rmanpass@rmancat
RUN
{
    ALLOCATE CHANNEL t1 type 'sbt_tape';
    # set archive log destination to '/u01/app/oracle/product/900/admin/db/arch1';
    RECOVER TABLESPACE users;
    SQL 'alter tablespace users online';
    RELEASE CHANNEL t1;
}

```

Restoring Archive Logs with RMAN

If RMAN has concurrent access to all backups, it automatically restores all necessary archive logs from previous backups for recovery. In Real Application Clusters environments, the restore procedure varies depending on the option you used to back up the archive logs.

If you share archive log directories, you can change the destination of the automatic restoration of archive logs with the `SET` clause to restore the files to a local directory of the node from where you begin recovery.

To restore the `USERS` tablespace from node 1, use an RMAN script similar to the following:

```
RMAN TARGET user/password@node1 catalog rman/rman@rman

RUN
{
  ALLOCATE CHANNEL t1 type 'sbt_tape';
  SET ARCHIVELOG DESTINATION TO '/u01/app/oracle/product/900/admin/ops/arch1';
  RESTORE TABLESPACE users;
  RECOVER TABLESPACE users;
  SQL 'alter tablespace users online';
  RELEASE CHANNEL t1;
}
```

If you backed up each node's log files using a central media management system, you can use the `RMAN SET AUTOLOCATE` command. If you use several channels for recovery, RMAN asks every channel for the required file if it does not find it in the first one. This feature enables you to recover a database using the local tape drive on the remote node:

```
RMAN TARGET user/password catalog rman/rman@rman

RUN
{
  ALLOCATE CHANNEL t1 type 'sbt_tape' parms 'ENV=(NSR_CLIENT=node1)';
  ALLOCATE CHANNEL t2 type 'sbt_tape' parms 'ENV=(NSR_CLIENT=node2)';
  ALLOCATE CHANNEL t3 type 'sbt_tape' parms 'ENV=(NSR_CLIENT=node3)';
  SET AUTOLOCATE ON;
  RECOVER TABLESPACE users;
  SQL 'ALTER TABLESPACE users ONLINE';
  RELEASE CHANNEL t1;
}
```

If you backed up the logs from each node without using a central media management system, you must first restore all the log files from the remote nodes and move them to the host from which you will start recovery. This means you must perform recovery in three steps:

1. Restore the data files.
2. Restore the archive logs.
3. Begin recovery.

```
RMAN TARGET user/password CATALOG rman/rman@rman

RUN
```

```

{
  ALLOCATE CHANNEL t1 TYPE 'sbt_tape' CONNECT SYS/sys@node1;
  RESTORE TABLESPACE users;
  RELEASE CHANNEL t1;
}

RUN
{
  ALLOCATE CHANNEL t1 TYPE 'sbt_tape' CONNECT SYS/sys@node2;
  RESTORE ARCHIVELOG
  # this line is optional if you don't want to restore ALL archive logs:
  FROM TIME "to_date('05.09.1999 00:00:00','DD.MM.YYYY HH24:Mi:SS')"
  LIKE '%/2_%';
  RELEASE CHANNEL t1;
}
RUN
{
  ALLOCATE CHANNEL t1 TYPE 'sbt_tape' CONNECT SYS/sys@node3;
  RESTORE ARCHIVELOG
  # this line is optional if you don't want to restore ALL archive logs:
  FROM TIME "to_date('05.09.1999 00:00:00','DD.MM.YYYY HH24:Mi:SS')"
  like '%/3_%';
  RELEASE CHANNEL t1;
}
EXIT

rcp node2:/u01/app/oracle/product/900/admin/ops/arch2
/u01/app/oracle/product/900/admin/ops/arch2
rcp node3:/u01/app/oracle/product/900/admin/ops/arch2
/u01/app/oracle/product/900/admin/ops/arch2

RMAN TARGET user/password catalog rman/rman@rman

RUN
{
  ALLOCATE CHANNEL t1 TYPE 'sbt_tape';
  ALLOCATE CHANNEL d1 type disk;
  RECOVER TABLESPACE users;
  SQL 'ALTER TABLESPACE USERS ONLINE';
}

```

Note: In the recover step, there is an 'sbt_tape' channel allocated so that the archive logs generated on node 1 will be automatically restored.

If you moved all archive logs to one node to back them up, recovery is as easy as recovery using shared directories. To make sure you have all the log files, copy all remote log files with your shell script as in this example:

```
/rcp_all_logs.sh
RMAN TARGET internal/sys@node1 catalog rman/rman@rman
RUN
{
  ALLOCATE CHANNEL t1 type 'sbt_tape' format 'al_t%t_s%s_p%p';
  BACKUP ARCHIVELOG ALL DELETE INPUT;
  RELEASE CHANNEL t1;
}
```


Part IV

Scaling Your Real Application Clusters Environment

Part Four explains how to add nodes and instances and how to delete instances. The chapter in Part Four is :

- [Chapter 9, "Adding Nodes and Instances and Deleting Instances in Real Application Clusters"](#)

Adding Nodes and Instances and Deleting Instances in Real Application Clusters

This chapter provides detailed procedures on how to add **nodes** and **instances** and delete instances in **Real Application Clusters**. The topics in this chapter are:

- [Adding Nodes to a Cluster](#)
- [Overview of Procedures for Adding Nodes](#)
- [Adding a Node at the Clusterware Layer](#)
- [Adding a Node at the Oracle Layer](#)
- [Deleting Instances](#)

See Also: *Oracle9i Real Application Clusters Installation and Configuration* for procedures on using the **Oracle Database Configuration Assistant (DBCA)** to create and delete databases

Adding Nodes to a Cluster

This section explains how to dynamically add nodes to Real Application Clusters environments. The topics in this section are:

- [Overview of Procedures for Adding Nodes](#)
- [Adding a Node at the Clusterware Layer](#)
- [Adding a Node at the Oracle Layer](#)

See Also: "[Deleting Instances](#)" on page 9-19 for procedures on deleting an instance

Overview of Procedures for Adding Nodes

The procedure for adding nodes to Real Application Clusters comprises two main steps:

- [Adding a Node at the Clusterware Layer](#)
- [Adding a Node at the Oracle Layer](#)

The method for adding nodes at the clusterware layer is operating-system dependent as described in this chapter. For the second step, adding a node at the Oracle layer, the procedures in this chapter require using the **Oracle Universal Installer (OUI)** and the **Oracle Database Configuration Assistant (DBCA)**. The procedures for the second step are generic and therefore apply to all platforms.

Adding a Node at the Clusterware Layer

On UNIX, unless you preconfigure your clusterware to accommodate additional nodes, you cannot dynamically add nodes. Therefore, you must stop the clusterware and reconfigure your **cluster** to accommodate additional nodes. Then follow the procedures under the heading "[Adding a Node at the Clusterware Layer on UNIX](#)".

On Windows NT and Windows 2000, Oracle supplies the cluster software or **operating system-dependent (OSD) layer**. This enables you to dynamically add nodes as described under the heading "[Adding a Node at the Clusterware Layer on Windows NT and Windows 2000](#)" on page 9-4.

The topics in this section are:

- [Adding a Node at the Clusterware Layer on UNIX](#)
- [Adding a Node at the Clusterware Layer on Windows NT and Windows 2000](#)

Adding a Node at the Clusterware Layer on UNIX

You can take advantage of the **scalability** and **high availability** features of Real Application Clusters by adding new nodes to existing UNIX clusters that uses a Real Application Clusters database. As mentioned, you can add nodes without stopping your database if you have preconfigured your UNIX cluster to accommodate additional nodes.

Oracle Corporation recommends using the following procedures to add new nodes to existing Real Application Clusters in UNIX environments. These procedures assume that there is already a UNIX cluster running your Real Application Clusters database.

To prepare the new node, ensure that it is an exact clone of the existing nodes in terms of vendor operating system, clusterware software, and so on. Then perform the following tasks in the order shown to add a node at the clusterware layer on UNIX:

1. [Connect Node to Cluster](#)
2. [Install Vendor Cluster Software](#)
3. [Create Raw Devices](#)

Connect Node to Cluster

Connect the new node's hardware to the existing cluster. This includes electrical connections, network **interconnects**, shared disk subsystem connections, and so on. Refer to your hardware vendor documentation for details on this step.

Install Vendor Cluster Software

Install the cluster software by referring to your vendor's installation procedures. Also use your vendor documentation for procedures to make the new node an active member of your cluster.

Create Raw Devices

If your platform supports a cluster file system and you not are using raw devices for your cluster database, then you do not need to create raw devices as described in this section. Proceed to "[Adding a Node at the Oracle Layer](#)" on page 9-9.

As part of the preinstallation steps before installing the Real Application Clusters software on the new node, create at least two new disk volumes to accommodate the redo logs for the new instance. Refer to *Oracle9i Real Application Clusters Installation and Configuration* for the requirements for these redo logs and the preinstallation steps.

Make the disk volumes for the redo logs the same size as the redo log volumes that you configured for your existing nodes' instances. Refer to your vendor documentation for procedures on creating disk volumes and raw devices.

If the cluster database to which you want to add a new node and instance uses **automatic undo management**, then create an additional raw volume for the new undo tablespace for the new instance. After completing this step, proceed to "[Adding a Node at the Oracle Layer](#)" on page 9-9 to add the node at the Oracle layer.

See Also: *Oracle9i Real Application Clusters Installation and Configuration* for more information about creating raw devices

Adding a Node at the Clusterware Layer on Windows NT and Windows 2000

You can take advantage of the scalability and high availability features of Real Application Clusters by dynamically adding new nodes to an existing Windows cluster that uses a Real Application Clusters database.

Oracle Corporation recommends using the following procedures to add a node in Windows environments. These procedures assume that there is already a Windows

cluster running your Real Application Clusters database. These procedures also assume that the new node uses the same operating system that is running on the existing nodes of your cluster. In other words, you have installed Microsoft Windows NT 4.0 or Windows 2000 on all the nodes in your cluster.

Perform the following tasks in the order shown to add a new node at the clusterware layer on Windows NT and Windows 2000:

1. [Connect Node to Cluster](#)
2. [Install Cluster Software](#)
3. [Create Disk Partitions](#)
4. [Create Links to Disk Partitions](#)

Connect Node to Cluster

Connect the new node's hardware to the existing cluster. This includes electrical connections, network interconnects, shared disk subsystem connections, and so on. Refer to your hardware vendor documentation for details on this step.

Install Cluster Software

If you are using your operating system vendor supplied clusterware on the existing nodes, also install this vendor software on the new node using your vendor's installation procedures.

However, if you are using Oracle Corporation's reference clusterware/operating system-dependent software on the existing nodes of your cluster, you must also install these Oracle OSD clusterware on the new node using the Cluster Setup Wizard. To do this, perform the following steps from one of the *existing* nodes before running the wizard:

1. If you use the **Virtual Interface Architecture (VIA)** interconnect on the existing cluster, make sure that VIA interconnect is also installed on the new node.
2. If you have a private interconnect network on the existing cluster, make sure that the private interconnect network is also installed on the new node.
3. Determine the disk drive on which the cluster software was installed on the existing nodes. Make sure that you have at least 2MB of free space on the same drive on the new node to install the OSD clusterware and the Object Link Manager.

4. Make sure that you can execute following from each of the existing nodes of your cluster:

```
NET USE \\host_name\C$
```

Where the *host_name* is the public network name of the new node. You have administrative privileges on each node if the operating system responds with:

```
Command completed successfully.
```

5. To install the Oracle OSD clusterware when the new node already has a version of your vendor OSD clusterware installed, make sure the vendor OSD clusterware is stopped. Otherwise, continue to the next step.
6. Insert CD number one from the Oracle9i release 1 (9.0.1) CD set into the new node's CD-ROM drive.
7. Navigate to the `PREINSTALL\CLUSTERSETUP` directory, for example:

```
cd \PREINSTALL\CLUSTERSETUP
```

8. Run the Cluster Setup Wizard by entering the following from the `\PREINSTALL\CLUSTERSETUP` directory:

```
clustersetup
```

9. The cluster setup wizard should display its Welcome Page. Click **Next** to proceed.
10. Check the **Node Addition** option and click **Next**.
11. Provide a public name for the new node you want to add. If the existing cluster also uses a high speed private network, provide a name for the private network interface for the new node and click **Next**.
12. Click **Finish**.

See Also:

- Cluster Setup Wizard online help
- *Oracle9i Database installation guide for Windows* for more information

Create Disk Partitions

As part of the preinstall requirements before installing the Oracle software on a node for Oracle Real Application Clusters, create raw disk partitions. Create at least two new disk partitions to accommodate the redo logs for the new instance. Make these disk partitions the same size as the redo log partitions that you configured for the existing nodes' instances. If the database to which you want to add a node uses automatic undo management, then create an additional logical partition for the undo tablespace.

See Also: *Oracle9i Real Application Clusters Installation and Configuration* for more information on configuring raw partitions on Windows NT and Windows 2000 platforms

For Windows 2000 only, use a *basic disk* as an extended partition for creating raw partitions.

From one of the existing nodes of the cluster:

1. Start Disk Administrator using the path: Start >Program >Administrative Tools >Disk Administrator
2. Click inside an unallocated part of the extended partition.
3. Choose **Create** from the Partition menu. A dialog box appears in which you should enter the size of the partition. Ensure you use the same sizes as those you used on your existing nodes.
4. Click on the newly created partition and select **Assign Drive Letter** from the Tool menu.
5. Select **Don't Assign Drive Letter**, and click **OK**.
6. Repeat the last four steps, Steps 2 through 5, for the second and any additional partitions.
7. Select **Commit Changes Now** from the Partition menu to save the new partition information.
8. For Windows NT only, you may need to reboot all the nodes in your cluster if you are using a pre-4.0 Service Pack. This makes all of the nodes recognize the new partitions.

Create Links to Disk Partitions

Create the links to the partitions that you created in the "[Create Disk Partitions](#)" section so the existing nodes recognize these links. In addition, the new node

should also recognize the pre-existing symbolic links to logical drives as well as the new links. To create these links, perform the following steps from one of the existing nodes:

1. Start the Object Link Manager by typing the following from the `%ORACLE_HOME%\bin` directory:

```
GUIOracleOBJManager
```

The Object Link Manager automatically detects the symbolic links to the logical drives and displays them in OLM's graphical user interface (GUI).

2. Recall the disk and partition numbers for the partitions you created. Look for the disk and partition numbers in the OLM GUI and perform the following tasks:
 - a. Right-click next to the box under the **New Link** column and enter the link name for the first partition.
 - b. Repeat step a. for the second and any additional partitions.

For example, if your Real Application Clusters database name is `db` and it consists of two instances running on two nodes and you are adding a third instance on the third node, your link names for your redo logs are `db_redo3_1`, `db_redo3_2`, and so on.

If your existing database uses automatic undo management, then enter the link name the logical parathion for the undo tablespace. For example, if your Real Application Clusters database name is `db` and it consists of two instances running on two nodes and you are adding a third instance on the third node, your link names for the undo tablespace will be `db_undotbs3`.

3. Select **Commit** from the Options menu. This creates the new links on the current node.
4. Select **Sync Nodes** from the Options menu. This makes the new links visible to the existing nodes of the cluster.
5. Select **Exit** from the Options menu to exit the Object Link Manager.

Adding a Node at the Oracle Layer

At this point, you have added the new node to the cluster at the clusterware layer. To add a node to an existing cluster at the Oracle layer, on all platforms perform the following steps:

- [Install the Oracle Software on the New Node](#)
- [Post-Installation Steps](#)
- [Add a Database Instance on the New Node](#)

The following procedures explain these steps in detail.

Install the Oracle Software on the New Node Follow these steps to install the Oracle software on the *new node*:

1. If your platform supports a cluster file system then proceed to the next section titled "[Post-Installation Steps](#)" on page 9-10.
2. On the new node, insert the Oracle9i release 1 (9.0.1) CD-ROM into the new node's CD-ROM drive.
3. Run the installer using one of the following procedures for your platform.
 - On UNIX, run the Installer on by executing the `runInstaller` command from the root directory of the first Oracle CD. The Installer displays the Welcome page.
 - On Windows NT and Windows 2000, run the Installer by executing the `setup.exe` command. The installer displays the Welcome page.
4. On the File Locations page in the Path field under the Destination heading, enter the `ORACLE_HOME` into which you are going to install the Oracle Enterprise Edition and Real Application Clusters software. The `ORACLE_HOME` you enter should be the same as the `ORACLE_HOME` that the existing nodes use.
5. On the Available Products page, select **Oracle9i Database**.
6. On the Install Types page, select **Enterprise Edition**. The Installer displays the Database Configuration page.
7. Follow the OUI instructions to install the same products and options that you installed on the existing nodes. Or select the **Software Only** configuration option on the Database Configuration page.
8. On the Node Selection page, select only the current or new node name. Do not select other pre-existing node names; the Oracle software is already on them.

9. When the OUI completes the installation of the Oracle software, run the post installation script `root.sh`. Then exit the Oracle Universal Installer and proceed to the next section, "[Post-Installation Steps](#)".

Post-Installation Steps From the *new node*, perform the following post-installation steps:

1. Run the Oracle Net Configuration Assistant from the `bin` directory in `ORACLE_HOME` to configure a new **listener** for the node.
2. Complete any post-installation steps for your platform by referring to your platform-specific Oracle documentation.

Add a Database Instance on the New Node To add an instance on the new node, go to one of the *existing nodes* that has a running instance and follow these steps:

1. Make sure the **Global Services Daemon (GSD)** is running on all the existing nodes as well as on the new node. If the GSD is not running:

—On UNIX, enter the following command to start it:

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

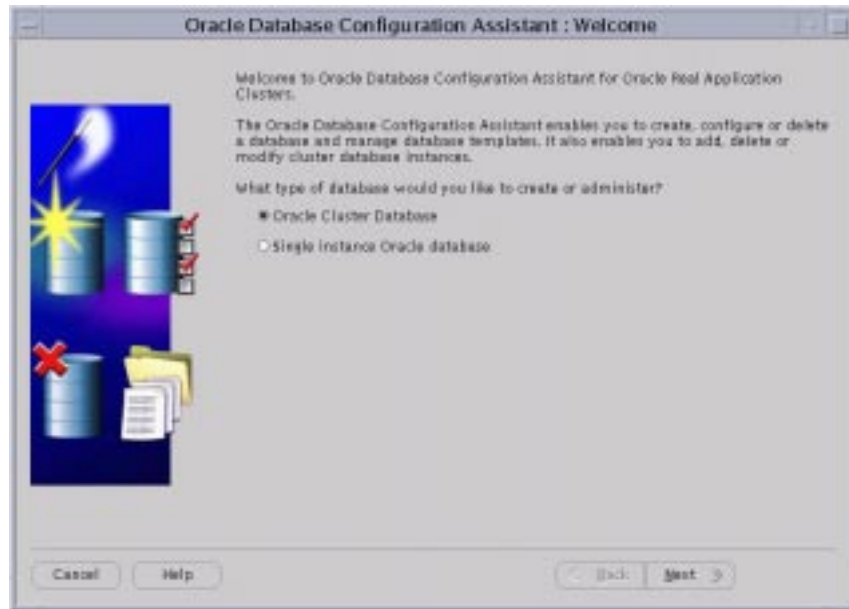
—On Windows NT and Windows 2000, enter the following to start the OracleGSDService:

```
%ORACLE_HOME%\bin\gsdservice -start
```

2. Launch the Oracle Database Configuration Assistant by typing `dbca` in the `bin` directory in `$ORACLE_HOME` on UNIX, or on Windows NT and Windows 2000, choose Start >Programs >Oracle for Windows NT/2000 - [HOME_NAME] >Database Administration >Database Configuration Assistant.

The DBCA wizard starts and displays the Welcome Page for Real Application Clusters shown in [Figure 9-1](#).

Figure 9-1 DBCA Welcome Page for Real Application Clusters



Note: If the Welcome Page for Real Application Clusters does not display, it means the Oracle Database Configuration Assistant was unable to:

- Communicate with the **Cluster Manager (CM)** software on Windows NT or Windows 2000
- Detect the **Global Cache Service (GCS)** software or the lists of nodes in the cluster on UNIX operating systems

To resolve this, refer to your vendor's operating system-dependent documentation and then restart the DBCA.

3. Select the **Oracle Real Application Clusters Database** option and click **Next**. After you click **Next**, the DBCA displays the Operations Page as shown in

Figure 9-2:

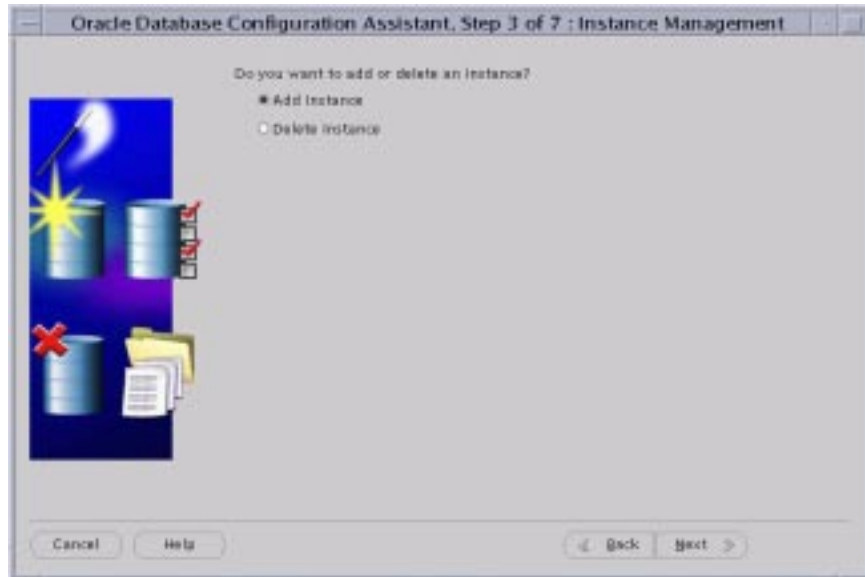
Figure 9-2 Operations Page



4. Select the **Instance Management** option and click **Next**.

After you click **Next**, the DBCA displays the Instance Management shown in [Figure 9-3](#).

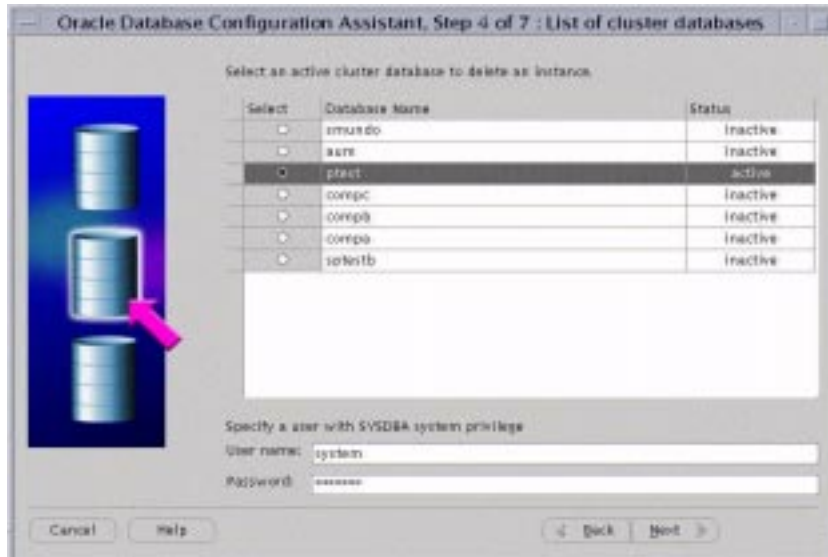
Figure 9-3 Instance Management Page



5. Select the **Add Instance** option and click **Next**.

After you click **Next**, the DBCA displays the List of Databases and their current status, such as **ACTIVE**, or **INACTIVE**, as shown in [Figure 9-4](#).

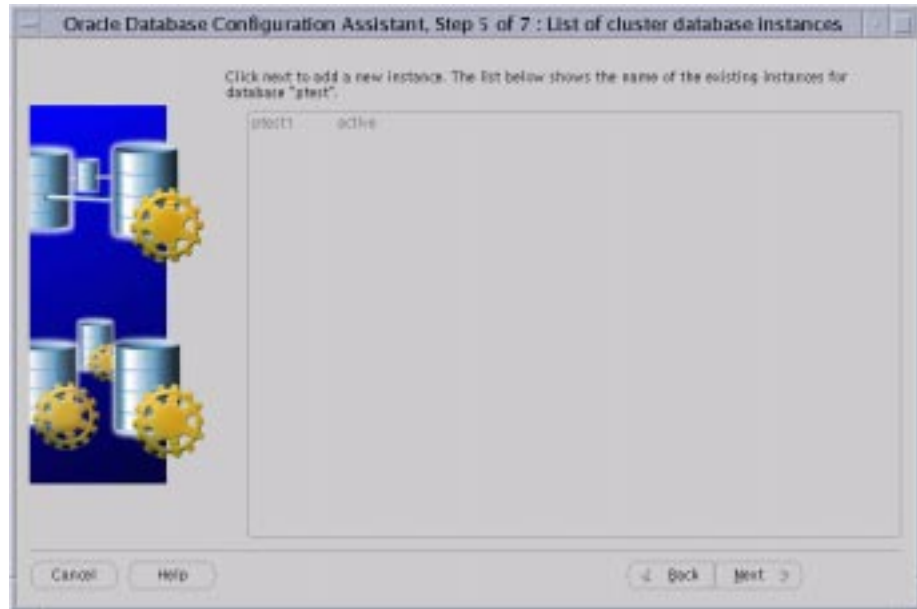
Figure 9-4 *List of Databases*



6. Select an active Real Application Clusters database name to add an instance to from the List of Databases page. If your user ID is not operating-system authenticated, the DBCA also prompts you for a user ID and password for a database user that has **SYSDBA** privileges. If prompted, enter a valid user ID and password. Click **Next**.

After you click **Next**, the DBCA displays the List of Instances Page showing the names of the instances that exist for the selected Real Application Clusters database as shown in [Figure 9-5](#).

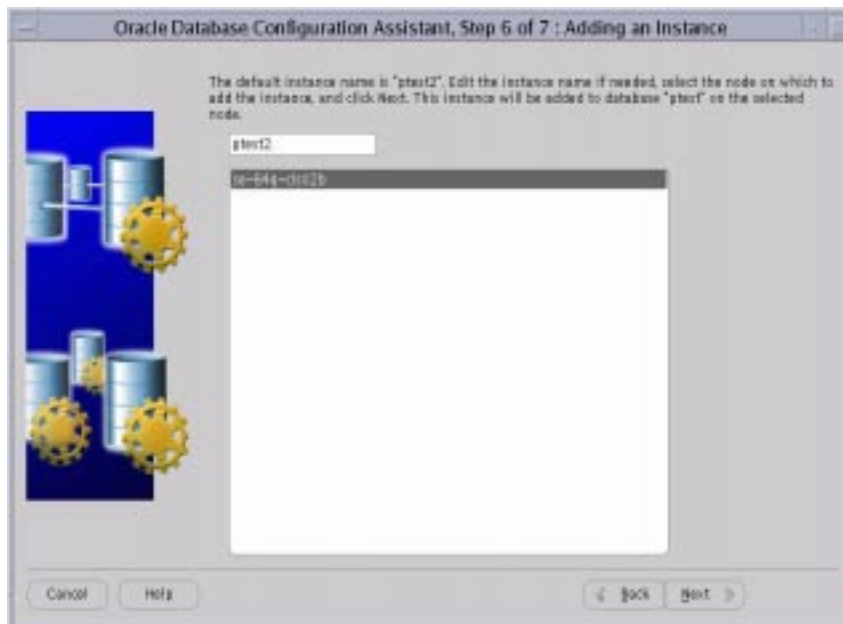
Figure 9-5 List of Instances



7. Click **Next**.

After you click **Next**, the DBCA displays the Adding an Instance page as shown in [Figure 9–6](#).

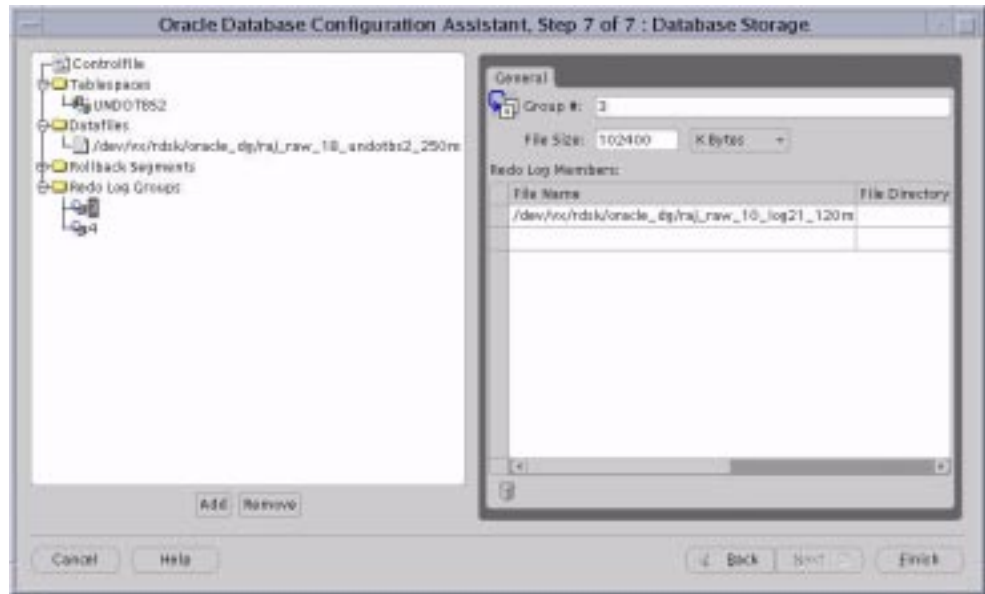
Figure 9–6 Adding an Instance



8. Enter the **instance name** in the field at the top of this page if the instance name that the DBCA provides does not match your existing instance name sequence. Then select the new node name from the list and click **Next**.

After you click **Next**, the DBCA displays the Storage Page as shown in [Figure 9-7](#).

Figure 9-7 Instance Storage Page



9. If your database uses **automatic undo management**, and your platform supports a cluster file system, then you can skip this step unless you want to change the default undo tablespace **data file** name that the DBCA provides. Otherwise, select the **Tablespaces** folder and expand it. Select the undo tablespace storage object. Another dialog will appear on the right-hand side. Change the default data file name to the raw device name (or change it to the cluster file system file name, if you do not want to use the default name) for the tablespace. Oracle Corporation recommends against changing the storage properties for the undo tablespace.
10. If your database uses rollback undo management, then select the **Rollback Segment** folder to change the rollback segment names if you do not want to use the default rollback segment names that the DBCA provides. Oracle Corporation recommends against changing the storage properties for the **rollback segments**.
11. If your platform supports a cluster file system, then skip this step unless you want to change the default redo log group file names. Otherwise, select the

Deleting Instances

The following procedures explain how to delete an instance with the Oracle Database Configuration Assistant. Where noted, refer to previous figures in this chapter. To delete an instance:

1. Go to the DBCA Operations Page shown in [Figure 9-2](#), select **Instance Management**, and click **Next**.

After you click **Next**, the DBCA displays the Instance Management Page shown in [Figure 9-9](#).

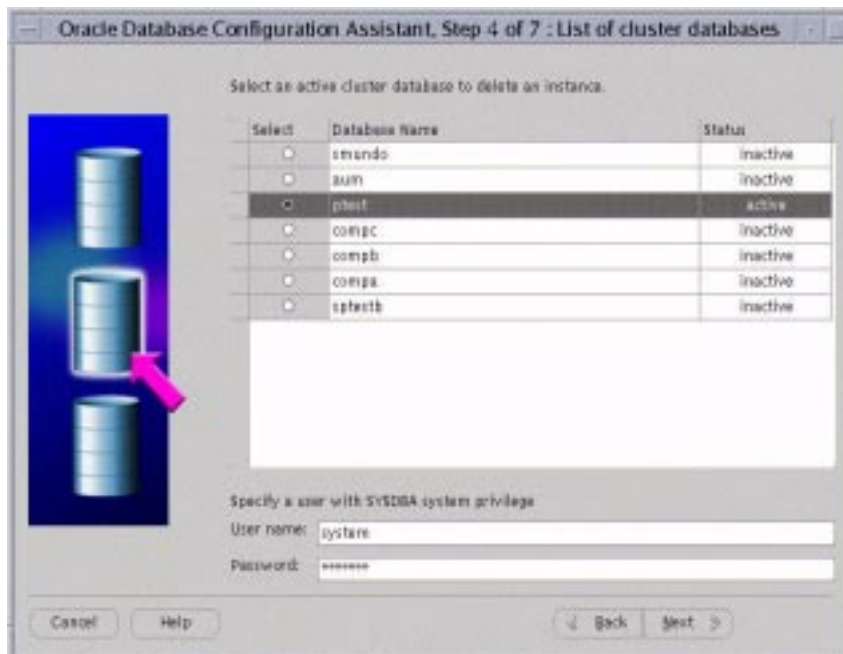
2. Select the **Delete Instance** option and click **Next**.

Figure 9-9 DBCA Instance Management Operation Selection Page



After you click **Next**, the DBCA displays the List of Databases page shown in [Figure 9-10](#).

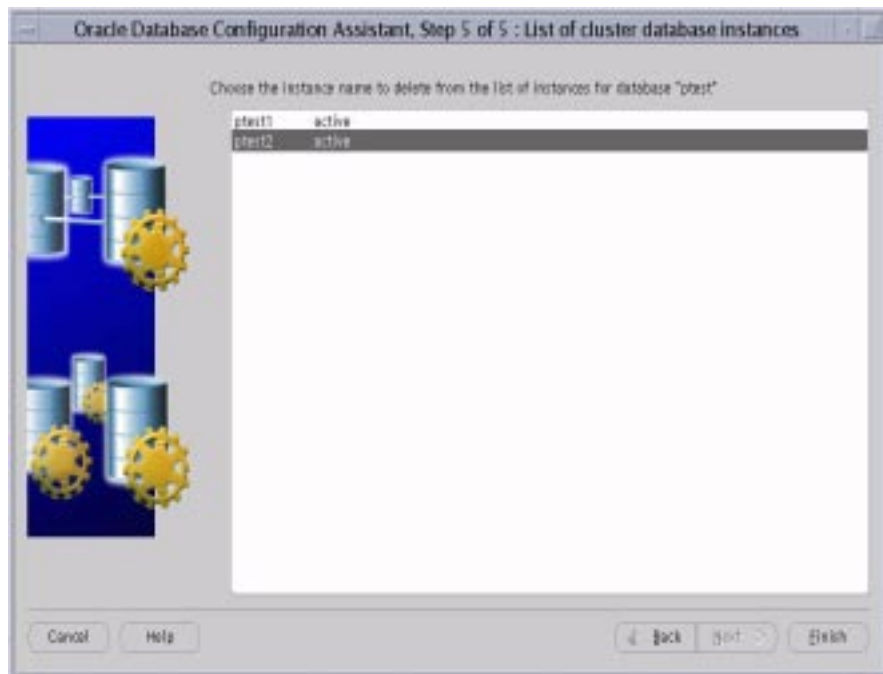
Figure 9-10 List of Databases



3. Select a Real Application Clusters database from which to delete an instance. If your user ID is not operating-system authenticated, then the DBCA also prompts you for a user ID and password for a database user that has SYSDBA privileges. If the DBCA prompts you for this, then enter a valid user ID and password. Click **Next**.

After you click **Next**, the DBCA displays the List of Instances Page shown in [Figure 9-11](#). The List of Instance Page shows the instances associated with the Real Application Clusters database that you selected as well as the status of each instance.

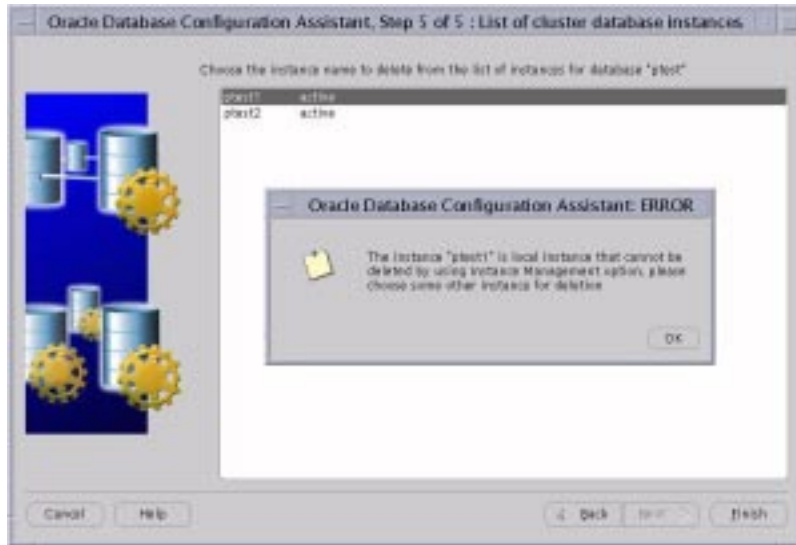
Figure 9-11 List of Instances



4. Select the instance you want to delete and click **Finish**.

- If you select the local instance, the DBCA displays a warning as shown in [Figure 9-12](#).

Figure 9-12 DBCA Warning Dialog for Selecting the Local Instance



- To proceed with the operation, click **OK** on the warning dialog and select an instance other than the local instance and click **Finish**.
- The DBCA displays a summary dialog as shown in [Figure 9-13](#). Click **OK** on the summary dialog.

Figure 9-13 DBCA Summary Dialog



After you click **OK**, the DBCA displays a progress dialog that shows the DBCA performing the instance deletion operation during which the DBCA removes the

instance and the instance's Oracle Net configuration. When the DBCA completes the deletion operation, the DBCA displays a dialog asking whether you want to perform another operation. Click **No** and exit the DBCA.

At this point, you have accomplished the following:

- Stopped the listeners associated with the selected instance
- Deleted the selected database instance from the instance's configured node
- Deleted the selected instance's services for Windows NT or Windows 2000
- Removed the Oracle Net configuration
- Deleted the Oracle Flexible Architecture directory structure from the instance's configured node

Part V

Reference

Part Five provides reference information for Real Application Clusters. The contents of Part Five are:

- [Appendix A, "Troubleshooting"](#)
- [Glossary](#)

Troubleshooting

This appendix explains how to manage trace files and how to contact Oracle Support Services. The topics in this appendix are:

- [Using Trace Files](#)
- [Contacting Oracle Support Services](#)

See Also: *Oracle9i Real Application Clusters Installation and Configuration* for information on troubleshooting **Oracle Enterprise Manager** Service Discovery failures.

Using Trace Files

This section discusses the following trace file subjects:

- [Background Thread Trace Files](#)
- [User Thread Trace Files](#)
- [Alert File](#)
- [Error Call Trace Stack](#)

Background Thread Trace Files

Real Application Clusters background threads use trace files to record occurrences and exceptions of database operations as well as errors. These detailed trace logs are helpful to Oracle support to debug problems in your **cluster** configuration. Background thread trace files are created regardless of whether the `BACKGROUND_DUMP_DEST` parameter is set in the **initdb_name.ora initialization parameter file**. If you set `BACKGROUND_DUMP_DEST`, the trace files are stored in the directory specified. If you do not set the parameter, the trace files are stored in:

- `$ORACLE_HOME/admin/db_name/bdump` on UNIX operating systems
- `%ORACLE_HOME%\admin\db_name\bdump` on Windows NT and Windows 2000 operating systems

The Oracle database creates a different trace file for each background thread. The name of the trace file contains the name of the background thread followed by the extension `.trc`, such as:

- `sidbwr.trc`
- `sidsmon.trc`

Trace information is reported in the trace files shown in [Table A-1](#):

Table A-1 Background Thread Trace Files

Trace File	Description
<code>sidbsp0.trc</code>	Trace file for Cache Fusion Global Cache Service (GCS) processing. This trace files shows errors associated with GCS.
<code>sidlckn.trc</code>	Trace file for the Global Cache Service Processes (LMSn) . This trace file shows lock requests for other background processes.
<code>sidlmdn.trc</code>	Trace file for the <code>LMDn</code> process. This trace file shows lock requests.
<code>sidlmon.trc</code>	Trace file for the LMON process. This trace file shows the status of the cluster.
<code>sidp00n.trc</code>	Trace file for the parallel execution processes.

User Thread Trace Files

Trace files are also created for user threads if the `USER_DUMP_DEST` parameter is set in the initialization parameter file. The trace files for the user threads have the form `oraxxxxx.trc`, where `xxxxx` is a 5-digit number indicating the **process identifier (PID)** on UNIX or the **thread ID** on Windows NT and Windows 2000.

Alert File

The **alert files**, `sidalert.log`, contains important information about error messages and exceptions that occur during database operations. Each **instance** has one alert

file; information is appended to the file each time you start the instance. All threads can write to the alert file.

The `sidalrt.log` file is in the directory specified by the `BACKGROUND_DUMP_DEST` parameter in the `initdb_name.ora` initialization parameter file. If you do not set the `BACKGROUND_DUMP_DEST` parameter, the `sidalrt.log` file is generated in:

- `$ORACLE_BASE/admin/db_name/bdump` on UNIX operating systems
- `%ORACLE_HOME%\admin\db_name\bdump` on Windows NT and Windows 2000 operating systems

Error Call Trace Stack

Oracle Worldwide Support might ask you to create an error call trace stack for a particular trace file. An error call trace stack provides a program trace of specific background or user threads in the database.

To create an error call trace stack:

1. Obtain the Oracle process ID for the background processes:

```
SELECT pid "Oracle Process Id",
       name
   from v$PROCESS, V$BGPROCESS
  where V$PROCESS.addr = V$BGPROCESS.paddr;
```

The output displayed should resemble the following:

```
Oracle Pro NAME
-----
      2 PMON
      3 LMON
      4 DBW0
      5 LGWR
      6 CKPT
      7 SMON
      8 RECO
      9 SNP0
     10 SNP1
     11 GCS0
```

2. Dump the trace stack to the trace file. For example, to dump the trace stack of LMON, enter:
 - a. Set the Oracle process ID to LMON. The process ID is 3 in this example:

```
ORADEBUG setorapid 3
```

- b. Dump the error stack to *sid1mon.trc*:

```
ORADEBUG dump errorstack 3
```

Contacting Oracle Support Services

If after reading this appendix, you still cannot resolve your problems, call Oracle Support Services to report the error. Please have the following information available:

- Your cluster hardware specifications, for example, a two-**node** cluster of Dell PowerEdge 6100 servers
- Your operating system version
- All five digits of the release number of your Oracle RDBMS (for example, 9.0.1.0.0)
- All five digits in the release number of your Oracle9i Real Application Clusters database
- On Windows NT and Windows 2000, the contents of HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\OSD key
- Cluster OSD upgrades from vendor
- Information about the particular operation that failed, for example, database startup or query
- A list of steps that reproduce the problem

Severe Errors

If an ORA-600 error appears, it will appear in the *sidalrt.log* file. If an ORA-600 error or any other severe errors appear in the *sidalrt.log* file, then provide all files in:

- `$ORACLE_HOME/admin/db_name/bdump` on UNIX operating systems
- `%ORACLE_HOME%\admin\db_name\bump` on Windows NT and Windows 2000 operating systems

Glossary

alert files

The alert files contain important information about error messages and exceptions that occur during Oracle database operations.

application programming interface (api)

The interface (or *calling conventions*) that an application program accesses operating system and other services. An API is defined at the source code level and provides a level of abstraction between the application and the kernel (or other privileged utilities) to ensure the portability of the code. A typical example of an API is the Oracle Call Interface (OCI).

auto-discovery

When you execute the Discover Node command from the **Console**, the Management Server contacts the Oracle Intelligent Agent installed on that **node** to discover the Oracle services installed on the node. The Management Server then places the new information in the repository, and updates the hierarchical tree in the Navigator window of the Console, displaying a broad view of all nodes and their respective services.

automatic undo management

A feature that automatically manages the sizing of undo tablespaces.

cache coherency

The synchronization of data in multiple caches so that reading a memory location by way of any cache will return the most recent data written to that location by way of any other cache. Sometimes also called *cache consistency*.

Cache Fusion

A *diskless* cache coherency mechanism used in **Real Application Clusters** that provides copies of blocks directly from the holding instance's memory cache to the requesting instance's memory cache. Cache Fusion is the collective term for the group of features that were introduced starting with Oracle8i Parallel Server.

cluster

A cluster is a set of instances, each typically running on different **nodes**, that coordinate with one another when accessing the shared database residing on disk.

Cluster Manager (CM)

Cluster Manager is an operating system-dependent component that discovers and tracks the membership state of nodes by providing a common view of cluster membership across the cluster. CM monitors process health. The LMON process, a background process that monitors the health of the **Global Cache Service (GCS)**, registers and de-registers from CM.

CM

See: **Cluster Manager (CM)**.

connect descriptor

A specially formatted description of the destination for a network connection. A connect descriptor contains destination service and network route information.

The destination service is indicated by using its **service name** for Oracle release 8.1 databases or greater or its **Oracle system identifier (sid)** for Oracle release 8.0 or version 7 databases. The network route provides, at a minimum, the location of the listener through use of a network address.

connect-time failover

See: **failover**.

connection load balancing

A feature that balances the number of active connections among various instances and **shared server dispatchers** for the same service. Because of service registration's ability to register with remote listeners, a listener is always aware of all instances and dispatchers regardless. This way, a listener can send an incoming client request for a specific service to the least loaded instance and least loaded dispatcher regardless of its location.

control file

A file that records the physical structure of a database and contains the database name, the names and locations of associated databases and online redo log files, the timestamp of the database creation, the current log sequence number, and checkpoint information.

Console

The **Oracle Enterprise Manager** Console gives you a central point of control for the Oracle environment through an intuitive graphical user interface (GUI) that provides powerful and robust system management. The Console provides menus, tool bars, and launch palettes that enable access to Oracle tools. The Console consists of four separate windows. See also: **Navigator Window**, **Group Window**, **Event Management Window**, and **job window**.

contention

Competition for resources. The term is used in networks to describe a situation where two or more **nodes** attempt to access the same resource at the same time. Contention is resolved by concurrency control mechanisms.

CR

Consistent Read.

daemon

Disk and Execution Monitor. A program that is not invoked explicitly, but lies dormant waiting for some condition to occur.

Database Administrator (DBA)

An individual responsible for the design and management of the database and for the evaluation, selection and implementation of the database management system. In smaller organizations, the data administrator and database administrator are often one in the same; however, when they are different, the database administrator's function is more technical. The database administrator would implement the database software that meets the requirements outlined by the organization's data administrator and systems analysts. DBA is also an acronym for Data Block Address.

data file

A data file is a physical operating system file on disk that was created by Oracle. It contains data structures such as tables and indexes. A data file can only belong to one database.

Data Gatherer

See: [Oracle Data Gatherer](#).

DBA

See: [Database Administrator \(DBA\)](#). Outside the context of Real Application Clusters, DBA is also an acronym for Data Block Address.

dedicated server

A server that requires a dedicated server process for each user process. There is one server process for each client. Oracle Net sends the address of an existing server process back to the client. The client then resends its connect request to the server address provided. Contrast with [shared server](#).

decision support system (DSS)

Database and application environments that help with decision support or data warehousing systems.

dispatcher

A process that enables many clients to connect to the same server without the need for a [dedicated server](#) process for each client. A dispatcher handles and directs multiple incoming network session requests to shared server processes. See also [shared server](#).

distributed recovery

The process of recovery over a cluster.

DSS

See: [decision support system \(DSS\)](#).

enqueuees

Enqueuees are shared memory structures that serialize access to database resources and are associated with a session or transaction. In Real Application Clusters, enqueuees can be global to a database. Enqueuees are local to one instance if Real Application Clusters is not enabled.

Enterprise Manager

See: [Oracle Enterprise Manager](#).

Enterprise Manager Configuration Assistant (EMCA)

A tool for creating, deleting, and modifying **Oracle Enterprise Manager** configurations and settings.

Event Management Window

Part of the **Console**. The Event Management window enables the administrator to remotely monitor critical database and system events.

exclusive (X) access mode

A write-only global block access mode. In this mode no other access is allowed.

extent

An extent is a specific number of contiguous data blocks, obtained in a single allocation, used to store a specific type of information.

extent allocation

The process used to assign a number of contiguous data blocks.

failover

The process of failure recognition and recovery.

free list groups

Free list groups are sets of free lists available for use by one or more instances.

Global Cache Service (GCS)

Real Application Clusters software that provides mechanisms to control the allocation and modification of Oracle resources.

Global Cache Service Processes (LMS_n)

Global Cache Service processes handle remote messages for the **Global Cache Service (GCS)**. Current Real Application Clusters software provides for up to 10 Lock Manager Servers. The number of Lock Manager Servers varies depending on the amount of messaging traffic among **nodes** in the cluster. The LMS_n handle Blocking Asynchronous Traps from the remote instance for GCS resources. For cross-instance read/write requests, the LMS_n will create a consistent read version of the block and send it to the requesting instance. The LMS_n also control the flow of messages to remote instances.

Global Cache Service (GCS) resources

Global Cache Service resources are global resources that coordinate access to data blocks in the buffer caches of multiple **Real Application Clusters** instances to provide cache coherency.

global database name

The global database name is the full name of the database that uniquely identifies it from any other database. The global database name is of the form `database_name.database_domain`—for example: `OP.US.ORACLE.COM`

The database name portion, `OP`, is a simple name you want to call your database. The database domain portion, `.US.ORACLE.COM`, specifies the database domain where the database is located, making the global database name unique. When possible, Oracle Corporation recommends that your database domain mirror the network domain.

The global database name is the default service name of database, as specified by the `SERVICE_NAMES` parameter in the `INITDB_NAME.ORA` file.

global dynamic performance views (GV\$)

Dynamic performance views storing information about *all* open instances in a Real Application Clusters cluster. (Not only the local instance.) In contrast, standard dynamic performance views (V\$) only store information about the local instance.

Global Enqueue Service (GES)

The Global Enqueue Service coordinates enqueues that are shared globally.

Global Enqueue Service Daemon (LMD)

The Global Enqueue Service Daemon is the resource agent process that manages requests for **Global Cache Service (GCS)** resources to control access to blocks. The LMD process also handles deadlock detection and remote resource requests. Remote resource requests are requests originating from another instance.

Global Enqueue Service Monitor (LMON)

The background LMON process monitors the entire cluster to manage global resources. LMON manages instance *deaths* and the associated recovery for the **Global Cache Service (GCS)**. In particular, LMON handles the part of recovery associated with global resources. LMON-provided services are also known as Cluster Group Services.

Globalization Support

The Globalization Support architecture enables you to store, process, and retrieve data in native languages. It ensures that database utilities and error messages, sort order, date, time, monetary, numeric, and calendar conventions automatically adapt to the native language and locale.

global resources

Inter-instance synchronization mechanisms designed to provide cache coherency for Real Application Clusters. The term can refer to both **Global Cache Service (GCS)** resources and **Global Enqueue Service (GES)** resources.

Global Services Daemon (GSD)

Formerly OPSD. The Global Services Daemon is a component that receives requests from SRVCTL to execute administrative job tasks, such as startup or shutdown. The command is executed locally on each **node**, and the results are sent back to SRVCTL. GSD is installed on the nodes by default. Do not delete the GSD process.

Group Window

Part of the **Console**. The Group window provides a customized graphical representation of key objects. It is created by the administrator.

GSD

See **Global Services Daemon (GSD)**.

GV\$

See: **global dynamic performance views (GV\$)**.

high availability

High availability refers to systems with redundant components that provide consistent and uninterrupted service, even in the event of hardware or software failures. This involves some degree of redundancy.

HOST command

A SQL*PLUS command to move from the SQL*Plus command line to the operating system command line.

hybrid

A hybrid database is one that has both OLTP and Data Warehouse processing characteristics.

IFILE

A parameter designating the continuation file of an **initialization parameter file**.

initialization parameter file

Files that contains information to initialize the database such as **spfile.ora**, **initdb_name.ora** and **initsid.ora**.

initsid.ora

An instance initialization file that contains parameters unique for an instance and points to **initdb_name.ora** for database parameters.

initdb_name.ora

A common database initialization file shared among the instance that contains database parameters.

instance

For a **Real Application Clusters** database, each **node** within a cluster usually has one instance of the running Oracle software that references the database. When a database is started, Oracle allocates a memory area called the **System Global Area (SGA)** and starts one or more Oracle processes. This combination of the SGA and the Oracle processes is called an instance. The memory and processes of an instance manage the associated database's data and serve the one or more users of the database. You can connect to any instance to access information within a Real Application Clusters database.

Each instance has unique Oracle System Identifier (*sid*), instance name, rollback segments, and thread ID.

instance membership recovery (IMR)

Instance Membership Recovery (IMR) is the method used by Real Application Clusters guaranteeing that all cluster members are functional or *active*. IMR polls and arbitrates the membership. Any members that do not show a heartbeat by way of the control file or who do not respond to periodic *are you alive?* messages are presumed *dead*.

instance name

Instance name represents the name of the instance and is used to uniquely identify a specific instance when multiple instances share common services names. The instance name is identified by the `INSTANCE_NAME` parameter in the `INITsid.ORA` file.

The instance name is the same as the Oracle System Identifier (*sid*). See also: [Oracle system identifier \(sid\)](#).

instance number

A number that associates extents of data blocks with particular instances. The instance number enables you to start up an instance and ensure that it uses the extents allocated to it for inserts and updates. This will ensure that it does not use space allocated for other instances. The instance cannot use data blocks in another free list unless the instance is restarted with that instance number.

You can use various SQL options with the `INSTANCE_NUMBER` initialization parameter to associate extents of data blocks with instances.

The instance number is depicted by the `INSTANCE_NUMBER` parameter in the instance initialization file, `initsid.ora`.

Intelligent Agent

See: [Oracle Intelligent Agent](#).

interconnect

The communication link between nodes.

Inter-Process Communication (IPC)

Inter-Process Communication is an operating system-dependent component that provides the reliable transfer of messages between instances on different nodes.

job window

Part of the [Console](#). The Job window enables the administrator to automate repetitive activities.

latch

A latch is a low level serialization mechanism that protects in-memory data structures in the [System Global Area \(SGA\)](#). Latches do not protect data files, are automatic, and are held for a very short time in exclusive mode. Because latches are synchronized within a **node**, they do not facilitate internode synchronization.

LDAP

See [Lightweight Directory Access Protocol \(LDAP\)](#).

Lightweight Directory Access Protocol (LDAP)

A protocol for accessing on-line directory services.

listener

The listener is a separate process that resides on the server whose responsibility is to listen for incoming client connection requests and manage the traffic to the server.

The listener brokers the client request, handing off the request to the server. Every time a client (or server acting as a client) requests a network session with a server, a listener receives the actual request. If the client's information matches the listener's information, the listener grants a connection to the server.

listener.ora

A configuration file for the listener that identifies the listener name and protocol addresses that the listener is accepting connection requests on services it is listening for.

The `listener.ora` file typically resides in `$ORACLE_HOME/network/admin` on UNIX platforms and `%ORACLE_HOME%\network\admin` on Windows NT and Windows 2000.

Oracle databases do not require identification of the database service because of [service registration](#). However, static service configuration is required for an Oracle databases if you plan to use [Oracle Enterprise Manager](#).

LMD

See [Global Enqueue Service Daemon \(LMD\)](#).

LMON

See [Global Enqueue Service Monitor \(LMON\)](#).

LMS_n

See [Global Cache Service Processes \(LMS_n\)](#).

Management Server

The [Oracle Enterprise Manager](#) Management Server provides centralized intelligence and distributed control between the [Console](#) and the managed [nodes](#), and processes system management tasks sent by the Console and administers the distribution of these tasks across the enterprise. The Management Server stores all system data, application data, and information about the state of managed nodes in a repository. The repository is a set of tables stored in a database. High performance and scalability is ensured because the workload is automatically shared and balanced when there are multiple Management Servers.

master free list

A list of blocks containing available space drawn from any extent in a table.

mean time between failures (MTBF)

The average time (usually expressed in hours) that a component works without failure. It is calculated by dividing the total number of failures into the total number of operating hours observed. The term can also mean the length of time a user can reasonably expect a device or system to work before an failure occurs.

mean time to failure (MTTF)

The average period of time that a component will work until failure.

mean time to recover (MTTR)

The average time that it takes to get a failed piece of hardware back on line. Outside the context of Real Application Clusters, the acronym MTTR is also used for Mean Time to Repair.

memory

The component used for programmatic execution and the buffering of data.

MTBF

See: [mean time between failures \(MTBF\)](#).

MTS

See [shared server](#).

MTTF

See: [mean time to failure \(MTTF\)](#).

MTTR

See: [mean time to recover \(MTTR\)](#).

multiple oracle homes

The capability of having more than one Oracle home directory on a machine.

multi-threaded server (MTS)

See [shared server](#).

N

See: [null \(N\)](#).

naming method

The method used by a client application to resolve a [net service name](#) to a connect descriptor.

Navigator Window

Part of the [Console](#). The Navigator window contains an object browser that provides an organized, consistent, and hierarchical view of the database objects in the network.

net service name

A simple name for a service that resolves to a connect descriptor. Users initiate a connect request by passing a user name and password along with a net service name in a connect string for the desired service:

```
CONNECT username/password@net_service_name
```

Depending on your needs, net service names can be stored in a variety of places, including:

- Local configuration file, `tnsnames.ora`, on each client
- Directory server
- Oracle Names server
- Active Directory Services
- External naming service, such as Novell Directory Services (NDS), Network Information Service, or Cell Directory Service (CDS)

Net8

See [Oracle Net](#).

network file system (NFS)

A protocol developed by Sun Microsystems, and defined in RFC 1094, that allows a computer to access files over a network as if the files were on local disks.

node

A node is a machine where an instance resides.

null (N)

Null indicates that an access mode is not assigned to a block or resource.

OLTP

See: [online transaction processing \(OLTP\)](#).

online transaction processing (OLTP)

The processing of transactions by computers in real time.

operating system-dependent (OSD) layer

A software layer that consists of several software components developed by vendors. The OSD layer maps the key operating system/cluster-ware services required for proper operation of [Real Application Clusters](#).

OPCTL

See [Server Control \(SRVCTL\)](#).

OPSM

See: [Server Management \(SRVM\)](#).

Oracle Database Configuration Assistant (DBCA)

A tool that enables you to create, delete, and modify a database as well as add and delete instances.

Oracle Data Gatherer

The Oracle Data Gatherer collects performance statistics for the Oracle Performance Manager. The Oracle Data Gatherer is part of the part of the Oracle Intelligent Agent process. The Oracle Data Gatherer must be installed on a [node](#) somewhere on the network.

Oracle Enterprise Edition

Oracle Enterprise Edition is an Object-Relational Database Management System (ORDBMS). It provides the applications and files to manage a database. All other Real Application Clusters components are layered on top of Oracle Enterprise Edition.

Oracle Enterprise Manager

A system management tool that provides an integrated solution for centrally managing your heterogeneous environment. Oracle Enterprise Manager combines a

graphical [Console](#), [Management Server](#), [Oracle Intelligent Agent](#), [repository database](#), and tools to provide an integrated, comprehensive systems management platform for managing Oracle products.

A product family consists of system management tools designed to efficiently manage the complete Oracle environment.

Oracle Intelligent Agent

A process that runs on each of the nodes. It functions as the executor of jobs and events sent by the [Console](#) by way of the [Management Server](#). High availability is ensured since the agent can function regardless of the status of the [Console](#) or network connections.

Oracle Net

Oracle Net is the foundation of Oracle's family of networking products, allowing services and their applications to reside on different computers and communicate as peer applications. The main function of Oracle Net is to establish network sessions and transfer data between a client machine and a server or between two servers. Once a network session is established, Oracle Net acts as a data courier for the client and the server.

Oracle Parallel Server

See [Oracle9i Real Application Clusters](#).

Oracle Parallel Server Communication Daemon (OPSD)

See [Global Services Daemon \(GSD\)](#).

Oracle Parallel Server Management (OPSM)

See [Server Management \(SRVM\)](#).

Oracle Performance Manager

An add-on application for [Oracle Enterprise Manager](#) that offers tabular and graphic performance statistics for [Real Application Clusters](#). The statistics represent the aggregate performance for all instances running on Real Applications.

Oracle9i Real Application Clusters

See [Real Application Clusters](#).

Oracle system identifier (*sid*)

The Oracle System Identifier (*sid*) identifies a specific instance of the running Oracle software. For an Real Application Clusters database, each **node** within the cluster has an instance referencing the database.

The database name, specified by the `DB_NAME` parameter in the `INITDB_NAME.ORA` file, and unique thread ID make up each node's *sid*. The thread ID starts at 1 for the first instance in the cluster, and is incremented by 1 for the next instance, and so on. See also: **instance name**.

OSD

See **operating system-dependent (OSD) layer**.

Oracle Universal Installer (OUI)

A tool to install the Oracle relational database software. You can also use the Oracle Universal Installer to launch the **Oracle Database Configuration Assistant (DBCA)**.

parameter file (PFILE)

A file used by the Oracle server that provide specific values and configuration settings that are used on database startup. The keyword `PFILE` is used in the startup command.

password file

A file created by the `ORAPWD` command. A database must use password files if you want to connect as `SYSDBA` over a network.

Performance Manager

See: **Oracle Performance Manager**.

PMON process

See: **Process Monitor (PMON)**.

preferred credentials

Each **Oracle Enterprise Manager** administrator can set up specific user names, passwords, and roles for nodes, listeners, databases, and other services that you administer in the network.

After these credentials are set up, you log in once to start the **Console** and are then automatically logged in as needed to the nodes. All login credentials are encrypted in the **repository database**.

process identifier (PID)

An integer used by the operating system kernel to uniquely identify a process.

Process Monitor (PMON)

A process monitor database process that performs process recovery when a user process fails. PMON is responsible for cleaning up the cache and freeing resources that the process was using. PMON also checks on dispatcher and server processes and restarts them if they have failed. As a part of [service registration](#), PMON registers instance information with the listener.

RAID

See: [Redundant Array of Independent Disks \(RAID\)](#).

reader/writer contention

In Real Application Clusters, if a session (reader) needs to read a data block that has been recently modified by another session in a different instance (writer), a concurrency control mechanism takes place to ensure read consistency. This mechanism involves [Cache Fusion](#).

Real Application Clusters

An architecture that allows multiple instances to access a shared database of data files. Real Application Clusters is also a software component that provides the necessary Real Application Clusters scripts, initialization files, and data files to make the Oracle Enterprise Edition and Real Application Clusters. Note: the Real Application Clusters product was formerly called Oracle Parallel Server.

Recovery Manager (RMAN)

An Oracle tool that enables you to back up, copy, restore, and recover data files, control files, and archived redo logs. It is included with the Oracle server and does not require separate installation. You can invoke RMAN as a command line utility from the operating system (O/S) prompt or use the GUI-based Enterprise Manager Backup Manager.

redo log file

Redo files contain records of all changes made to data in the database buffer cache. Every Oracle database has a set of two or more redo log files. The set of redo log files for a database is collectively known as the database's redo log. A redo log is made up of redo entries (also called redo records). Each of these is a group of change vectors describing a single atomic change to the database. The primary function of the redo log is to record all changes made to data. Should a failure

prevent modified data from being permanently written to the data files, the changes can be obtained from the redo log and work is never lost. Redo log files are critical in protecting a database against failures. To protect against a failure involving the redo log itself, Oracle allows a multiplexed redo log so that two or more copies of the redo log can be maintained on different disks.

redo log kick

See [redo log switch](#).

redo log switch

When Oracle switches from using one online archive redo log to using another. All records generated to this point are switched out and archived.

Redundant Array of Independent Disks (RAID)

Most often a hardware architecture that combines multiple hard disk drives to allow rapid access to a large volume of stored data. You can also implement RAID as a software architecture, although this is less common.

repository database

A repository database is a set of tables in an Oracle database that stores data required by [Oracle Enterprise Manager](#). This database is separate from the database on the nodes.

RMAN

See: [Recovery Manager \(RMAN\)](#).

rollback segments

Rollback segments are records of old values of data that were changed by each transaction (whether committed or not committed.) Rollback segments are used to provide read consistency, to roll back transactions, and to recover the database. Each **node** typically has two rollback segments, that are identified with a naming convention of `RBSthread_id_rollback_number` by the `ROLLBACK_SEGMENTS` parameter in the `INITsid.ORA` file.

scalability

Scalability is the ability to add additional nodes to Real Application Clusters applications and achieve markedly improved scale-up and speed-up.

seed database

A preconfigured, ready-to-use database that requires minimal user input to create. See also: [starter database](#).

Server Control (SRVCTL)

Formerly OPSCTL. Server Management uses the SRVCTL utility (installed on each node) to manage configuration information that is used by some Oracle tools. For example, SRVCTL serves as a single point of control between the Oracle Intelligent Agent and the nodes. Only one node's Oracle Intelligent Agent is used to communicate with SRVCTL. SRVCTL on that [node](#) then communicates to the other nodes through Oracle Net.

Server Management (SRVM)

A comprehensive and integrated system management solution for Real Application Clusters. Server Management enables you to manage multi-instance databases running in heterogeneous environments through an open client/server architecture through [Oracle Enterprise Manager](#).

In addition to managing cluster databases, Server Management enables you to schedule jobs, perform event management, monitor performance, and obtain statistics to tune cluster databases. Note that SRVM is not a separate product. It is an integral part of [Oracle Enterprise Manager](#).

server parameter file

A binary parameter file that resides on the Oracle Server. This file contains parameter settings that are both global and instance-specific. These parameter settings are persistent across instance shutdown and instance startup events.

service discovery

When you execute the Discover Node command from the [Console](#), the [Management Server](#) contacts the [Oracle Intelligent Agent](#) installed on that node to discover the Oracle services installed on the node. The Management Server then places the new information in the repository and updates the hierarchical tree in the Navigator window of the [Console](#), displaying a broad view of all nodes and their respective services.

service name

A Service name is a logical representation of a database. This is the way a database is presented to clients. A database can be presented as multiple services and a service can be implemented as multiple database instances. The service name is a string that is the global database name, a name comprised of the database name

(DB_NAME) and domain name (DB_DOMAIN), entered during installation or database creation.

If you are not sure what the global database name is, you can obtain it from the combined values of the SERVICE_NAMES parameter in the common database initialization file, `initdbname.ora`.

The service name is included in the CONNECT_DATA part of the connect descriptor.

service registration

A feature whereby the **Process Monitor (PMON)** automatically registers information with a listener. Because this information is registered with the listener, the `listener.ora` file does not need to be configured with this static information.

Service registration provides the listener with the following information:

- Service name(s) for each running instance of the database
- Instance name(s) of the database
- Service handlers (dispatchers and dedicated servers) available for each instance
This allows the listener to direct a client's request appropriately.
- Dispatcher, instance, and node load information
This load information allows the listener to determine which dispatcher can best handle a client connection's request. If all dispatchers are blocked, the listener can spawn a dedicated server for the connection.

SGA

See: **System Global Area (SGA)**.

shared current (scur)

The buffer state name for shared access mode to a block.

shared mode (s)

Shared is a protected read block access mode. No writes are allowed in shared mode. In shared mode, any number of users can have simultaneous read access to a resource. See also: **exclusive (X) access mode**.

shared server

A server that is configured to allow many user processes to share very few server processes, so the number of users that can be supported is increased. With shared server configurations, many user processes connect to a dispatcher. The dispatcher

directs multiple incoming network session requests to a common queue. An idle shared server process from a shared pool of server processes picks up a request from the queue. This means a small pool of server processes can server a large amount of clients. shared server is the opposite of dedicated server.

sid

See: [Oracle system identifier \(sid\)](#).

spfile.ora

The binary parameter file that resides on the Oracle Server.

snapshot control file

RMAN creates a snapshot control file to resynchronize from a read-consistent version of the control file. This is a temporary snapshot control file. RMAN only needs a snapshot control file when resynchronizing with the recovery catalog or when making a backup of the current control file.

sqlnet.ora File

A configuration file for the client or server that specifies:

- Client domain to append to unqualified service names or net service names
- Order of naming methods the client should use when resolving a name
- Logging and tracing features to use
- Route of connections
- Preferred Oracle Names servers
- External naming parameters
- Oracle Advanced Security parameters

The `sqlnet.ora` file typically resides in the `$ORACLE_HOME/network/admin` directory on UNIX platforms and in the `%ORACLE_HOME%\network\admin` directory on Windows platforms.

SRVCTL

Formerly OPSCTL. See [Server Control \(SRVCTL\)](#).

starter database

See: [seed database](#).

startup (START)

Startup is an operating system-dependent component that provides one-time configuration to startup functionality.

SYSDBA

SYSDBA is a special database administration role that contains all system privileges with the `ADMIN OPTION`, and the `SYSOPER` system privilege. SYSDBA also permits `CREATE DATABASE` and time-based recovery.

SYSOPER

SYSOPER is a special database administration role that permits a database administrator to perform any of the following commands: `STARTUP`, `SHUTDOWN`, `ALTER DATABASE OPEN/MOUNT`, `ALTER DATABASE BACKUP`, `ARCHIVE LOG`, and `RECOVER`. SYSOPER includes the `RESTRICTED SESSION` privilege.

System Global Area (SGA)

A group of shared memory structures that contain data and control information for one Oracle database instance. The SGA and Oracle processes constitute an Oracle instance. Oracle automatically allocates memory for an SGA whenever you start an instance and the operating system reclaims the memory when you shut down the instance. Each instance has only one SGA.

TAF

See: [transparent application failover \(TAF\)](#).

thread

Each Oracle instance has its own set of online redo log groups. These groups are called a thread of online redo. In non-Real Application Clusters environments, each database has only one thread that belongs to the instance accessing it. In Real Application Clusters environments, each instance has a separate thread, that is, each instance has its own online redo log. Each thread has its own current log member.

thread ID

The number of the redo thread that is to be used by the instance. The thread starts at 1 node for the first instance in the cluster, and is incremented by 1 for the next instance, and so on. Threads are declared by the `THREAD` parameter in the `INITsid.ORA` file. When Oracle generates redo log files, the files include the thread ID. This enables you to easily identify a particular node's log files.

tnsnames.ora file

A file that contains net service names. This file is needed on clients, nodes, the [Console](#), and the Oracle Performance Manager machine.

trace files

Each server and background process can write to an associated trace file. When an internal error is detected by a process, it dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, while other information is intended for Oracle Support Services. Trace file information is also used to tune applications and instances.

transaction free list

A list of blocks freed by uncommitted transactions.

transparent application failover (TAF)

A runtime failover for high-availability environments, such as Real Application Clusters and Oracle Real Application Clusters Guard, TAF refers to the failover and re-establishment of application-to-service connections. It allows client applications to automatically reconnect to the database if the connection fails, and optionally resume a SELECT statement that was in progress. This reconnect happens automatically from within the Oracle Call Interface (OCI) library.

Virtual Interface Architecture (VIA)

A memory-based networking interface.

X

See: [exclusive \(X\) access mode](#).

Index

A

- abort mode, warning notice, 5-10
- ACTIVE_INSTANCE_COUNT parameter, 2-12
- adding nodes to a cluster, 9-2
- administering instances
 - with Server Management, 5-2
- administration
 - issues, general, 1-2
 - with SQL and SQL*Plus, 4-10
 - with SRVCTL, 4-2
- affinity
 - awareness, 7-8
- ALERT file, 8-4
- alert log
 - managing, 3-9
- alert logs, B-2
- ALLOCATE EXTENT option
 - DATAFILE option, 3-11
 - instance number, 3-10
 - INSTANCE option, 3-11
 - SIZE option, 3-11
- allocation
 - free space, 3-11
 - sequence numbers, 3-12
- ALTER DATABASE ADD LOGFILE
 - statement, 4-17
- ALTER DATABASE statement
 - CLOSE clause, 4-17
- ALTER ROLLBACK SEGMENT statement, 3-6
- ALTER SESSION statement
 - SET INSTANCE clause, 3-11
- ALTER SYSTEM ARCHIVE LOG CURRENT
 - statement, 4-17
 - ALTER SYSTEM ARCHIVE LOG statement, 4-17
 - CURRENT clause, 7-7
 - global log switch, 7-7
 - THREAD clause, 4-17
- ALTER SYSTEM CHECK DATAFILES statement
 - instance recovery, 8-4
- ALTER SYSTEM CHECKPOINT LOCAL
 - statement, 4-17
- ALTER SYSTEM CHECKPOINT statement, 7-6
 - global versus local, 4-17
 - specifying an instance, 4-17
- ALTER SYSTEM privilege, 7-7, 7-8
- ALTER SYSTEM SWITCH LOGFILE
 - statement, 4-17, 7-7
 - DBA privilege, 7-7
- applications
 - insert-intensive, 3-9
 - performance, 3-9
 - tuning performance, 3-9
- ARCHIVE LOG command, 4-17
- ARCHIVE LOG statement
 - CURRENT clause, 7-7
 - global log switch, 7-7
- archive logging
 - enabling, 7-4
- archive logs
 - backup, 7-6
 - central NFS for high availability, 6-23
 - configurations, implementation considerations
 - for, 6-25
 - cross shared cross writing with one local and n remote configuration, 6-21
 - cross-mounting for writing configuration, 6-17
 - destinations, shared on UNIX, 6-17

- file format and destination, 7-5
- hybrid configuration, 6-24
- no cross-mounting configuration, 6-16
- one remote configuration, 6-13
- recovering from one node, 7-12
- restoring and recovering from all local nodes with release 1 (9.0.1), 7-13
- restoring and recovering from all local nodes with release 8.1.5, 7-13
- shared read configuration, 6-10
- archived logs
 - configuration, selecting, 6-4
 - non-shared configuration, 6-6
- archiving redo log files
 - forcing a log switch, 7-7
 - identified in control file, 3-8
 - log sequence number, 7-5
- AUTOLOCATE
 - RMAN option, 8-15
- Automatic Undo Management, 2-17, 3-2
 - using, 3-2
- automatic undo management
 - overriding, 3-4
 - using, 3-2
- availability
 - data files, 8-4
 - single-node failure, 8-2
 - steps of recovery, 8-4

B

- background processes
 - SMON, 4-16, 8-2
- background thread trace files, B-1
- BACKGROUND_DUMP_DEST parameter, B-1, B-3
- backing up the database, 7-2
- backups, 7-2
 - and RMAN in Real Application Clusters, 7-8
 - archive log, 7-6
 - database, 7-2
 - files used in recovery, 8-6
 - general information about, 1-3
 - of all files from one node, 7-12
 - of local files from each node, 7-11

- of multiple nodes using RMAN, 7-10
- offline, 7-3
- online, 7-2
- parallel, 7-2
- selecting a method and type, 7-2
- to shared archivelog destinations, 7-11
- using shared directories, 7-10
- block media recovery, 8-7
- blocks
 - associated with instance, 3-11, 8-2
- buffer cache
 - instance recovery, 8-2

C

- cache
 - recovery, 8-5
 - sequence cache, 3-12, 3-13
- CACHE option, CREATE SEQUENCE, 3-12
- central NFS for high availability archive
 - logging, 6-23
- CHECK DATAFILES clause
 - instance recovery, 8-4
- checkpoints, 7-6
 - forcing, 7-6
- client-side parameter files
 - naming conventions, 2-6
 - using, 2-5
- closed backups, 7-3
- Cluster Database
 - Destination Type tasks, 5-21
- Cluster Database Shutdown dialog box, 5-10
- Cluster Database Shutdown Progress dialog box, 5-11
- Cluster Database Started message box, 5-9
- Cluster Database Startup dialog box, 5-8
- Cluster Database Startup Results dialog box, 5-9
- Cluster Database Stopped message box, 5-11
- Cluster databases
 - disconnecting from, 5-10
- cluster databases
 - displaying objects in Console, 5-2
- Cluster Manager (CM)
 - software, 4-10
- CLUSTER_DATABASE parameter, 2-14

- CLUSTER_DATABASE_INSTANCES
 - parameter, 2-14
- CLUSTER_INTERCONNECTS parameter, 2-12, 2-14
- clusterware layer
 - adding a node, 9-3
- comments
 - specifying in server parameter file, 2-5
- committed data
 - checkpoint, 7-6
 - instance failure, 8-2
 - sequence numbers, 3-13
- common parameter files
 - using multiple, 2-10
- complete media recovery, 8-6
- concurrency
 - maximum number of instances, 3-10
 - sequences, 3-13
- configuring
 - archive logs for backup and recovery, 6-26
 - Recovery Manager, 6-26
- CONNECT command, 4-13, 4-14, 4-17
 - forcing a checkpoint, 7-7
- connect strings, 4-14
- CONNECT SYS
 - example of, 2-19
- connecting
 - to instances, 4-13
 - to remote instances, 4-13
- connecting to instances, 4-13
- Console
 - Navigator window with Cluster Database
 - objects, 5-2
 - right-mouse menu, 5-7
 - Edit, 5-14
 - Related Tools, 5-8
 - Results, 5-8, 5-11
 - Shutdown, 5-7, 5-9
 - Startup, 5-7, 5-8
 - View Edit Details, 5-8
 - scheduling jobs, 5-18
 - setting Cluster Database events, 5-24
 - starting Oracle Parallel Server database, 5-8
 - stopping database, 5-9
 - viewing Cluster Database status, 5-14
 - viewing shutdown results, 5-11
 - viewing startup results, 5-11
- Console, starting, 5-2
- contention
 - disk, 3-5
 - rollback segment, 3-5
 - sequence number, 3-12
 - table data, 3-5
- control file
 - autobackup in RMAN, 6-3
- control file snapshots
 - configuring backups of in RMAN, 6-2
- control files
 - backing up, 7-1
 - MAXLOGHISTORY, 3-8
- CONTROL_FILES parameter, 2-12, 8-10
 - same for all instances, 2-12
- CREATE DATABASE statement
 - MAXINSTANCES clause, 3-10
 - MAXLOGHISTORY clause, 3-8
- Create Job property sheet, 5-18 to 5-23
 - General tab, 5-19
 - Parameters tab, 5-21
 - Tasks tab, 5-21
- CREATE SEQUENCE statement, 3-12, 3-13
 - CACHE clause, 3-12, 3-13
 - CYCLE clause, 3-12
 - description, 3-12
 - ORDER clause, 3-12, 3-13
- CREATE TABLE statement
 - FREELIST GROUPS clause, 3-11
 - FREELISTS clause, 3-11
- creating
 - SPFILE, 2-2
- cross, 6-17
- cross shared cross writing with one local and n
 - remote archive log configuration, 6-21
- cross-mounting for writing archive log
 - configuration, 6-17
- CURRENT clause
 - checkpoints, 7-7
 - forcing a global log switch, 7-7
- CYCLE option, CREATE SEQUENCE, 3-12

D

- data dictionary
 - sequence cache, 3-13
- data files
 - access for instance recovery, 8-4
 - adding, 3-2
 - backing up, 7-1
 - instance recovery, 8-4
 - parallel recovery, 8-6
- database
 - backups, 7-2
 - number of archived log files, 3-8
 - number of instances, 3-10
 - quiescing, 4-16
 - starting NOMOUNT, 8-10
- Database Configuration Assistant
 - adding instances, 9-10
 - and log files, 3-8
 - deleting instances, 9-19
 - placement of IFILE parameter, 2-10
- DB_BLOCK_SIZE parameter, 2-12
 - same for all instances, 2-12
- DB_DOMAIN parameter, 2-12
- DB_FILES parameter, 2-12
 - same for all instances, 2-12
- DB_NAME parameter, 2-7, 2-12, 2-14
 - same for all instances, 2-12
- DBA_ROLLBACK_SEGS view, 3-5
- DBCA
 - adding instances, 9-10
 - deleting instances, 9-19
- disaster recovery, 8-9
- DISCONNECT command, 4-14
- disconnecting from instances, 4-14
 - multiple sessions, 4-16
 - user process, 4-15
- disk
 - contention, 3-5
 - rollback segments, 3-5
- DISPATCHERS parameter, 2-15
 - for the shared server, 2-15
- distributed resources
 - sequence, 3-12
- DML_LOCKS parameter, 2-12, 2-15

- dropping a redo log file
 - log switch, 7-7

E

- Edit Parallel Server dialog box, 5-14
- error messages
 - ORA-9291, 6-26
 - parameter values, 2-19
- errors
 - call trace stack, B-3
 - ORA-600, B-4
- exclusive mode
 - specifying thread number, 2-19
- export
 - raw device configuration information, 4-8
- Export utility
 - and free lists, 3-9
- extents
 - rollback segment, 3-5
 - size, 3-5

F

- failure
 - access to files, 8-4
 - instance, 8-2
 - instance recovery, 8-4
 - media, 8-6
 - node, 8-2
- failures
 - multiple node, 8-3
- Fast-start checkpointing, 8-3
- Fast-start fault recovery, 8-3
- features, new, xxviii
- file management, 3-2
- files
 - ALERT, 8-4
 - archiving redo log, 7-5
 - control file, 3-8
 - dropping, 7-7
 - redo log, 7-5
 - renaming, 7-7
 - restricted operations, 7-7
 - used in recovery, 8-6

- FORCE mode, warning notice, 5-8
- foreground processes
 - instance shutdown, 4-15
- free list groups
 - assigned to instance, 3-10
- free lists, 3-11
 - and Export utility, 3-9
- free space
 - managing with SQL, 3-11
- FREELIST GROUPS
 - storage option, instance number, 3-10
- FREELIST GROUPS option, 3-10

G

- GCS
 - recovery steps, 8-4
- General tab, from Create Job property sheet, 5-19
- Global Cache Service resource
 - sequence, 3-12
- GLOBAL clause
 - forcing a checkpoint, 4-17, 7-7
- global constant parameters
 - same for all instances, 2-12
- Global Services Daemon (GSD), 4-9, 9-10
- Globalization Support parameters, 2-16
- groups
 - redo log files, 3-7
 - unique numbers, 3-8
 - VSLOGFILE, 3-8
- GSD
 - as a background process for manageability
 - tools, 4-9
 - UNIX implementation of, 4-9
 - Windows implementation of, 4-9
- gsdservice -install
 - create services, 4-9
- gsdservice -remove
 - to stop and delete services, 4-9
- gsdservice -start
 - to start services, 4-9

H

- history

- archive, 8-8
- HOST command, 4-17
- hybrid archive logging, 6-24

I

- IFILE
 - parameter placement in instance-specific parameter files, 2-10
- IFILE parameter
 - multiple files, 2-10
 - overriding values, 2-10
 - specifying identical parameters, 2-18
- import
 - raw device configuration information, 4-8
- incomplete media recovery, 8-7
- initdb_name.ora file
 - BACKGROUND_DUMP_DEST parameter, B-1, B-3
 - DB_NAME parameter, 2-7
 - USER_DUMP_DEST parameter, B-2
- initdb_name.ora file
 - described, 2-8
- INITIAL storage parameter
 - rollback segments, 3-5
- initialization parameter files
 - definition, 2-6
 - for instances, 2-1
 - initdb_name.ora, 2-8
 - initsid.ora, 2-7
- initialization parameters
 - duplicate values, 2-10
 - identical for all instances, 2-12
 - multiple instance issues regarding, 2-14
 - setting for multiple instances, 2-12
 - that must be identical on all instances, 2-12
 - that must be unique on all instances, 2-13
- init.ora file
 - defined, 2-7
- initsid.ora file
 - described, 2-7
- initsid.ora file
 - described, 2-7
- inserts
 - performance, 3-9

INSTANCE clause
 SHOW INSTANCE command, 4-14

INSTANCE_NAME parameter
 unique values for instances, 2-15

INSTANCE_NUMBER
 setting, 2-19

INSTANCE_NUMBER parameter, 2-13, 2-19
 and SQL options, 3-11
 exclusive or shared mode, 2-19
 recommended settings for, 2-19
 unique values for instances, 2-19

instances
 adding at the Oracle layer, 9-10
 associated with data block, 3-11
 associating with free list groups, 3-9
 associating with free lists, 3-10
 connecting to, 4-13
 current, 4-14, 7-7
 deleting with the DBCA, 9-19
 failure, 8-3
 failures, recovery from, 8-2
 initialization parameter files, 2-1
 initsid.ora file, 2-7
 instance number, 3-10
 maximum number, 3-5, 3-10
 recovery, 4-16, 8-2
 recovery, abnormal shutdown, 4-16
 recovery, access to files, 8-4
 recovery, data file access, 8-4
 recovery, global checkpoint, 7-7
 recovery, multiple failures, 8-3
 remote, 2-18, 4-14
 rollback segment required, 3-5
 Server Management, 5-2
 setting, 4-13
 shutting down, 4-15
 sid, 2-7
 startup and shutdown results, 5-11, 5-15
 startup order, 2-19
 thread number, 2-19
 verifying, 4-15

Introduction, 1-1

J

jobs
 choosing tasks for, 5-21
 creating for cluster database, 5-18
 setting up rights for running, 5-22
 specifying information for, 5-19
 specifying parameters for server management tasks, 5-21

L

LISTENER
 parameter for the shared server, 2-15

LOCAL clause
 forcing a checkpoint, 4-17, 7-7

local instances
 nodes, 4-14

log files
 redo log file, 7-1

log history, 8-8

log sequence numbers, 7-5

log switches, 3-8, 7-6
 forcing, 7-7, 7-8
 forcing on a closed thread, 7-8
 forcing on closed threads, 7-8

LOG_ARCHIVE_DEST parameter, 2-12, 6-8, 8-9
 specifying for recovery, 6-8, 8-9

LOG_ARCHIVE_DEST_n parameter, 7-4

LOG_ARCHIVE_FORMAT parameter, 2-16, 6-8, 7-4, 7-5, 8-9
 same for all instances, 6-8, 8-9
 used in recovery, 6-8, 8-9

LOG_ARCHIVE_START parameter
 automatic archiving, 2-10

LOG_CHECKPOINT_TIMEOUT parameter
 inactive instance, 7-7

LSNRCTL utility
 START command, 4-10, 4-11

M

Massively Parallel Processing System, 3-10

MAX_COMMIT_PROPAGATION_DELAY
 parameter, 2-12, 2-16

MAXEXTENTS storage parameter

- automatic allocations, 3-12
- MAXINSTANCES clause, 3-10
- MAXINSTANCES option, 3-10
- MAXINSTANCES parameter, 3-11
 - assigning free lists to instances, 3-10
- MAXLOGHISTORY option, 3-8
- media failures, 8-6
 - recovery from, 8-5
- media recovery
 - block media recovery, 8-7
 - complete, 8-6
 - incomplete, 8-7
 - incomplete, O/S utilities, 8-7
 - log history, 8-8
- memory
 - SGA, 3-13
- messages
 - access to files, 8-4
 - ALERT file, 8-4
 - instance shutdown, 4-15
- migrating
 - Oracle8i configurations to Oracle9i, 4-8
- MINEXTENTS storage parameter
 - automatic allocations, 3-12
- modified data
 - instance recovery, 8-2
- modulo, 3-10
- multiple node failures, 8-3
- multiple nodes
 - starting from one node, 2-18
- multiplexed redo log files, 3-7
 - example, 3-8

N

- naming conventions
 - for client-side parameter files, 2-6
- Navigator window
 - Cluster Database objects, 5-2
 - Parallel Server Instances folder, 5-5
 - right-mouse menu, 5-7, 5-11
 - Edit, 5-14
 - Related Tools, 5-8
 - Results, 5-8
 - Shutdown, 5-7, 5-9

- Startup, 5-7, 5-8
 - View Edit Details, 5-8
 - setting Cluster Database events, 5-24
 - starting Oracle Parallel Server database, 5-8
 - stopping database, 5-9
 - viewing Cluster Database status, 5-14
 - viewing shutdown results, 5-11
 - viewing startup results, 5-11
- new features, xxviii
- NEXT storage parameter, 3-5
- NFS
 - for high availability, 6-25
 - on UNIX, using, 6-25
- no cross-mounting archive log configuration, 6-16
- nodes
 - adding at the clusterware layer, 9-3
 - adding at the Oracle layer, 9-9
 - adding to cluster, 9-2
 - affinity awareness, 7-8
 - backing up multiple with RMAN using several channels, 7-10
 - failure of, 8-2
 - local, 2-18
 - parallel backup, 7-2
 - remote, 4-14
- NOMOUNT option, 8-10
- non-shared archive log destinations
 - and backups from each node, 7-11
- non-shared archive logging
 - restore and recovery, 6-9
- non-shared archived log configuration, 6-6
- NOORDER option, CREATE SEQUENCE, 3-13
- number generator, 3-12

O

- offline backups
 - parallel, 7-2
 - redo log files, 7-3
- offline tablespaces
 - restrictions, 3-5
- one remote archive log configuration, 6-13
- online backups, 7-4
 - block media recovery, 8-7
 - parallel, 7-2

- redo log files, 7-2
- online recovery, 8-2, 8-4
- online redo log files
 - archiving, 7-1, 7-2
 - log switch, 7-7
 - thread of redo, 2-19
- open backups, 7-2
- operating system
 - privileges, 4-18
- operating system utilities
 - for recovery, 8-8
- operating system-specific Oracle documentation
 - archived redo log name, 7-5
- ORA-600, B-4
- ORA-9291 message, 6-26
- Oracle Database Configuration Assistant
 - troubleshooting, 9-11
- Oracle Enterprise Manager
 - Events
 - Cluster Database events, 5-24
 - performing administration with, 1-3
 - scheduling jobs, 5-18
 - starting
 - Oracle Parallel Server database, 5-8
 - stopping database, 5-9
 - viewing
 - Cluster Database status, 5-14
 - shutdown results, 5-11
 - startup results, 5-11
- Oracle layer
 - adding a node, 9-9
- Oracle Parallel Server databases
 - mounting, 5-8
 - shutdown, 5-11
- Oracle Real Application Clusters
 - creating a job on, 5-18
- Oracle8i
 - migrating to Oracle9i, 4-8
- Oracle9i
 - migrating from Oracle8i, 4-8
- Oracle9i Real Application Clusters
 - overview, 1-1
- oraxxxx.trc file, B-2
- ORDER option, 3-12, 3-13
- Output tab

- from Cluster Database Operation Results dialog box, 5-13

P

- parallel backups, 7-2
- parallel mode
 - sequence restrictions, 3-13
- parallel recovery, 8-6, 8-12, 8-13
- Parallel Server Instances folder
 - in Console Navigator window, 5-5
- Parallel Server Operation Results dialog box, 5-11
- Parallel Server Startup/Shutdown Results dialog box, 5-11
- PARALLEL_MAX_SERVERS parameter, 8-12, 8-13
- Parameter, 2-1
- parameter file
 - server file, backing up, 2-2
 - server file, exporting, 2-4
 - server file, setting parameters in, 2-2
 - server file, using, 2-2
- parameter files
 - backing up, 7-1
 - client-side, using, 2-5
 - common file, 2-18
 - duplicate values, 2-10
 - instance-specific, 2-11
 - location, 2-11
 - PFILE, 2-18
 - remote instance, 2-18
 - remote instances, 4-14
- parameters
 - database creation, 3-10
 - initialization, 2-1, 3-1
 - instance-specific settings in server parameter file, 2-4
 - management of, 1-2
 - setting for multiple instances, 2-12
 - storage, 3-5
 - that must be identical on all instances, 2-12
 - types of, 2-12
- Parameters tab
 - from Create Job property sheet, 5-21
- partitioning data
 - rollback segments, 3-5

- performance
 - application, 3-9
 - caching sequences, 3-13
 - inserts and updates, 3-9
 - rollback segments, 3-5
 - sequence numbers, 3-13
- private rollback segments, 2-13, 3-5
 - specifying, 3-6
- privileges
 - ALTER SYSTEM, 7-7, 7-8
- PROCESSES parameter, 2-17
- PROTOCOL
 - parameter for the shared server, 2-15
- public rollback segments, 3-5

Q

- quiesce database
 - in Real Application Clusters, 4-16

R

- raw device configuration information
 - export, 4-8
 - import, 4-8
- Real Application Clusters
 - initialization parameter files, 2-7
- Real Application Clusters databases
 - starting, 4-10
- RECOVER command, 4-17, 8-13
- recovery, 8-1
 - access to files, 8-4
 - after SHUTDOWN ABORT, 4-16
 - block media recovery, 8-7
 - disaster, 8-9
 - from an offline backup, 8-9
 - from an online backup, 8-9
 - from multiple node failure, 8-3
 - from single-node failure, 8-2
 - general information about, 1-3
 - global checkpoint, 7-7
 - incomplete media, 8-7
 - instance, 4-16, 8-2
 - log history, 8-8
 - media failure, 7-7, 8-6

- media failures, 8-5
- online, 8-2
- parallel, 8-12, 8-13
- PARALLEL_MAX_SERVERS parameter, 8-12, 8-13
- recovery time, 7-7
- setting parallelism, 8-12, 8-13
- steps of, 8-4
- using operating system utilities, 8-8
- using redo log, 7-2

- Recovery Manager
 - archive log backup, 7-6
 - configure archive logs, 6-26
 - disaster recovery, 8-9
 - incomplete media recovery, 8-7
 - media failures, 8-6
- RECOVERY_PARALLELISM parameter, 2-17, 8-12, 8-13
- redo log files
 - archiving, 7-1, 7-2, 7-7
 - backup, 7-2, 7-3
 - dropping, 7-7
 - identified in control file, 3-8
 - instance recovery, 8-2
 - log sequence number, 7-5
 - renaming, 7-7
 - using, 3-7
- redo log groups, 3-7
- redo logs
 - format and destination specifications, 7-5
- Registering, 5-24
- remote instances, 2-18, 4-14
- renaming a file
 - log switch, 7-7
- resources
 - associating with free list groups, 3-9
 - releasing, 8-2
- restrictions
 - cached sequence, 3-13
 - offline tablespace, 3-5
- RETRY option
 - STARTUP PARALLEL command, 4-12
- RMAN
 - archived log configuration schemes, 6-5
 - AUTOLOCATE option, 8-15

- backups in Real Application Clusters, 7-8
- in Real Application Clusters, overview of
 - using, 6-2
 - restoring archive logs with, 8-14
- rollback segment
 - SYSTEM, 3-4
- rollback segments
 - contention, 3-5
 - multiple, 3-5
 - online, 3-5
 - private, 3-5
 - public, 3-5
 - public versus private, 3-5
 - tablespace, 3-5
- ROLLBACK_SEGMENTS parameter, 2-13, 2-17, 3-6
- rolling back
 - instance recovery, 8-2
- ROW_LOCKING parameter, 2-12, 2-13

S

- scaling
 - adding instances, 9-10
 - adding nodes, 9-3
 - adding nodes and instances, 9-1
 - general information about, 1-3
- Select Instances to Start dialog box, 5-9
- Select Instances to Stop dialog box, 5-11
- sequence number generator, 3-12
 - distributed resources, 3-12
 - in Real Application Cluster databases, 3-12
 - restriction, 3-13
 - skipping sequence numbers, 3-13
- sequence numbers
 - CACHE option, 3-12
 - ORDER option, 3-13
- sequences
 - data dictionary cache, 3-12, 3-13
 - log sequence number, 7-5
 - not cached, 3-13
 - timestamp, 3-13
- SERIALIZABLE parameter, 2-12
- Server Control
 - using, 4-2
- Server Management
 - administration of instances, 5-2
- server parameter file
 - backing up, 2-2
 - backups and restores of, 8-10
 - exporting, 2-4
 - instance specific settings, 2-4
 - setting values in, 2-2
 - using, 2-2
- SERVICE_NAMES parameter, 2-12
- sessions
 - multiple, 4-14, 4-16
- SESSIONS_PER_USER parameters, 2-17
- SET INSTANCE command, 2-18, 4-13
 - example of, 2-18
 - instance startup, 4-14
- setting instances, 4-13
- shared archivelog destinations
 - backups to, 7-11
- shared directories
 - and backups, 7-10
- shared drives
 - using, on Windows platforms, 6-26
- shared mode
 - instance number, 2-19
 - instance recovery, 8-2
- shared read archive log configuration, 6-10
- shared server
 - parameters for, 2-15
- SHOW INSTANCE command, 4-14, 4-17
- SHOW PARAMETER command, 4-17
- SHOW PARAMETERS command, 4-17
 - example of, 2-19
- SHOW SGA command, 4-17
- SHUTDOWN ABORT command, 4-16
- SHUTDOWN command, 4-17
 - ABORT option, 4-16
 - IMMEDIATE option, 4-16
 - specifying an instance, 4-14
- SHUTDOWN TRANSACTIONAL, 4-16
- shutdown types, 5-10
 - abort, 5-10
 - immediate, 5-10
 - normal, 5-10
 - shutdown database and other services, 5-10

- shut down database only, 5-10
- shutting down
 - instances, 4-15
- shutting down a cluster database
 - setting parameters, 5-23
- shutting down a database
 - viewing results, 5-11
- shutting down an instance
 - abnormal shutdown, 4-16
 - archiving redo log files, 7-7
 - forcing a log switch, 7-7
 - lost sequence numbers, 3-13
- shutting down an Oracle Parallel Server
 - database, 5-11
- shutting down instances, 4-15
- sidart.log file, B-2
- sidbsp0.trc file, B-2
- siddbwr.trc file, B-2
- sidlckn.trc file, B-2, B-4
- sidlmdn.trc file, B-2
- sidlmon.trc file, B-2
- sidp00n.trc file, B-2
- sidsmon.trc file, B-2
- SMON process
 - instance recovery, 8-2, 8-3
 - recovery after SHUTDOWN ABORT, 4-16
- snapshot control file, 6-3
- SPFILE
 - creating, 2-2
- SPFILE parameter, 2-17
- SQL statements
 - instance-specific, 4-17
- SQL*Plus sessions
 - multiple, 4-16
- SRVCONFIG
 - importing and exporting raw device
 - configuration with, 4-8
- SRVCTL
 - using, 4-2
- srvctl add db command, 4-6
- srvctl add instance command, 4-6
- srvctl config command, 4-5
- srvctl delete db command, 4-6
- srvctl delete instance command, 4-6
- srvctl get env command, 4-5
- srvctl move instance command, 4-7
- srvctl rename instance command, 4-7
- srvctl set env command, 4-7
- srvctl start command, 4-3
- srvctl status command, 4-4
- srvctl stop command, 4-4
- srvctl unset env command, 4-7
- SRVCTL utility
 - commands that update the configuration, 4-5
 - using to administer instance configurations, 4-2
- start services
 - with gsdservice, 4-9
- starting
 - listener, 4-10, 4-11
 - Oracle Parallel Server database, 5-8
 - Real Application Clusters database, 4-10
- Starting OEM Console, 5-2
- starting up
 - remote instance, 4-14
 - rollback segments, 3-5
- startup
 - and parameters, 2-18
 - on remote nodes, 2-18
 - remote instance, 2-18
 - results, 5-11
 - setting cluster database parameters for, 5-21
 - startup order, 2-19
- STARTUP command, 2-18, 4-17
 - specifying an instance, 4-14
- startup types, 5-8
 - force, 5-8
 - mount, 5-8
 - no mount, 5-8
 - open, 5-8
 - restrict, 5-9
- Status Details tab, 5-12
 - from Cluster Database Operation Results dialog
 - box, 5-12
 - from Edit Cluster Database dialog box, 5-16
- Status tab
 - from Edit Cluster Database dialog box, 5-15
- stop and delete services
 - with gsdservice, 4-9
- stopping
 - database, 5-9

- storage
 - administering, 1-2
- storage options
 - extent size, 3-5
 - rollback segment, 3-5
- SYSDBA
 - privilege for connecting, 4-14
- SYSDBA privilege, 5-22
- SYSOPER privilege, 5-22
 - for connecting, 4-14
- system change numbers
 - archive file format, 7-5
- System Global Area (SGA)
 - sequence cache, 3-12
- SYSTEM rollback segment, 3-4

T

- tables
 - allocating extents, 3-11
 - contention, 3-5
 - tablespace, 3-5
- tablespaces
 - active rollback segments, 3-5
 - offline, 3-5
 - parallel backup, 7-2
 - parallel recovery, 8-6
 - rollback segment, 3-5
 - switching for undo, 3-3
 - tables, 3-5
 - taking offline, 3-5
- Tasks tab
 - from Create Job property sheet, 5-21
- THREAD clause, 4-17, 7-7
- THREAD parameter, 2-13, 2-17
- THREAD statement
 - for backing up archivelogs, 7-10
- threads
 - archive file format, 7-5
 - archiving redo log files, 7-7
 - enabled, 8-8
 - example, 3-7
 - exclusive mode, 2-19
 - forced log switch, 7-7
 - number of groups, 3-7

- threads of redo, 3-7
- trace files, B-1
 - background thread trace files, B-1
 - error call trace stack, B-3
 - managing, 3-9
 - oraxxxx.trc, B-2
 - sidalrt.log, B-2
 - sidbsp0.trc, B-2
 - siddbwr.trc file, B-2
 - sidlckn.trc, B-2, B-4
 - sidlmdn.trc, B-2
 - sidlmon.trc, B-2
 - sidp00n.trc, B-2
 - sidsmon.trc file, B-2
- transactions
 - committed data, 7-6
 - instance failure, 8-2
 - rolling back, 8-2
 - sequence numbers, 3-12
 - waiting for recovery, 8-2
- troubleshooting, B-1 to B-4
 - general information about, 1-4
 - Oracle Database Configuration Assistant, 9-11
 - trace files, B-1

U

- undo tablespace
 - switching, 3-3
- UNDO_MANAGEMENT parameter, 3-2, 3-4
- UNDO_TABLESPACE parameter, 3-3
- updates
 - performance, 3-9
- user processes
 - associating with free lists, 3-11
 - free lists, 3-11
 - instance shutdown errors, 4-15
- user trace files, B-2
- USER_DUMP_DEST parameter, B-2
- users
 - associating with free list groups, 3-9
- utilities
 - operating system, for recovery, 8-8

V

V\$ACTIVE_INSTANCES, 4-15
V\$ACTIVE_INSTANCES table, 5-15
V\$FAST_START_SERVERS
 view, 8-13
V\$FAST_START_TRANSACTIONS
 view, 8-13
V\$LOGFILE view, 3-8

