Oracle8*i*

Documentation Addendum

Release 3 (8.1.7)

September 2000

Part No. A85455-01

ORACLE®

Oracle8*i* Documentation Addendum, Release 3 (8.1.7)

Part No. A85455-01

Copyright © 2000, Oracle Corporation. All rights reserved.

Authors: Lance Ashdown, Mark Bauer, Ruth Baylis, Michele Cyran, Joyce Fee, Paul Lane, Anna Logan, Lenore Luscher, Kevin MacDowell, Colin McGregor, Debbie Steiner, Randy Urbano

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 of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

1 Introduction

Summary of New Features 1-2
How This Addendum Is Organized 1-6
Other Books Being Released for 8.1.7 1-7

2 interMedia Text

Overview of New and Updated Features 2-3
Catalog (CTXCAT) Index 2-3
Keyless Index 2-5
Multi-Column Datastore 2-6
URL Datastore Username and Password Support 2-6
File and URL Datastore Security 2-6
Procedure Filter 2-7
Multi-Language Stoplist 2-7
Prefix Indexing 2-7
Single Token Index Optimization 2-8
Document Tokens Service 2-8
Thesaurus Translation Calls 2-8
New Fuzzy Operator Functionality 2-9
Inso Filter Enhancements 2-9
Supplied French Knowledge Base 2-10
User-Defined Knowledge Bases 2-10
CATSEARCH New SQL Operator 2-13
CREATE INDEX Updated Syntax 2-16

ALTER INDEX Updated Syntax 2-20
MULTI_COLUMN_DATASTORE New Datastore Type 2-24
PROCEDURE_FILTER New Filter Type 2-28
Procedure Filter Preference Example 2-32
Prefix Indexing New Feature 2-33
Enabling Prefix Indexing 2-35
Fuzzy Operator New Syntax 2-36
CTX_DDL.ADD_INDEX New Procedure 2-38
CTX_DDL.CREATE_INDEX_SET New Procedure 2-41
CTX_DDL.DROP_INDEX_SET New Procedure 2-42
CTX_DDL.REMOVE_INDEX New Procedure 2-43
CTX_DDL.OPTIMIZE_INDEX Updated Syntax 2-44
CTX_DDL.CREATE_STOPLIST Updated Syntax 2-46
CTX_DDL.ADD_STOPWORD Updated Syntax 2-47
CTX_THES.CREATE_TRANSLATION New Procedure 2-49
CTX_THES.DROP_TRANSLATION New Procedure 2-50
CTX_THES.HAS_RELATION New Procedure 2-51
CTX_THES.UPDATE_TRANSLATION New Procedure 2-52
CTX_DOC.TOKENS New Procedure 2-53
CTX_DOC.THEMES Updated Syntax 2-56
CTX_DOC.GIST Updated Syntax 2-59

3 New Character Set Scanner Utility

Overview of Choosing and Migrating Character Sets 3-2
Data Truncation 3-2
Character Set Conversions 3-4
Database Character Set Migration 3-6
Data Scanning 3-7
Conversion of Data 3-7
What is the Character Set Scanner Utility? 3-9
Conversion Tests on Character Data 3-9
Access Privileges 3-10
Restrictions 3-10
Database Containing Data from Two or More Character Sets 3-10
Database Containing Data not from the Database Character Set 3-11

Scan Modes in the Scanner 3-11
Full Database Scan 3-11
User Tables Scan 3-11
Single Table Scan 3-11
Using The Scanner 3-12
Before Using the Scanner 3-12
Compatibility 3-13
Invoking the Scanner 3-13
Getting Online Help 3-14
The Parameter File 3-15
Scanner Parameters 3-16
ARRAY 3-16
BOUNDARIES 3-16
CAPTURE 3-17
FEEDBACK 3-17
FROMCHAR 3-17
FROMNCHAR 3-18
FULL 3-18
HELP 3-18
LASTRPT 3-18
LOG 3-19
MAXBLOCKS 3-19
PARFILE 3-20
PROCESS 3-20
SUPPRESS 3-20
TABLE 3-21
TOCHAR 3-21
TONCHAR 3-21
USER 3-21
USERID 3-22
Sample Scanner Sessions 3-22
Sample Session of Full Database Scan 3-22
Sample Session of User Tables Scan 3-23
Sample Session of Single Table Scan 3-24
Scanner Reports 3-25

Database Scan Summary Report 3-26
Individual Exception Report 3-32
Storage and Performance Considerations in the Scanner 3-34
Storage Considerations 3-34
Performance Consideration 3-35
Reference Material 3-36
Scanner Tables 3-36
Scanner Messages 3-38
Subsets and Supersets 3-45

4 Oracle Parallel Server

Raw Partition Tablespace Size Requirements 4-2
Connecting to Secondary Instances 4-2
Recovery Manager Procedures for Oracle Parallel Server 4-2
New Parameter OPS_INTERCONNECTS for Solaris 4-2

5 Net8

Configuring Directory Server Access for Oracle8*i* Feature Integration 5-2
Configuring Directory Access During Installation 5-2
Configuring Directory Access After Installation 5-5
Adding Users to and Removing Users from the OracleNetAdmins Group 5-8
Configuring Session Data Unit for Net8 Performance 5-9
SDU Configuration on Clients 5-9
SDU Configuration on the Server 5-10
INSTANCE_ROLE Parameter for Primary and Secondary Instance Configurations 5-11
Oracle Names Control Utility Command Changes 5-14
DOMAIN_HINT Command Not Supported 5-14
REGISTER_NS and UNREGISTER_NS Commands for Oracle Names Server Registration 5-17

6 PL/SQL Supplied Packages

Package Overview 6-2
Using Oracle Supplied Packages 6-2
Creating New Packages 6-3

Referencing Package Contents 6-5
DBMS_LDAP 6-5
Summary of Subprograms 6-6
Exception Summary 6-8
Data-Type Summary 6-10
FUNCTION init 6-11
FUNCTION simple_bind_s 6-14
FUNCTION bind_s 6-16
FUNCTION unbind_s 6-18
FUNCTION compare_s 6-19
FUNCTION search_s 6-21
FUNCTION search_st 6-24
FUNCTION first_entry 6-27
FUNCTION next_entry 6-29
FUNCTION count_entries 6-31
FUNCTION first_attribute 6-33
FUNCTION next_attribute 6-35
FUNCTION get_dn 6-37
FUNCTION get_values 6-39
FUNCTION get_values_len 6-41
FUNCTION delete_s 6-43
FUNCTION modrdn2_s 6-44
FUNCTION err2string 6-47
FUNCTION create_mod_array 6-48
PROCEDURE populate_mod_array (String Version) 6-49
PROCEDURE populate_mod_array (Binary Version) 6-51
FUNCTION modify_s 6-53
FUNCTION add_s 6-55
PROCEDURE free_mod_array 6-57
FUNCTION count_values 6-58
FUNCTION count_values_len 6-59
FUNCTION rename_s 6-60
FUNCTION explode_dn 6-63
FUNCTION open_ssl 6-64
DBMS_OBFUSCATION_TOOLKIT 6-66

Overview of Key Management 6-66
DESEncrypt Procedure 6-69
Parameter Descriptions 6-69
Encryption Procedure Restriction 6-70
DESDecrypt Procedure 6-70
Parameter Descriptions 6-70
DESDecryption Procedure Restriction 6-71
DES Encryption Code Example 6-71
DES3Encrypt Procedure 6-73
Parameter Descriptions 6-73
Encryption Procedure Restriction 6-74
DES3Decrypt Procedure 6-74
Parameter Descriptions 6-75
DES3Decrypt Procedure Restriction 6-75
DES3 Encryption Code Example 6-76

7 Recovery Manager

Preparing a Standby Database Using RMAN 7-2
About Standby Database Preparation Using RMAN 7-2
Creating the Standby Control File Using RMAN 7-3
Making Image Copies of Standby Control Files 7-6
Naming the Standby Database Datafiles When Using RMAN 7-6
Naming the Standby Database Online Redo Logs When Using RMAN 7-8
Creating a Standby Database Using RMAN 7-9
About Standby Creation Using RMAN 7-9
Restrictions When Creating a Standby Database Using RMAN 7-12
Starting RMAN and the Standby Instance 7-12
Creating a Standby Database on a Remote Host with the Same Directory Structure 7-13
Creating a Standby Database on a Remote Host with a Different Directory Structure 7-15
Creating a Standby Database on the Local Host 7-20
Using Image Copies to Create a Standby Database 7-20
Backing Up Files at the Standby Site Using RMAN 7-25
About RMAN Backups of Standby Database Datafiles and Archived Logs 7-25
Restrictions When Making RMAN Standby Database Backups 7-26
Interpreting the RC_ARCHIVED_LOG View 7-26

Determining When to Back Up Standby Database Archived Redo Logs 7-29
Backing Up a Standby Database Using RMAN 7-31
Performing a Test Backup Using RMAN 7-33
Setting the Default Location of the RMAN Snapshot Control File 7-33
Crosschecking and Deleting on Multiple RMAN Channels 7-34
About Allocating Multiple RMAN Channels for Maintenance Commands 7-34
How RMAN Crosschecks and Deletes on Multiple Channels 7-35
Crosschecking Simultaneously on Disk and Tape Channels: Example 7-36
Crosschecking on Multiple OPS Nodes: Example 7-36
Deleting Simultaneously on Disk and Tape Channels: Example 7-36
Releasing Multiple Channels with One Command: Example 7-36
Connecting RMAN to a Target Database in an OPS Cluster 7-36
Selected RMAN Maintenance Commands No Longer Require a Catalog 7-37
Troubleshooting RMAN Character Set Errors: Scenario 7-38
Diagnosis of Cause 7-38
Solution 7-38
Setting Recovery Catalog Compatibility Level No Longer Required 7-38
Recovery Catalog Compatibility Changes 7-39
About RMAN Compatibility 7-39
RMAN Compatibility Matrix 7-39
RMAN Compatibility: Scenario 7-41
RMAN Syntax Diagram Changes 7-41
BACKUP 7-42
COPY 7-45
DEBUG 7-46
DUPLICATE 7-47
SET 7-49
Recovery Catalog View Changes 7-50
Backup and Recovery Dynamic Performance View Changes 7-51

8 Reference

ORACLE_TRACE_ENABLE 8-2
V$ARCHIVED_LOG 8-2
V$BACKUP_DATAFILE 8-3
V$BACKUP_SET 8-4

V$DATAFILE_COPY 8-5
V$PROXY_DATAFILE 8-7
V$SQL_SHARED_CURSOR 8-8

9 Replication

New Replication Features 9-2
Reduced Quiesce for Single Master Replication Environments 9-2
Documentation Additions 9-5
UTL_FILE_DIR Initialization Parameter and Generated Replication Files 9-5
Procedural Replication and Snapshot Sites 9-5
Avoiding Problems When Adding a New Snapshot Site 9-6
Documentation Updates 9-7
Replication of Sequences Is Not Supported 9-7
Updated "Base Table" Section 9-7
Guidelines for Scheduled Links 9-9
Guidelines for Scheduled Purges of a Deferred Transaction Queue 9-12
Serial and Parallel Propagation 9-14
The Replication Management Tool in DBA Studio 9-15
Usage Scenarios for the Replication Management Tool 9-16
Logging in to the Replication Management Tool 9-17
The Replication Management Tool Interface 9-18
The Replication Management Tool Wizards 9-29
Flowchart for Creating a Replicated Environment 9-40

10 Replication Management API Reference

Changes to the DBMS_REPCAT_INSTANTIATE Package 10-2
The offline_dirpath Parameter Added to INSTANTIATE_OFFLINE_REPAPI 10-2
Changes to the DBMS_REPCAT_RGT Package 10-6
The offline_dirpath Parameter Required in INSTANTIATE_OFFLINE_REPAPI 10-6
Documentation Additions 10-10
Switching Master Sites that are Running Different Releases of Oracle 10-10

11 Oracle Label Security Error Messages

12 Standby Database

Revised Script to Identify the Logs in a Gap Sequence 12-2
Additions to Compatibility and Operational Requirements 12-3
Compatibility and Operational Requirements 12-3
Changes to Enabling Online Changes to the Initialization Parameter Settings 12-4
Changes to Receiving Archived Redo Logs While in Read-Only Mode 12-5
Changes to Creating the Standby Database Files 12-5
Creating the Standby Database Files 12-5
Revised Scenario 1: Creating a Standby Database on the Same Host 12-9
Scenario 1: Creating a Standby Database on the Same Host 12-9
Revised Scenario 2: Creating a Standby Database on a Remote Host 12-21
Scenario 2: Creating a Standby Database on a Remote Host 12-21

13 Using Oracle Trace

Overview of Oracle Trace 13-2
Using Oracle Trace Data 13-2
Using Oracle Trace Manager 13-4
Managing Collections 13-4
Collecting Event Data 13-5
Accessing Collected Data 13-5
Using Oracle Trace Data Viewer 13-6
Using Oracle Trace Predefined Data Views 13-7
Viewing Oracle Trace Data 13-13
Getting More Information on a Selected Query in Oracle Trace 13-17
Manually Collecting Oracle Trace Data 13-20
Using the Oracle Trace Command-Line Interface 13-20
Using Initialization Parameters to Control Oracle Trace 13-24
Controlling Oracle Trace Collections from PL/SQL 13-28
Accessing Oracle Trace Collection Results 13-31
Formatting Oracle Trace Data to Oracle Tables 13-31
Running the Oracle Trace Statistics Reporting Utility 13-32

Index

Send Us Your Comments

***Oracle8i Documentation Addendum*, Release 3 (8.1.7)**

Part No. A85455-01

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) 506-7228 Attn: ST Oracle8*i* Documentation Manager
- Postal service:
  Oracle Corporation
  ST/Oracle8*i* Documentation Manager
  500 Oracle Parkway 4OP12
  Redwood Shores, CA 94065
  USA

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

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

Preface

This book contains supplemental information on various areas of Oracle functionality that have been updated in release 8.1.7.

This preface contains this topic:

- Conventions

Conventions

This section describes the conventions used in the text and code examples of the Oracle8*i* 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.	The C datatypes such as **ub4**, **sword**, or **OCINumber** are valid. When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles, syntax clauses, or placeholders.	*Oracle8i Concepts* You can specify the *parallel_clause*. Run U*old_release*.SQL where *old_release* refers to the release you installed prior to upgrading.
UPPERCASE monospace (fixed-width font)	Uppercase monospace typeface indicates elements supplied by the system. Such elements include executables, parameters, privileges, datatypes, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, user names, and roles.	You can specify this clause only for a `NUMBER` column. You can change this value in an `ALTER TABLE` statement. These are grouped by the `DEPTNO` column. Specify the `ROLLBACK_SEGMENTS` parameter. Use the `DBMS_STATS.GENERATE_STATS` procedure.
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates 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, user names and roles, program units, and parameter values.	The `deptno`, `dname`, and `loc` columns are in the `scott.dept` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect to the `sales@sf.acme.com` database. Connect as `oe` user.

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (digits [ , precision ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
. . .	Horizontal ellipsis points indicate either:	
	■ That we have omitted parts of the code that are not directly related to the example	`CREATE TABLE ... AS subquery;`
	■ That you can repeat a portion of the code	`SELECT col1, col2, ... , col`*n* `FROM emp;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other punctuation	You must enter punctuation other than brackets, braces, vertical bars, and ellipsis points as it is shown.	
Italics	Italicized text indicates variables for which you must supply particular values.	`STARTUP PFILE=init`*sid*`.ora` In this example, the entire string `init`*sid*`.ora` is a placeholder for a parameter file that must contain your particular instance ID or SID.

Convention	Meaning	Example

`UPPERCASE`	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT ename, empno FROM emp;`
		`SQLPLUS username/password`
		`INTO TABLENAME 'table'`

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.	`SELECT ename, empno FROM emp;` `SQLPLUS scott/tiger`

1

Introduction

Each chapter in this addendum represents a book in the generic server documentation set. However, not all of the books in the documentation set are represented by the chapters here. Rather, there are a number of the generic server books being released in their entirety due to the volume of new information.

Many new features, as well as changes due to bug fixes and developer feedback, are described in these chapters. Note that this addendum is not a summary of all the new features for this release. Instead, it is a collection of chapters, each representing a single book, that have been impacted by various features new for this release.

This chapter contains the following sections:

- Summary of New Features
- How This Addendum Is Organized

Summary of New Features

The following list describes new features for this release:

> **Note:** Not all of the new features listed here are described in this addendum. Rather, the new features for this release are described both within this addendum and various books that are being released in their entirety. Typically, new features for a release are described in detail in *Getting to Know Oracle8i*; however this book is not being revised in the current release. Thus, a condensed version of new features is listed here.

- Enhanced Support of Internet Applications
  - Oracle HTTP Server powered by Apache as the default HTTP listener for serving static HTML and stateless servlets. The database can now receive direct HTTP calls without the need for an application server or an outside listener.
- PL/SQL Gateway
  - An embedded PL/SQL Gateway is included, which provides native, out-of-the-box support in the Oracle server for deploying PL/SQL-based database applications on the web.
- Java Enhancements
  - Oracle8*i* JVM (formerly known as Oracle JServer), is Oracle's Java Virtual Machine in both the database and the new Oracle Internet Application Server (iAS). With its new features and containers, Oracle 8*i* JVM is a comprehensive enterprise Java platform that can be used to implement all layers of a typical enterprise Java application.
  - Oracle8*i* JVM Accelerator. The JVM Accelerator allows a Java program to be translated, compiled, and deployed as native C compiled code, thereby improving Java performance.
  - Shared Read Only Objects. Read-only objects can now be shared across sessions. In other words, all sessions now point to the same bits.
  - Oracle Servlet Engine. The Oracle Servlet Engine is Servlet 2.2 compliant and supports stateful Servlets where each Web client gets a database session.

- Oracle JavaServer Pages Engine (JSPs). Oracle8i JVM embeds a JSP engine (JSP 1.1 compliant) within this release.
- Enterprise Java Beans 1.1 Support. This release fully complies with the Enterprise Java Beans (EJB) 1.1 specification, which mandates Entity EJB support (bean-managed and container-managed) and XML deployment descriptors (both standard and provider-specific).
- Java Transaction API (JTA Support). This release supports the Java Transaction API (JTA) standard, including support for both server-side and client-side transaction demarcation.
- Enhanced Support of JDBC and SQLJ. In this release JDBC drivers have complete production support for JDBC 2.0. This release adds support for XA Resource API, PL/SQL tables of scalars, and support for basic statement caching that minimizes cursor creation and teardown overhead.

■ Enhanced XML Support

XML as an enabling technology for internet application development has been significantly expanded. Components include:

- XML Parsers, which enable programmatic access to XML documents supporting W#C DOM 1.0 and SAX 1.0 standards
- XSL Processors, transforming any XML document to another or other text-based format such as HTML conforming with the W3C XSLT and XPath 1.0 standards
- XML Class Generators, generating class files for the creation of XML documents from DTDs for use in applications, applets, and Java Server Pages
- XML Java Beans, adding visual and non-visual XML functionality, including viewing, parsing, and transforming XML documents
- XSQL Servlets, which produce dynamic XML or HTML documents from SQL queries over the internet with support for SQL operations and action handlers
- In addition to the XDKs, this release also includes the XML SQL Utility (XSU) for Java and PL/SQL. This component enables applications to directly serve XML documents from the results of SQL queries.

- Security Enhancements
  - This release supports longer encryption keys for RC4 (256-bit) and Triple DES (Data Encryption Standard), providing more secure and robust security.
  - SSL encryption of HTTP connections and strong encryption for thin JDBC connections
- Managed Standby Enhancements
  - You can now use the DUPLICATE statement to create a standby database.
  - Recovery Manager (RMAN) backup and recovery is supported on a standby database.
  - You can now create a standby control file within RMAN.
  - You can now create a complete standby database without first creating an RMAN backup.
- Replication Enhancements
  - Replication no longer requires a quiesce to create a single master site or add objects to a single master group.
- Oracle Parallel Server (OPS) Enhancements
  - Raw partition tablespace size requirements have increased.
  - Procedures for configuring secondary instance connections are simplified.
  - Procedures for connecting Recovery Manager (RMAN) to target databases in OPS clusters have been modified.
  - There is a new parameter, OPS_INTERCONNECTS, which provides information about additional cluster interconnects for use in OPS environments. Oracle uses information from this parameter to distribute traffic among interfaces.
- NLS Enhancements
  - Unicode 3.0 is supported in the Oracle character sets UTF8 and UTFE. It is synchronized with ISO/IEC 10646-1 second edition, and includes an additional 10,307 code points.
  - The new Hong Kong government character set coding standard (HKSCS) is now supported.

- A new utility, the Database Character Set Migration Utility, reports on the feasibility of migrating to a new database character set. It scans the data in the database (data dictionary and application data) and provides an analysis of the current data, highlighting data that will be replaced during the migration.
- Users can now view Traditional Chinese data on Traditional Chinese clients and Simplified Chinese data on Simplified Chinese clients, regardless of the type of Chinese data (Traditional or Simplified) being stored on the database.

■ Other Enhancements

- Object DDL can now be viewed; DDL actions can be logged; database object reports are now available.
- Resource Management Wizard is available to assist in configuring and managing the Database Resource Manager
- Customization of the content and formatting of enhanced notifications (paging and email) is now supported.
- The Paging Server can be configured from the Enterprise Manager console.
- Multiple nodes can be refreshed at the same time from the Oracle Enterprise Manager console.
- New interMedia Audio, Image, and Video Java classes and Visual Information Retrieval Java classes are provided. interMedia Audio, Image, and Video and Visual Information Retrieval have been enhanced to support image, audio, and video export from a database to a file.
- New interMedia Locator generic geocoding interface to third-party geocoding software for batch geocoding is provided.
- Oracle Spatial has been enhanced to support projection management (coordinate systems), linear referencing, and R*tree indexing.
- Oracle Internet Directory has been enhanced to support the use of different algorithms for encrypting directory passwords.
- Oracle Objects for OLE includes full support for Unicode (UTF8), a new Code Wizard for Stored Procedures to speed development time, and execution of SQL and PL/SQL statements in non-blocking (asynchronous) modes.

How This Addendum Is Organized

This book contains chapters that represent specific books in the Oracle generic documentation set. Specifically, each chapter represents a specific book, and contains new features that would appear in that book if it were released in its entirety. For example, all of the new information that would appear in *Net8 Administrator's Guide* for this release appears in Chapter 5, "Net8" of this addendum.

The following chapters (or books) are included in this addendum:

Chapter 1, "Introduction" briefly describes the organization of this book and release 8.1.7 documentation sources.

Chapter 2, "interMedia Text" describes the new and updated features for interMedia Text for release 8.1.7.

Chapter 3, "New Character Set Scanner Utility" introduces the Character Set Scanner Utility, a National Language Support utility in release 8.1.7 for checking data before migrating character sets.

Chapter 4, "Oracle Parallel Server"describes the changes to Oracle Parallel Server for release 8.1.7.

Chapter 5, "Net8" describes new and changed features for Net8 for this release.

Chapter 6, "PL/SQL Supplied Packages" describes new packages for this release.

Chapter 7, "Recovery Manager" new features and changes for Recovery Manager.

Chapter 8, "Reference" includes parameters, data dictionary views, and dynamic performance views that have been revised to include new features or changes as a result of documentation corrections.

Chapter 9, "Replication"describes new features in Oracle replication. This chapter also describes changes to existing replication features, in addition to documentation additions and corrections.

Chapter 10, "Replication Management API Reference" includes changes to the DBMS_REPCAT_INSTANTIATE and DBMS_REPCAT_RGT packages as well as some documentation additions.

Chapter 11, "Oracle Label Security Error Messages" includes all new and revised Oracle Label Security error messages for this release.

Chapter 12, "Standby Database" describes new and changed features for standby databases for this release.

Chapter 13, "Using Oracle Trace" describes changes to Oracle Trace.

Other Books Being Released for 8.1.7

In addition to the chapters in this addendum, the *Master Index* and the following books have been revised in their entirety:

Oracle8i Server and SQL*Plus

- *Oracle8i Migration*
- *Oracle8i SQL Reference*
- *SQL*Plus Getting Started for Windows*
- *SQL*Plus Quick Reference*
- *SQL*Plus User's Guide and Reference*

Oracle8i Server Application Development

- *Oracle8i Application Developer's Guide - XML*
- *Oracle8i Supplied Java Packages Reference*
- *Oracle8i XML Reference*

Oracle8i Directory, Networking and Security

- *Oracle Advanced Security Administrator's Guide*
- *Oracle Internet Directory Administrator's Guide*
- *Oracle Internet Directory Application Developer's Guide*

Oracle8i interMedia, Spatial, Time Series, and Visual Information Retrieval Options

- *Oracle Spatial User's Guide and Reference*
- *Oracle Visual Information Retrieval User's Guide and Reference*
- *Oracle Visual Information Retrieval Java Classes User's Guide and Reference*
- *Oracle interMedia Audio, Image, and Video User's Guide and Reference*
- *Oracle interMedia Locator User's Guide and Reference*
- *Oracle interMedia Audio, Image, and Video Java Classes User's Guide and Reference*

Oracle8i Integration Server

- *Oracle Integration Server Overview*
- *Oracle Workflow Guide*
- *Oracle Message Broker Administrator's Guide*

Oracle8i Java Documentation

- *Oracle8i Java Tools Reference*
- *Oracle8i Java Developer's Guide*
- *Oracle JavaServer Pages Developer's Guide and Reference*
- *Oracle8i Enterprise JavaBeans Developer's Guide and Reference*
- *Oracle8i CORBA Developer's Guide and Reference*
- *Oracle8i JDBC Developer's Guide and Reference*
- *Oracle8i SQLJ Developer's Guide and Reference*
- *Oracle8i Oracle Servlet Engine User's Guide*

Oracle Enterprise Manager

- *Oracle Enterprise Manager Administrator's Guide*
- *Oracle Enterprise Manager Concepts Guide*
- *Oracle Enterprise Manager Configuration Guide*
- *Oracle Enterprise Manager Messages Manual*
- *Oracle Intelligent Agent User's Guide*
- *Oracle SNMP Support Reference Guide*

2

interMedia Text

This chapter describes the new and updated features for *inter*Media Text, release 8.1.7.

The following topics are covered:

- Overview of New and Updated Features
- CATSEARCH New SQL Operator
- CREATE INDEX Updated Syntax
- ALTER INDEX Updated Syntax
- Fuzzy Operator New Syntax
- MULTI_COLUMN_DATASTORE New Datastore Type
- PROCEDURE_FILTER New Filter Type
- Prefix Indexing New Feature
- CTX_DDL.ADD_INDEX New Procedure
- CTX_DDL.CREATE_INDEX_SET New Procedure
- CTX_DDL.DROP_INDEX_SET New Procedure
- CTX_DDL.REMOVE_INDEX New Procedure
- CTX_DDL.OPTIMIZE_INDEX Updated Syntax
- CTX_DDL.CREATE_STOPLIST Updated Syntax
- CTX_DDL.ADD_STOPWORD Updated Syntax
- CTX_THES.CREATE_TRANSLATION New Procedure
- CTX_THES.DROP_TRANSLATION New Procedure
- CTX_THES.HAS_RELATION New Procedure

- CTX_THES.UPDATE_TRANSLATION New Procedure
- CTX_DOC.TOKENS New Procedure
- CTX_DOC.THEMES Updated Syntax
- CTX_DOC.GIST Updated Syntax

Overview of New and Updated Features

The following sections outline the new and updated features for interMedia Text, Release 8.1.7. These features are:

- Catalog (CTXCAT) Index
- Keyless Index
- Multi-Column Datastore
- URL Datastore Username and Password Support
- File and URL Datastore Security
- Procedure Filter
- Multi-Language Stoplist
- Prefix Indexing
- Single Token Index Optimization
- Document Tokens Service
- Thesaurus Translation Calls
- New Fuzzy Operator Functionality
- Inso Filter Enhancements
- Supplied French Knowledge Base
- User-Defined Knowledge Bases

Catalog (CTXCAT) Index

With previous releases of *inter*Media Text, you can create a text index of type CONTEXT. This type of index is suited for indexing document collections that contain large coherent documents.

For this release, you can create a new type of index called CTXCAT in addition to the CONTEXT indextype. This is a combined index on a text column and one or more other columns. It is also known as a catalog index.

The CTXCAT indextype is suited for indexing short text fragments, such as names, addresses and item descriptions that are stored in separate columns. This type of index offers better query performance for structured queries.

You query a CTXCAT index with the CATSEARCH operator in the WHERE clause of a SELECT statement.

Indexing Example

Consider a table called AUCTION with the following schema:

```
create table auction(
    item_id number,
    title varchar2(100),
    category_id number,
    price number,
    bid_close date);
```

Assume that queries on the table involve a mandatory text query clause and optional structured conditions on `category_id`. Results must be sorted based on either `bid_close`, `category_id`, or `price`.

You can create a catalog index to support the different types of structured queries a user might enter.

To create the indexes, first create the index set preference then add the required indexes to it.

The following example creates the index set preference and adds five different indexes to it:

```
begin
    ctx_ddl.create_index_set('auction_iset');
    ctx_ddl.add_index('auction_iset','bid_close');               /* index A */
    ctx_ddl.add_index('auction_iset','category_id, bid_close');  /* index B */
    ctx_ddl.add_index('auction_iset','bid_close, category_id');  /* index C */
    ctx_ddl.add_index('auction_iset','price, bid_close');        /* index D */
    ctx_ddl.add_index('auction_iset','bid_close, price');        /* index E */
end;
```

Create the combined catalog index with CREATE INDEX as follows:

```
create index auction_titlex on AUCTION(title) indextype is CTXCAT parameters
('index set auction_iset');
```

Querying Example

To query the title column for the word *camera*, you can issue regular and mixed queries as follows:

```
select from AUCTION where CATSEARCH(title, 'camera', NULL)>0;
```

The following query uses index A:

```
select from AUCTION where CATSEARCH(title, 'camera', 'bid_close=20-FEB-2000')>0;
```

The following query uses index B:

```
select from AUCTION where CATSEARCH(title, 'camera', 'category_id=99 order by
bid_close desc')>0;
```

The following query uses index C:

```
select from AUCTION where CATSEARCH(title, 'camera', 'bid_close=20-FEB-2000
order by category_id')>0;
```

The following query uses index D:

```
select from AUCTION where CATSEARCH(title, 'camera', 'price=200 order by bid_
close')>0;
```

The following query uses index E:

```
select from AUCTION where CATSEARCH(title, 'camera', 'bid_close=20-FEB-2000
order by price')>0;
```

See Also:

CREATE INDEX Updated Syntax

CTX_DDL.ADD_INDEX New Procedure

CTX_DDL.CREATE_INDEX_SET New Procedure

CTX_DDL.DROP_INDEX_SET New Procedure

CTX_DDL.REMOVE_INDEX New Procedure

Keyless Index

In previous releases, your text table required a primary key to be indexed. This is no longer required in this release. You can create text table without a primary key and and index your text column.

Multi-Column Datastore

Use the new MULTI_COLUMN_DATASTORE type when your text is stored in more than one column in your text table. During indexing, the system concatenates the text columns and indexes the text as a single document.

See Also: MULTI_COLUMN_DATASTORE New Datastore Type

URL Datastore Username and Password Support

With the URL_DATASTORE, you can store URLs in your text column for indexing. This release supports embedding username and password for FTP protocol URLs, as in:

```
ftp://username:password@ftp.hostname.com/dir/file.doc
```

See Also: *Oracle8i interMedia Text Reference* for more information about how to use the URL_DATASTORE.

File and URL Datastore Security

FILE and URL datastores access files on the database machine. This might be undesirable for sensitive sites, since any user can browse the file system accessible to the oracle user using these datastores.

For better security, this release introduces the FILE_ACCESS_ROLE system parameter. You or your DBA can set this system parameter to the name of a database role. If set, any user attempting to create an index using FILE or URL datastores must have this role, or the index creation will fail. For instance, if the DBA does:

```
begin
ctx_adm.set_parameter('FILE_ACCESS_ROLE','WHITEHAT');
end;
```

then when user SCOTT tries to index:

```
create index foox on foo(text) indextype is ctxsys.context parameters('datastore
ctxsys.file_datastore')
```

Oracle checks if SCOTT has the role WHITEHAT. If he does, then the create index will proceed as normal. If not, then the create index will fail.

Note: This check is made only at create index time. Setting this parameter or granting/revoking the named role has no effect on existing indexes using the file datastore

---

See Also: *Oracle8i interMedia Text Reference* for more information about how to use the FILE_DATASTORE and URL_DATASTORE.

Procedure Filter

Use the new PROCEDURE_FILTER filter preference to filter your documents with a PL/SQL or Java stored procedure. The stored procedure is called each time a document needs to be filtered during indexing.

Note: This feature is different from the existing USER_FILTER which calls an external executable to perform document filtering.

See Also: PROCEDURE_FILTER New Filter Type

Multi-Language Stoplist

Oracle8*i inter*Media Text supports multi-language stoplists. A multi-language stoplist is useful when you index a multi-language column, such as a text column with English, German, and Japanese documents.

See Also:

CTX_DDL.CREATE_STOPLIST Updated Syntax

CTX_DDL.ADD_STOPWORD Updated Syntax

Prefix Indexing

With normal indexing, right-truncated wildcard queries such as TO% can possibly expand into a large wordlist and degrade query performance.

To improve query performance of right-truncated wildcard queries, you can create a prefix index. This type of index records token prefixes. The trade-off is a bigger index for improved wildcard searching.

Prefix indexing is different from sub-string indexing which also improves wildcard queries.

See Also:

Prefix Indexing New Feature

Single Token Index Optimization

*inter*Media Text supports the optimization of single tokens in the index. The optimization of specific tokens in the index saves time over optimizing the entire index.

See Also:

CTX_DDL.OPTIMIZE_INDEX Updated Syntax

ALTER INDEX Updated Syntax

Document Tokens Service

Document services has the following two new features:

- You can identify the text tokens in a particular document using the CTX_DOC.TOKENS.
- You can specify the number of themes and theme summaries returned for CTX_DOC.THEMES and CTX_DOC.GIST

See Also:

CTX_DOC.TOKENS New Procedure

CTX_DOC.THEMES Updated Syntax

CTX_DOC.GIST Updated Syntax

Thesaurus Translation Calls

The CTX_THES PL/SQL package supports adding, updating, and removing translations in your thesaurus. These new procedures are the following:

- CTX_THES.CREATE_TRANSLATION New Procedure

- CTX_THES.DROP_TRANSLATION New Procedure
- CTX_THES.HAS_RELATION New Procedure
- CTX_THES.UPDATE_TRANSLATION New Procedure

New Fuzzy Operator Functionality

The fuzzy query operator now supports similarity scoring. You can order the results so that results with high similarity to the query word are ranked higher than results with low similarity.

You can also limit the number of expanded terms.

See Also: Fuzzy Operator New Syntax

Inso Filter Enhancements

Oracle8*i* *inter*Media Text supports most document formats for indexing with Inso filter technology.

See Also: *Oracle8i interMedia Text Reference* for more information about the Inso filter and *all* supported document formats

Newly Supported Formats

In this release, support for the following formats has been added:

Format	Version
Microsoft Word 2000	Word 2000
Microsoft Excel 2000	Excel 2000
Microsoft PowerPoint 2000	PowerPoint 2000
Corel WordPerfect for Windows	Versions through 9.0
QuattroPro for Windows	Versions through 9.0
Corel Presentations	Versions 8.0 and 9.0
Microsoft Project	Project 98
Visio graphics format	Visio 4, 5 2000

Format	Version
Ichitaro	Version 5, 6, 7, 8, an 9 (Text and paragraph attributes only for versions 5 and 6.)
CDR (if tiff image is embedded)	Corel Draw version 2.0 - 9.0
MSG	Microsoft Outlook mail format (Some limitations in field support.)

Supported Platforms

Inso filter technology is supported on the following platforms:

- Sun Solaris Sparc (2.5.1 - 7)
- IBM AIX RS 6000 (4.1.4 - 4.3)
- HP / UX HP9000 (9.0 - 11.0)
- DEC UNIX (4.0)
- SGI IRIX (6.3 "New 32 -bit")
- Microsoft Windows NT on Intel (4.0)
- Microsoft Windows 2000 (release candidate 2)
- DEC Alpha Windows NT (4.0)

Supplied French Knowledge Base

Oracle8*i inter*Media Text now provides a knowledge base for French in addition to English. The knowledge base is the language specific data used during theme analysis.

For other languages, you must provide your own thesauri. One or more thesauri in a language can be compiled to produce an interMedia Text knowledge base for that language using the ctxkbtc compiler.

User-Defined Knowledge Bases

In this release, Oracle8*i inter*Media Text extends theme functionality to other languages by allowing you to load your own knowledge base for any single-byte whitespace delimited language, including Spanish and French.

Theme functionality includes theme indexing, ABOUT queries, theme highlighting, and the generation of themes, gists, and theme summaries with CTX_DOC.

You extend theme functionality by adding a user-defined knowledge base. For example, you can create a Spanish knowledge base from a Spanish thesuarus.

To load your language-specific knowledge base, follow these steps:

1. Load your custom thesaurus using ctxload.
2. Set NLS_LANG so that the language portion is the target language. The charset portion must be a single-byte character set.
3. Compile the loaded thesaurus using `ctxkbtc`:

```
ctxkbtc -user ctxsys/ctxsys -name my_lang_thes
```

A knowledge base is compiled from the loaded thesaurus. To use this knowledge base in an index, specify the NLS_LANG language as the THEME_LANGUAGE attribute value for the BASIC_LEXER preference.

Limitations

The following limitations hold for adding knowledge bases:

- Oracle supplies knowledge bases in English and French only. You must provide your own thesaurus for any other language.
- You can only add knowledge bases for languages with single-byte character sets. You cannot create a knowledge base for languages which can be expressed only in multi-byte character sets. If the database is a multi-byte universal character set, such as UTF-8, the NLS_LANG parameter must still be set to a compatible single-byte character set when compiling the thesaurus.
- Adding a knowledge base works best for whitespace delimited languages.
- You can have at most one knowledge base per NLS language.

> **See Also:** *Oracle8i interMedia Text Reference* for more information about theme functionality

Knowledge Base Character Set

Knowledge bases can be in any single-byte character set. Supplied knowledge bases are in WE8ISO8859P1. You can store an extended knowledge base in another character set such as US7ASCII.

CTXKBTC Knowledge Base Compiler Supports Stopthemes

The ctxkbtc executable now takes an additional argument as follows:

```
ctxkbtc -user ctxsys/ctxsys -stoplist <stoplistname>
```

Stopwords in the stoplist are added to the knowledge base as useless words that are prevented from becoming themes or contributing to themes. You can still add stopthemes after running this command using CTX_DLL.ADD_STOPTHEME.

Hierarchical Query Feedback

Obtaining hierarchical query feedback information such as broader terms, narrower terms and related terms does not work in languages other than English and French.

In other languages, the knowledge bases are derived entirely from your thesauri. In such cases, Oracle recommends that you obtain hierarchical information from your thesauri.

CATSEARCH New SQL Operator

Use the CATSEARCH operator to search catalog indexes. Use this operator in the WHERE clause of a SELECT statement.

Syntax

```
CATSEARCH(
    [schema.]column,
    text_query       VARCHAR2,
    structured_query VARCHAR2,
RETURN NUMBER;
```

[schema.]column

Specify the text column to be searched on. This column must have a CTXCAT index associated with it.

text_query

Specify the query expression that defines your search in `column`. The CATSEARCH operator supports only the following query operations:

- Logical AND
- Logical OR (|)
- Logical NOT (-)
- " " (exact phrases quoted)

These operators have the following syntax:

Operation	Syntax	Description of Operation
Logical AND	a b c	Returns rows that contain a, b, and c.
Logical OR	a | b | c	Returns rows that contain a, b, or c.
Logical NOT	a - b	Returns rows that contain a and not b.
hyphen with no space	a-b	Hyphen ignored. Words such as *web-site* treated as a single query term.

Operation	Syntax	Description of Operation
" "	"a b c"	Returns rows that contain the phrase "a b c". For example, entering "XYZ CD Player" means return all rows that contain this phrase exactly.

structured_query

Specify the structured conditions and the ORDER BY clause. There must exist an index for any column you specify. For example, if you specify `'category_id=1 order by bid_close'`, you must have an index for `'category_id, bid_close'` as specified with CTX_DDL.ADD_INDEX.

With `structured_query`, you can use standard SQL syntax with only the following operators:

- =
- <=
- >=
- >
- <
- IN
- BETWEEN

When you use these operators, the following limitations apply:

- The left-hand side (the column name) must be a column named in at least one of the indexes of the index set
- The left-hand side must be a plain column name. Functions and expressions are not allowed.
- The right-hand side must be composed of literal values. Functions, expressions, other columns, subselects, are not allowed.
- Multiple criteria can be combined with AND. OR is not supported.

For example, these expressions are supported:

```
catsearch(text, 'dog', 'foo > 15')
catsearch(text, 'dog', 'bar = ''SMITH''')
```

```
catsearch(text, 'dog', 'foo between 1 and 15')
catsearch(text, 'dog', 'foo = 1 and abc = 123')
```

And these expression are not supported:

```
catsearch(text, 'dog', 'upper(bar) = ''A''')
catsearch(text, 'dog', 'bar LIKE ''A%''')
catsearch(text, 'dog', 'foo = abc')
catsearch(text, 'dog', 'foo = 1 or abc = 3')
```

Examples

A typical query with CATSEARCH might include a structured clause as follows to find all rows that contain the word *camera* with id of 99 ordered by `bid_close`:

```
select from AUCTION where CATSEARCH(title, 'camera', 'category_id=99 order by
bid_close desc')> 0;
```

The following query finds all rows with the exact phrase *XYZ CD Player*:

```
select from AUCTION where CATSEARCH(title, '"XYZ CD Player"', 'order by bid_
close desc')> 0;
```

The following query finds all rows with the terms *XYZ* and *CD* and *Player*:

```
select from AUCTION where CATSEARCH(title, 'XYZ CD Player', 'order by bid_close
desc')> 0;
```

The following query finds all rows with the term *CD-Player*:

```
select from AUCTION where CATSEARCH(title, 'CD-Player', 'order by bid_close
desc')> 0;
```

The following query finds all rows with the term *CD* and not *Player*:

```
select from AUCTION where CATSEARCH(title, 'CD - Player', 'order by bid_close
desc')> 0;
```

The following query finds all rows with the terms *CD* or *DVD* or *Speaker*:

```
select from AUCTION where CATSEARCH(title, 'CD | DVD | Speaker', 'order by bid_
close desc')> 0;
```

CREATE INDEX Updated Syntax

Note: This section describes the CREATE INDEX statement as it pertains to creating an interMedia Text domain index. In addition, it only describes those changes that are new for this release.

For a complete description of the CREATE INDEX statement, see *Oracle8i interMedia Text Reference* and *Oracle8i SQL Reference*.

Purpose

Use CREATE INDEX to create an *inter*Media Text index. An *inter*Media Text index is an Oracle domain index of either type `context` or `ctxcat`.

You must create an appropriate *inter*Media Text index to issue CONTAINS or CATSEARCH queries.

You can create two types of *inter*Media Text indexes:

- CONTEXT index. This is an index on one text column. You query this index with the CONTAINS operator in the WHERE clause of a SELECT statement.
- CTXCAT index, new for this release. This is a combined index on a text column and one or more other columns.You query this index with the CATSEARCH operator in the WHERE clause of a SELECT statement.

Required Privileges

You do not need the CTXAPP role to create an *inter*Media Text index. If you have Oracle grants to create a B-tree index on the text column, you have sufficient permission to create a text index. The issuing owner, table owner, and index owner can all be different users, which is the standard behavior for regular B-tree indexes.

Syntax for New CATALOG Indextype

```
CREATE INDEX [schema.]index on [schema.]table(column) INDEXTYPE IS ctxsys.ctxcat
[PARAMETERS(paramstring)] [PARALLEL n];
```

Parameter String Limitations for CTXCAT

To create a CTXCAT index, you use CREATE INDEX and optionally specify preferences in the parameter string like you do when you create a context index.

However, the keywords you can use in a CTXCAT parameter string are limited to:

Parameter	Use
INDEX SET	Specify the index set to use to create the catalog index.
DATASTORE	Specify the datastore preference.
LEXER	Specify your lexer preference.
MEMORY	Specify memory size.
STOPLIST	Specify your stoplist.
STORAGE	Specify your storage preference.
WORDLIST	Specify your wordlist preference.

Supported Preferences for CTXCAT

When you create an index of type ctxcat, you can use only the following index preferences:

Table 2–1

Preference Class	Supported Types
Datastore	DIRECT_DATASTORE
Filter	None
Lexer	BASIC_LEXER CHINESE_VGRAM_LEXER JAPANESE_VGRAM_LEXER KOREAN_LEXER
Wordlist	PREFIX_INDEX attribute of BASIC_WORDLIST for Japanese data.
Storage	BASIC_STORAGE
Stoplist	GENERIC_STOPLIST
Section Group	None

CTXCAT Supported Column Types

The index set can take up to 99 indexes, each made up of ordered lists of columns from the base table. However, there are two important restrictions on these columns:

- The allowable column types are NUMBER, DATE, and CHAR/VARCHAR2 with maximum length <= 30 bytes.
- NULL values are not allowed. A NULL in a user column will cause an index error and the row will not be indexed.

CTXCAT System Parameters

The catalog index has its own set of system parameters for setting preference defaults. This allows an installation to have one set of default preferences for context indexes, and another set for ctxcat indexes. The new system parameters are:

- DEFAULT_CTXCAT_LEXER
- DEFAULT_CTXCAT_STOPLIST
- DEFAULT_CTXCAT_WORDLIST
- DEFAULT_CTXCAT_STORAGE
- DEFAULT_CTXCAT_INDEX_SET

Note: While you can specify a wordlist preference for ctxcat indexes, most of the attributes do not apply, since the catsearch query language does not support wildcarding, fuzzy, and stemming. The only attribute of the wordlist preference that is useful is PREFIX_INDEX, for Japanese data.

CTXCAT Example

Refer to the section Catalog (CTXCAT) Index.

Syntax for CONTEXT Indextype

```
CREATE INDEX [schema.]index on [schema.]table(column) INDEXTYPE IS
ctxsys.context [PARAMETERS(paramstring)] [PARALLEL n];
```

See Also: *Oracle8i interMedia Text Reference* for more information about how to create a CONTEXT index.

ALTER INDEX Updated Syntax

Note: This section describes the ALTER INDEX statement as it pertains to managing an interMedia Text domain index. In addition it only describes those changes that are new for this release.

For a complete description of the ALTER INDEX statement, see *Oracle8i interMedia Text Reference* and the *Oracle8i SQL Reference*.

Purpose

Use ALTER INDEX REBUILD syntax to perform the following maintenance tasks for a text index:

- Rebuild the index using different preferences. See REBUILD Syntax.
- Resume a failed index operation (creation or optimization). See REBUILD Syntax.
- Process DML in batch (synchronize). See REBUILD Syntax.
- Optimize the index, fully or by token. Token optimization is new for this release. See REBUILD Syntax.
- Add stopwords to the index. See REBUILD Syntax.
- Add sections and stop sections to the index. See REBUILD Syntax.

REBUILD Syntax

The following syntax is used to rebuild the index, resume a failed operation, perform batch DML, add stopwords to the index, add sections and stop sections to index, or optimize the index:

```
ALTER INDEX [schema.]index REBUILD [ONLINE] [PARAMETERS (paramstring)];
```

ONLINE

Optionally specify the `ONLINE` parameter for nonblocking operation, which allows the index to be queried during an ALTER INDEX synchronize or optimize operation. You cannot specify `ONLINE` for replace, resume, or when adding stopwords or stop sections.

PARAMETERS (paramstring)

Optionally specify `paramstring`. If you do not specify `paramstring`, Oracle rebuilds the index with existing preference settings.

The syntax for `paramstring` is as follows:

```
paramstring = 'replace [datastore datastore_pref]
                       [filter filter_pref]
                       [lexer lexer_pref]
                       [wordlist wordlist_pref]
                       [storage storage_pref]
                       [stoplist stoplist]
                       [section group section_group]
                          [memory memsize]

|    resume [memory memsize]
|    optimize [token index_token | fast | full [maxtime (time | unlimited)]
|    sync [memory memsize]
|    add stopword word [language language]
|    add zone section section_name tag tag
|    add field section section_name tag tag [(VISIBLE | INVISIBLE)]
|    add attr section section_name tag tag@attr
|    add stop section tag'
```

optimize [token index_token | fast | full [maxtime (time | unlimited)]

Optimizes the index. Specify `token`, `fast`, or `full` optimization. You typically optimize after you synchronize the index.

When you optimize in `token` mode, Oracle optimizes only the index token `index_token` in `full` mode. Use this method of optimization to quickly optimize frequently searched terms.

When you optimize in `fast` mode, Oracle works on the entire index, compacting fragmented rows. However, in `fast` mode, old data is not removed.

When you optimize in `full` mode, you can optimize the whole index or a portion. This method compacts rows and removes old data.

You use the `maxtime` parameter to specify in minutes the time Oracle is to spend on the optimization operation. Oracle starts the optimization where it left off and optimizes until complete or until the time limit has been reached, whichever comes first. Specifying a time limit is useful for automating index optimization, where you set Oracle to optimize the index for a specified time on a regular basis.

When you specify maxtime `unlimited`, the entire index is optimized. This is the default. When you specify 0 for `maxtime`, Oracle performs minimal optimization.

Examples

Token Optimization

The following statement optimizes the token *Oracle* in token mode:

```
ALTER INDEX newsindex REBUILD PARAMETERS('optimize token Oracle');
```

Fast Optimization

The following statement optimizes `newsindex` in fast mode:

```
ALTER INDEX newsindex REBUILD PARAMETERS('optimize fast');
```

Full Optimization

To specify an optimization operation to last for three hours (180 minutes), issue the following statement:

```
ALTER INDEX newsindex REBUILD PARAMETERS('optimize full maxtime 180');
```

To optimize the entire index without regard to time, issue the following statement:

```
ALTER INDEX newsindex REBUILD PARAMETERS('optimize full maxtime unlimited');
```

To optimize the entire index and to allow queries to be issued during the optimization, issue the following statement:

```
ALTER INDEX newsindex REBUILD ONLINE PARAMETERS('optimize full maxtime
unlimited');
```

MULTI_COLUMN_DATASTORE New Datastore Type

Use this datastore when your text is stored in more than one column. During indexing, the system concatenates the text columns and indexes the text as a single document.

MULTI_COLUMN_DATASTORE has the following attributes:

Attribute	Attribute Value
columns	Specify a comma separated list of columns to be concatenated during indexing. You can also specify any expression allowable for the select statement column list for the base table. This includes expressions, PL/SQL functions, column aliases, and so on. NUMBER and DATE column types are supported. They are converted to text before indexing using the default format mask. The TO_CHAR function can be used in the column list for formatting. RAW and BLOB columns are directly concatenated as binary data. LONG, LONG RAW, NCHAR, and NCLOB, nested table columns and collections are not supported. You can specify no more than 500 columns.

Indexing and DML

To index, you must create a dummy column to specify in the CREATE INDEX statement. This column's contents are not made part of the virtual document, unless its name is specified in the columns attribute.

The index is synchronized only when the dummy column is updated. You can create triggers to propagate changes if needed.

Security

Only CTXSYS is allowed to create preferences for the MULTI_COLUMN_DATASTORE type. Any other user who attempts to create a MULTI_COLUMN_DATASTORE preference receives an error.

Oracle makes this restriction because when the columns attribute contains a function call, the call is made by the CTXSYS schema. The potential exists for a malicious CTXAPP users to execute arbitrary functions for which they do not have execute permission.

If this is too restrictive, you can create a stored procedure under CTXSYS to create MULTI_COLUMN_DATASTORE preferences. The effective user is CTXSYS, who

creates and owns the preferences. However, you can call this procedure from any schema as CTXSYS.

> **Note:** For even better security, you can drop your CTXSYS stored procedures or preferences immediately after creating the index.For optimal security, consider using the USER_DATASTORE type, which checks user permissions during indexing.

MULTI_COLUMN_DATASTORE Example

The following example creates a multi-column datastore preference called `my_multi` with three text columns:

```
begin
ctx_ddl.create_preference('my_multi', 'MULTI_COLUMN_DATASTORE');
ctx_ddl.set_attribute('my_multi', 'columns', 'column1, column2, column3');
end;
```

Tagging Behavior

During indexing, the system creates a virtual document for each row. The virtual document is composed of the contents of the columns concatenated in the listing order with column name tags automatically added. For example:

```
create table mc(id primary key, name varchar2(10), address varchar2(80));
insert into mc values(1, 'John Smith', '123 Main Street');
exec ctx_ddl.set_attibute('mymds', 'columns', 'name, address');
```

This produces the following virtual text for indexing:

```
<NAME>
John Smith
</NAME>
<ADDRESS>
123 Main Street
</ADDRESS>
```

The system indexes the text between the tags, ignoring the tags themselves. To index these tags as sections, you can optionally create field sections with the BASIC_SECTION_GROUP.

> **Note:** No section group is created when you use the MULTI_COLUMN_DATASTORE. To create sections for these tags, you must create a section group.

When you use expressions or functions, the tag is composed of the first 30 characters of the expression unless a column alias is used.

For example, if your expression is as follows:

```
exec ctx_ddl.set_attibute('mymds', 'columns', '4 + 17');
```

then it produces the following virtual text:

```
<4 + 17>
21
</4 + 17>
```

If your expression is as follows:

```
exec ctx_ddl.set_attibute('mymds', 'columns', '4 + 17 col1');
```

then it produces the following virtual text:

```
<col1>
21
<col1>
```

The tags are in uppercase unless the column name or column alias is in lowercase and surrounded by double quotes. For example:

```
exec ctx_ddl.set_attibute('mymds', 'COLUMNS', 'foo');
```

produces the following virtual text:

```
<FOO>
content of foo
</FOO>
```

For lowercase tags, use the following:

```
exec ctx_ddl.set_attibute('mymds', 'COLUMNS', 'foo "foo"');
```

This expression produces:

```
<foo>
content of foo
</foo>
```

PROCEDURE_FILTER New Filter Type

Use the PROCEDURE_FILTER filter preference type to filter your documents with a stored procedure. The stored procedure is called each time a document needs to be filtered.

This type has the following attributes:

Attribute	Purpose	Allowable Values
procedure	Name of the filter stored procedure.	Any CTXSYS owned procedure. The procedure can be a PL/SQL or Java stored procedure.
input_type	Type of input argument for stored procedure.	VARCHAR2, BLOB, CLOB, FILE
output_type	Type of output argument for stored procedure.	VARCHAR2, CLOB, FILE
rowid_parameter	Include rowid parameter?	TRUE/FALSE
format_parameter	Include format parameter?	TRUE/FLASE
charset_parameter	Include charset parameter?	TRUE/FLASE

procedure

Specify the name of the stored procedure to use for filtering. The procedure can be a PL/SQL or Java stored procedure. The procedure can be a safe callout or call a safe callout.

The procedure must be owned by CTXSYS and have one of the following signatures:

```
PROCEDURE(IN BLOB, IN OUT NOCOPY CLOB)
PROCEDURE(IN CLOB, IN OUT NOCOPY CLOB)
PROCEDURE(IN VARCHAR, IN OUT NOCOPY CLOB)
PROCEDURE(IN BLOB, IN OUT NOCOPY VARCHAR2)
PROCEDURE(IN CLOB, IN OUT NOCOPY VARCHAR2)
PROCEDURE(IN VARCHAR2, IN OUT NOCOPY VARCHAR2)
PROCEDURE(IN BLOB, IN VARCHAR2)
PROCEDURE(IN CLOB, IN VARCHAR2)
PROCEDURE(IN VARCHAR2, IN VARCHAR2)
```

The first argument is the content of the unfiltered row as passed out by the datastore. The second argument is for the procedure to pass back the filtered document text.

The procedure attribute is mandatory and has no default.

input_type

Specify the type of the input argument of the filter procedure. You can specify one of the following:

Type	Description
BLOB	The input argument is of type BLOB. The unfiltered document is contained in the BLOB passed in.
CLOB	The input argument is of type CLOB. The unfiltered document is contained in the CLOB passed in. No pre-filtering or character set conversion is done. If the datastore outputs binary data, that binary data is written directly to the CLOB, with NLS doing implicit mapping to character data as best it can.
VARCHAR2	The input argument is of type VARCHAR2. The unfiltered document is contained in the VARCHAR2 passed in. The document can be a maximum of 32767 bytes of data. If the unfiltered document is greater than this length, an error is raised for the document and the filter procedure is not called.
FILE	The input argument is of type VARCHAR2. The unfiltered document content is contained in a temporary file in the file system whose filename is stored in the VARCHAR2 passed in. For example, the value of the passed-in VARCHAR2 might be 'tmp/mydoc.tmp' which means that the document content is stored in the file '/tmp/mydoc.tmp'. The file input type is useful only when your procedure is a safe callout, which can read the file.

The input_type attribute s not mandatory. If not specified, BLOB is the default.

output_type

Specify the type of output argument of the filter procedure. You can specify one of the following types:

Type	Description
CLOB	The output argument is IN OUT NOCOPY CLOB. Your procedure must write the filtered content to the CLOB passed in.
VARCHAR2	The output argument is IN OUT NOCOPY VARCHAR2. Your procedure must write the filtered content to the VARCHAR2 variable passed in.
FILE	The output argument must be IN VARCHAR2. On entering the filter procedure, the output argument is the name of a temporary file. The filter procedure must write the filtered contents to this named file. Using a FILE output type is useful only when the procedure is a safe callout, which can write to the file.

The output_type attribute is not mandatory. If not specified, CLOB is the default.

rowid_ parameter

When you specify TRUE, the rowid of the document to be filtered is passed as the first parameter, before the input and output parameters.

For example, with INPUT_TYPE BLOB, OUTPUT_TYPE CLOB, and ROWID_ PARAMETER TRUE, the filter procedure must have the signature as follows:

```
procedure(in rowid, in blob, in out nocopy clob)
```

This attribute is useful for when your procedure requires data from other columns or tables. This attribute is not mandatory. The default is FALSE.

format_parameter

When you specify TRUE, the value of the format column of the document being filtered is passed to the filter procedure before input and output parameters, but after the rowid parameter, if enabled.

You specify the name of the format column at index time in the parameters string, using the keyword `'format column <columnname>'`. The parameter type must be IN VARCHAR2.

The format column value can be read via the rowid parameter, but this attribute allows a single filter to work on multiple table structures, because the format

attribute is abstracted and does not require the knowledge of the name of the table or format column.

FORMAT_PARAMETER is not mandatory. The default is FALSE.

charset_parameter

When you specify TRUE, the value of the charset column of the document being filtered is passed to the filter procedure before input and output parameters, but after the rowid and format parameter, if enabled.

You specify the name of the charset column at index time in the parameters string, using the keyword `'charset column <columnname>'`. The parameter type must be IN VARCHAR2.

CHARSET_PARAMETER attribute is not mandatory. The default is FALSE.

Parameter Order

ROWID_PARAMETER, FORMAT_PARAMETER, and CHARSET_PARAMETER are all independent. The order is rowid, the format, then charset, but the filter procedure is passed only the minimum parameters required.

For example, assume that INPUT_TYPE is BLOB and OUTPUT_TYPE is CLOB. If your filter procedure requires all parameters, the procedure signature must be:

```
(id IN ROWID, format IN VARCHAR2, charset IN VARCHAR2, input IN BLOB, output IN
OUT NOCOPY CLOB)
```

If your procedure requires only the ROWID, then the procedure signature must be:

```
(id IN ROWID,input IN BLOB, ouput IN OUT NOCOPY CLOB)
```

Create Index Requirements

In order to create an index using a PROCEDURE_FILTER preference, the index owner must have execute permission on the procedure.Oracle checks this at index time, which is similar to the security measures for USER_DATASTORE.

Error Handling

The filter procedure can raise any errors needed through the normal PL/SQL raise_application_error facility. These errors are propagated to the CTX_USER_INDEX_ERRORS view or reported to the user, depending on the context in which the filter is invoked.

Procedure Filter Preference Example

Consider a filter procedure CTXSYS.NORMALIZE that you define with the following signature:

```
PROCEDURE NORMALIZE(id IN ROWID, charset IN VARCHAR2, input IN CLOB,
output IN OUT NOCOPY VARCHAR2);
```

To use this procedure as your filter, you set up your filter preference as follows:

```
begin
ctx_ddl.create_preference('myfilt', 'procedure_filter');
ctx_ddl.set_attribute('myfilt', 'procedure', 'normalize');
ctx_ddl.set_attribute('myfilt', 'input_type', 'clob');
ctx_ddl.set_attribute('myfilt', 'output_type', 'varchar2');
ctx_ddl.set_attribute('myfilt', 'rowid_parameter', 'TRUE');
ctx_ddl.set_attribute('myfilt', 'charset_parameter', 'TRUE');
end;
```

Prefix Indexing New Feature

You can create a prefix index to improve right-truncated wildcard searches such as TO%. Without a prefix index, right truncated wildcard queries are expanded using equivalence. This type of expansion can possibly result in large wordlists, degrading query performance.

To enable prefix indexing, use the BASIC_WORDLIST preference type. The following new attributes have been added:

Table 2–2

Attribute	Attribute Values
index_prefix	Specify YES to enable prefix indexing. Prefix indexing improves performance for right truncated wildcard searches such as TO%. Defaults to NO.
prefix_length_min	Specify the minimum length of indexed prefixes. Defaults to 1.
prefix_length_max	Specify the maximum length of indexed prefixes. Defaults to 64.

index_prefix

Specify `yes` to enable prefix indexing. Prefix indexing improves performance for right truncated wildcard searches such as TO%. Defaults to NO.

> **Note:** Enabling prefix indexing increases index size.

Prefix indexing chops up tokens into multiple prefixes to store in the $I table.For example, words TOKEN and TOY are normally indexed like this in the $I table:

Token	Type	Information
TOKEN	0	DOCID 1 POS 1
TOY	0	DOCID 1 POS 3

With prefix indexing, Oracle indexes the prefix substrings of these tokens as follows with a new token type of 6:

Token	Type	Information
TOKEN	0	DOCID 1 POS 1
TOY	0	DOCID 1 POS 3

Token	Type	Information
T	6	DOCID 1 POS 1 POS 3
TO	6	DOCID 1 POS 1 POS 3
TOK	6	DOCID 1 POS 1
TOKE	6	DOCID 1 POS 1
TOKEN	6	DOCID 1 POS 1
TOY	6	DOCID 1 POS 3

Wildcard searches such as TO% are now faster because Oracle does no expansion of terms and merging of result sets. To obtain the result, Oracle need only examine the (TO,6) row.

prefix_length_min

Specify the minimum length of indexed prefixes. Defaults to 1.

For example, setting `prefix_length_min` to 3 and `prefix_length_max` to 5 indexes all prefixes between 3 and 5 characters long.

> **Note:** A wildcard search whose pattern is below the minimum length or above the maximum length is searched using the slower method of equivalence expansion and merging.

prefix_length_max

Specify the maximum length of indexed prefixes. Defaults to 64.

For example, setting `prefix_length_min` to 3 and `prefix_length_max` to 5 indexes all prefixes between 3 and 5 characters long.

> **Note:** A wildcard search whose pattern is below the minimum length or above the maximum length is searched using the slower method of equivalence expansion and merging.

Enabling Prefix Indexing

The following example sets the wordlist preference for prefix indexing. The example specifies that Oracle create token prefixes between 3 and 4 characters long:

```
begin
    ctx_ddl.create_preference('mywordlist', 'BASIC_WORDLIST');
    ctx_ddl.set_attribute('mywordlist','INDEX_PREFIX','YES');
    ctx_ddl.set_attribute('mywordlist','PREFIX_MIN_LENGTH',3);
    ctx_ddl.set_attribute('mywordlist','PREFIX_MAX_LENGTH', 4);
end
```

Fuzzy Operator New Syntax

Use the fuzzy operator to expand queries to include words that are spelled similarly to the specified term. This type of expansion is helpful for finding more accurate results when there are frequent misspellings in your document set.

The new fuzzy syntax enables you to rank the result set so that documents that contain words with high similarity to the query word are scored higher than documents with lower similarity. You can also limit the number of expanded terms.

Unlike stem expansion, the number of words generated by a fuzzy expansion depends on what is in the index. Results can vary significantly according to the contents of the index.

Supported Languages

Oracle8*i inter*Media Text supports fuzzy definitions for English, German, Italian, Dutch, Spanish, and OCR.

Stopwords

If the fuzzy expansion returns a stopword, the stopword is not included in the query or highlighted by CTX_DOC.HIGHLIGHT or CTX_DOC.MARKUP.

Base-Letter Conversion

If base-letter conversion is enabled for a text column and the query expression contains a fuzzy operator, Oracle operates on the base-letter form of the query.

New Syntax

```
fuzzy(term, score, numresults, weight)
```

Parameter	Description
term	Specify the word on which to perform the fuzzy expansion. Oracle expands term to include words only in the index.
score	Specify a similarity score. Terms in the expansion that score below this number are discarded. Use a number between 1 and 80. The default is 60.
numresults	Specify the maximum number of terms to use in the expansion of term. Use a number between 1 and 5000. The default is 100.

Parameter	Description
weight	Specify WEIGHT or W for the results to be weighted according to their similarity scores. Specify NOWEIGHT or N for no weighting of results.

Examples

Consider the CONTAINS query:

```
...CONTAINS(TEXT, 'fuzzy(government, 70, 6, weight)', 1) > 0;
```

This query expands to the first six fuzzy variations of *government* in the index that have a similarity score over 70.

In addition, documents in the result set are weighted according to their similarity to *government*. Documents containing words most similar to government receive the highest score.

You can skip unnecessary parameters using the appropriate number of commas. For example:

```
'fuzzy(government,,,weight)'
```

Backward Compatibility

The old fuzzy syntax from previous releases is still supported. This syntax is as follows:

Parameter	Description
?term	Expands term to include all terms with similar spellings as the specified term.

CTX_DDL.ADD_INDEX New Procedure

Use this procedure to add a B-tree index to a catalog index preference. You create a catalog index preference to create a CTXCAT index.

> **Note:** Invoking CTX_DDL.ADD_INDEX after creating a CTXCAT index does not update the domain index to include the new index. You must add B-tree indexes with this procedure and then create your CTXCAT index with CREATE INDEX.

Syntax

```
CTX_DDL.ADD_INDEX(
    set_name in varchar2,
    column_list varchar2,
    storage_clause varchar2);
```

set_name
Specify the name of the index set.

column_list
Specify a comma separated list of columns upon which to create the B-tree index. Order your columns according to your queries.

For example, if your structured query clause is `'column1=99 order by column2'`, specify `'column1, column2'` for optimal query performance.

Similarly, if your structured query clause is `'column2=200 order by column1'`, specify `'column2, column1'` for optimal query performance.

storage_clause
Specify a storage clause.

Example

Indexing

Consider a table called AUCTION with the following schema:

```
create table auction(
    item_id number,
    title varchar2(100),
    category_id number,
    price number,
    bid_close date);
```

Assume that queries on the table involve a mandatory text query clause and optional structured conditions on `category_id`. Results must be sorted based on either `bid_close`, `category_id`, or `price`.

You can create a catalog index to support the different types of structured queries a user might enter.

To create the indexes, first create the index set preference then add the required indexes to it.

The following example creates the index set preference and adds five different indexes to it:

```
begin
    ctx_ddl.create_index_set('auction_iset');
    ctx_ddl.add_index('auction_iset','bid_close');               /* index A */
    ctx_ddl.add_index('auction_iset','category_id, bid_close');  /* index B */
    ctx_ddl.add_index('auction_iset','bid_close, category_id');  /* index C */
    ctx_ddl.add_index('auction_iset','price, bid_close');        /* index D */
    ctx_ddl.add_index('auction_iset','bid_close, price');        /* index E */
end;
```

Create the combined catalog index with CREATE INDEX as follows:

```
create index auction_titlex on AUCTION(title) indextype is CTXCAT parameters
('index set auction_iset');
```

Querying

To query the title column for the word *camera*, you can issue regular and mixed queries as follows:

```
select from AUCTION where CATSEARCH(title, 'camera', NULL)>0;
```

The following query uses index A:

```
select from AUCTION where CATSEARCH(title, 'camera', 'bid_close=20-FEB-2000')>0;
```

The following query uses index B:

```
select from AUCTION where CATSEARCH(title, 'camera', 'category_id=99 order by
bid_close desc')>0;
```

The following query uses index C:

```
select from AUCTION where CATSEARCH(title, 'camera', 'bid_close=20-FEB-2000
order by category_id')>0;
```

The following query uses index D:

```
select from AUCTION where CATSEARCH(title, 'camera', 'price=200 order by bid_
close')>0;
```

The following query uses index E:

```
select from AUCTION where CATSEARCH(title, 'camera', 'bid_close=20-FEB-2000
order by price')>0;
```

CTX_DDL.CREATE_INDEX_SET New Procedure

Use this procedure to create an index set for CTXCAT index types. You name this index set in the parameter clause of CREATE INDEX when you create a CTXCAT index.

Syntax

```
CTX_DDL.CREATE_INDEX_SET(set_name in  varchar2);
```

set_name
Specify the name of the index set. You name this index set in the parameter clause of CREATE INDEX when you create a CTXCAT index.

CTX_DDL.DROP_INDEX_SET New Procedure

Use this procedure to drop an index set.

Syntax

```
CTX_DDL.DROP_INDEX_SET(set_name in varchar2);
```

set_name
Specify the name of the index set to drop.

CTX_DDL.REMOVE_INDEX New Procedure

Use this procedure to remove a column from the column list.

Syntax

```
CTX_DDL.REMOVE_INDEX(set_name in varchar2, column_list in varchar2);
```

set_name
Specify the name of the index set to drop.

column_list
Specify the column list to remove from the index set preference.

CTX_DDL.OPTIMIZE_INDEX Updated Syntax

Use this procedure to optimize the index. You optimize your index after you synchronize it. Optimizing the index removes old data and minimizes index fragmentation. Optimizing the index can improve query response time

You can optimize in fast, full, or token mode. In token mode, you specify a specific token to be optimized. You can use token mode to optimize index tokens that are frequently searched, without spending time on optimizing tokens that are rarely referenced. An optimized token can improve query response time for that token.

> **Note:** Optimizing an index can result in better response time only if you insert, delete, or update documents in your base table after your initial indexing operation.

Using this procedure to optimize the index is the same as optimizing with the ALTER INDEX statement.

Syntax

```
CTX_DDL.OPTIMIZE_INDEX(
    idx_name  in  varchar2,
    optlevel  in  varchar2,
    maxtime   in  number default null
    token     in varchar2 default null
);
```

idx_name

Specify the name of the index.

optlevel

Specify optimization level as a string. You can specify one of the following methods for optimization:

Value	Description
FAST or CTX_DDL.OPTLEVEL_FAST	This method compacts fragmented rows. However, old data is not removed.

Value	Description
FULL or CTX_DDL.OPTLEVEL_FULL	In this mode you can optimize the entire index or a portion of the index. This method compacts rows and removes old data.
TOKEN	This method lets you specify a specific token to be optimized. Oracle does a FULL optimization on the token you specify with token. Use this method to optimize those tokens that are searched frequently.

maxtime

Specify maximum optimization time, in minutes, for FULL optimize.

When you specify the symbol CTX_DDL.MAXTIME_UNLIMITED (or pass in NULL), the entire index is optimized. This is the default.

token

Specify the token to be optimized.

Example

The following two examples optimize the index for fast optimization.

```
begin
ctx_ddl.optimize_index('myidx','FAST');
end;
```

```
begin
ctx_ddl.optimize_index('myidx',CTX_DDL.OPTLEVEL_FAST);
end;
```

The following example optimizes the index token *Oracle*:

```
begin
ctx_ddl.optimize_index('myidx','token', TOKEN=>'Oracle');
end;
```

CTX_DDL.CREATE_STOPLIST Updated Syntax

Use this procedure to create a new, empty stoplist. Stoplists can contain words or themes that are not to be indexed.

You can also create multi-language stoplists to hold language-specific stopwords. A multi-lingual stoplist is useful when you index a table that contains documents in different languages, such as English, German, and Japanese.

You can add either stopwords, stopclasses, or stopthemes to a stoplist using ADD_STOPWORD, ADD_STOPCLASS, or ADD_STOPTHEME.

You can specify a stoplist in the parameter string of CREATE INDEX or ALTER INDEX to override the default stoplist CTXSYS.DEFAULT_STOPLIST.

Syntax

```
CTX_DDL.CREATE_STOPLIST(
    stoplist_name IN VARCHAR2,
    stoplist_type IN VARCHAR2 DEFAULT 'BASIC_STOPLIST');
```

stoplist_name
Specify the name of the stoplist to be created.

stoplist_type
Specify MULTI_STOPLIST to create a stoplist with language-specific stopwords.

> **Note:** When indexing a multi-lingual table with a multi-lingual stoplist, your table must have a language column.

Specify BASIC_STOPLIST to create a stoplist for a single language. This is the default.

Example

The following code creates a stoplist called `mystop`:

```
begin
ctx_ddl.create_stoplist('mystop', 'BASIC_STOPLIST');
end;
```

CTX_DDL.ADD_STOPWORD Updated Syntax

Use this procedure to add a single stopword to a stoplist. To create a list of stopwords, you must call this procedure once for each word.

> **Note:** The maximum number of stopwords, stopthemes, and stopclasses you can add to a stoplist is 4095.

Syntax

```
CTX_DDL.ADD_STOPWORD(
    stoplist_name  in   varchar2,
    stopword       in   varchar2,
    language       in varchar2 default NULL
);
```

stoplist_name
Specify the name of the stoplist.

stopword
Specify the stopword to be added.

Language-specific stopwords must be unique across the other stopwords specific to the language. For example, it is valid to have a German *die* and an English *die* in the same stoplist.

The maximum number of stopwords, stopthemes, and stopclasses you can add to a stoplist is 4095.

language
Specify the language of `stopword` when `stoplist_name` is of type MULTI_STOPWORD. You must specify the NLS name or abbreviation of an Oracle-supported language.

Otherwise, specify NULL.

Example

Single Language Stoplist

The following example adds the stopwords *because*, *notwithstanding*, *nonetheless*, and *therefore* to the stoplist `mystop`:

```
begin
ctx_ddl.add_stopword('mystop', 'because');
ctx_ddl.add_stopword('mystop', 'notwithstanding');
ctx_ddl.add_stopword('mystop', 'nonetheless');
ctx_ddl.add_stopword('mystop', 'therefore');
end;
```

Multi-Language Stoplist

The following example adds the German word *die* to a multi-language stoplist:

```
begin
    ctx_ddl.add_stopword('mulitstop', 'die', 'd');
end;
```

Note: You can add stopwords after you create the index with ALTER INDEX.

CTX_THES.CREATE_TRANSLATION New Procedure

Use this procedure to create a new translation for a phrase in a specified language.

Syntax

```
CTX_THES.CREATE_TRANSLATION(tname       in    varchar2,
                            phrase      in    varchar2,
                            language    in    varchar2,
                            translation in    varchar2);
```

tname
Specify the name of the thesaurus, using no more than 30 characters.

phrase
Specify the phrase in the thesaurus to which to add a translation. Phrase must already exist in the thesaurus, or an error is raised.

language
Specify the language of the translation, using no more than 10 characters.

translation
Specify the translated term, using no more than 256 characters.

If a translation for this phrase already exists, this new translation is added without removing that original translation, so long as that original translation is not the same. Adding the same translation twice results in an error.

Example

The following code adds the Spanish translation for *dog* to *my_thes*:

```
begin
   ctx_thes.create_translation('my_thes', 'dog', 'SPANISH', 'PERRO');
end;
```

CTX_THES.DROP_TRANSLATION New Procedure

Use this procedure to remove one or more translations for a phrase.

Syntax

```
CTX_THES.DROP_TRANSLATION (tname       in    varchar2,
                           phrase      in    varchar2,
                           language    in    varchar2 default null,
                           translation in    varchar2 default null);
```

tname
Specify the name of the thesaurus, using no more than 30 characters.

phrase
Specify the phrase in the thesaurus to which to remove a translation. The phrase must already exist in the thesaurus or an error is raised.

language
Optionally, specify the language of the translation, using no more than 10 characters. If not specified, the translation must also not be specified and all translations in all languages for the phrase are removed. An error is raised if the phrase has no translations.

translation
Optionally, specify the translated term to remove, using no more than 256 characters. If no such translation exists, an error is raised.

Example

The following code removes the Spanish translation for *dog*:

```
begin
   ctx_thes.drop_translation('my_thes', 'dog', 'SPANISH', 'PERRO');
end;
```

CTX_THES.HAS_RELATION New Procedure

Use this procedure to test that a thesaurus relation exists without actually doing the expansion. The function returns TRUE if the phrase has any of the relations in the specified list.

Syntax

```
CTX_THES.HAS_RELATION(phrase in varchar2,
                      rel in varchar2,
                      tname in varchar2 default 'DEFAULT')
  returns boolean;
```

phrase
Specify the phrase.

rel
Specify a single thesaural relation or a comma-separated list of relations, except PT. Specify 'ANY' for any relation.

tname
Specify the thesaurus name.

Example

The following example returns TRUE if the phrase "cat" in the DEFAULT thesaurus has any broader terms or broader generic terms:

```
begin
 ctx_thes.has_relation('cat','BT,BTG');
end;
```

CTX_THES.UPDATE_TRANSLATION New Procedure

Use this procedure to update an existing translation.

Syntax

```
CTX_THES.UPDATE_TRANSLATION(tname       in     varchar2,
                            phrase      in     varchar2,
                            language    in     varchar2,
                            translation in     varchar2,
                            new_translation in varchar2);
```

tname
Specify the name of the thesaurus, using no more than 30 characters.

phrase
Specify the phrase in the thesaurus to which to update a translation. The phrase must already exist in the thesaurus or an error is raised.

language
Specify the language of the translation, using no more than 10 characters.

translation
Specify the translated term to update no more than 256 characters. If no such translation exists, an error is raised.

You can specify NULL if there is only one translation for the *phrase*. An error is raised if there is more than one translation for the term in the specified language.

new_translation
Optionally, specify the new form of the translated term.

Example

The following code updates the Spanish translation for *dog*:

```
begin
   ctx_thes.drop_translation('my_thes', 'dog', 'SPANISH', 'PERRO', 'CAN');
end;
```

CTX_DOC.TOKENS New Procedure

Use this procedure to identify all text tokens in a document. The tokens returned are those tokens which are inserted into the index.

Stop words are not returned. Section tags are not returned because they are not text tokens.

Syntax 1: In-Memory Table Storage

```
CTX_DOC.TOKENS(index_name      IN VARCHAR2,
               textkey         IN VARCHAR2,
               restab          IN OUT NOCOPY TOKEN_TAB);
```

Syntax 2: Result Table Storage

```
CTX_DOC.TOKENS(index_name      IN VARCHAR2,
               textkey         IN VARCHAR2,
               restab          IN VARCHAR2,
               query_id        IN NUMBER DEFAULT 0);
```

index_name
Specify the name of the index for the text column.

textkey
Specify the unique identifier (usually the primary key) for the document.

The textkey parameter can be one of the following:

- a single column primary key value
- encoded specification for a composite (multiple column) primary key.
- the rowid of the row containing the document

You toggle between primary key and rowid identification using CTX_DOC.SET_KEY_TYPE.

restab
You can specify that this procedure store results to either a table or to an in-memory PL/SQL table.

The tokens returned are those tokens which are inserted into the index. Stop words are not returned. Section tags are not returned because they are not text tokens.

Token Table

To store results to a table, specify the name of the table. Token tables can be named anything, but must include the following columns, with names and data types as specified.

Table 2–3

Column Name	Type	Description
QUERY_ID	NUMBER	The identifier for the results generated by a particular call to CTX_DOC.TOKEN (only populated when table is used to store results from multiple TOKEN calls)
TOKEN	VARCHAR2(64)	The token string in the text.
OFFSET	NUMBER	The position of the token in the document, relative to the start of document which has a position of 1.
LENGTH	NUMBER	The character length of the token.

In-Memory Table

To store results to an in-memory table, specify the name of the in-memory table of type TOKEN_TAB. The TOKEN_TAB datatype is defined as follows:

```
type token_rec is record (
    token varchar2(64);
    offset number;
    length number;
);

type token_tab is table of token_rec index by binary_integer;
```

CTX_DOC.TOKENS clears the TOKEN_TAB you specify before the operation.

query_id
Specify the identifier used to identify the row(s) inserted into `restab`.

Examples

In-Memory Tokens

The following example generates the tokens for document 1 and stores them in an in-memory table, declared as `the_tokens`. The example then loops through the table to display the document tokens.

```
declare
 the_tokens ctx_doc.token_tab;

begin
 ctx_doc.tokens('myindex','1',the_tokens);
 for i in 1..the_tokens.count loop
  dbms_output.put_line(the_tokens(i).token);
  end loop;
end;
```

CTX_DOC.THEMES Updated Syntax

Use the CTX_DOC.THEMES procedure to generate a list of themes for a document. Each theme is stored as a row in either a result table or in-memory PL/SQL table you specify.

Syntax 1: In-Memory Table Storage

```
CTX_DOC.THEMES(
    index_name      IN VARCHAR2,
    textkey         IN VARCHAR2,
    restab          IN OUT THEME_TAB,
    full_themes     IN BOOLEAN DEFAULT FALSE,
    numthemes       IN NUMBER DEFAULT 50);
```

Syntax 2: Result Table Storage

```
CTX_DOC.THEMES(
    index_name      IN VARCHAR2,
    textkey         IN VARCHAR2,
    restab          IN VARCHAR2,
    query_id        IN NUMBER DEFAULT 0,
    full_themes     IN BOOLEAN DEFAULT FALSE,
    numthemes       IN NUMBER DEFAULT 50);
```

index_name

Specify the name of the index for the text column.

textkey

Specify the unique identifier (usually the primary key) for the document.

The textkey parameter can be one of the following:

- a single column primary key value
- an encoded specification for a composite (multiple column) primary key. When textkey is a composite key, you must encode the composite textkey string using the CTX_DOC.PKENCODE procedure.
- the rowid of the row containing the document

You toggle between primary key and rowid identification using CTX_DOC.SET_KEY_TYPE.

restab

You can specify that this procedure store results to either a table or to an in-memory PL/SQL table.

To store results in a table, specify the name of the table.

> **See Also:** For more information about the structure of the theme result table, see the theme table description in the see *Oracle8i interMedia Text Reference*.

To store results in an in-memory table, specify the name of the in-memory table of type THEME_TAB. The THEME_TAB datatype is defined as follows:

```
type theme_rec is record (
   theme varchar2(2000);
   weight number;
);

type theme_tab is table of theme_rec index by binary_integer;
```

CTX_DOC.THEMES clears the THEME_TAB you specify before the operation.

query_id

Specify the identifier used to identify the row(s) inserted into `restab`.

full_themes

Specify whether this procedure generates a single theme or a hierarchical list of parent themes (full themes) for each document theme.

Specify TRUE for this procedure to write full themes to the THEME column of the result table.

Specify FALSE for this procedure to write single theme information to the THEME column of the result table. This is the default.

numthemes

Specify the number of themes to retrieve. For example, if you specify 10, the top 10 themes are returned for the document. The default is 50.

If you specify 0 or NULL, this procedure returns all themes in a document. If the document contains more than 50 themes, only the top 50 themes show conceptual hierarchy.

Examples

In-Memory Themes

The following example generates the top 10 themes for document 1 and stores them in an in-memory table called `the_themes`. The example then loops through the table to display the document themes.

```
declare
 the_themes ctx_doc.theme_tab;

begin
 ctx_doc.themes('myindex','1',the_themes, numthemes=>10);
 for i in 1..the_themes.count loop
  dbms_output.put_line(the_themes(i).theme||':'||the_themes(i).weight);
  end loop;
end;
```

Theme Table

The following example creates a theme table called CTX_THEMES:

```
create table CTX_THEMES (query_id number,
                         theme varchar2(2000),
                         weight number);
```

Single Themes

To obtain a list of the top 20 themes where each element in the list is a single theme, issue a statement like the following:

```
begin
     ctx_doc.themes('newsindex','34','CTX_THEMES',1,full_themes => FALSE,
     numthemes=> 20);
end;
```

Full Themes

To obtain a list of the top 20 themes where each element in the list is a hierarchical list of parent themes, issue a statement like the following:

```
begin
    ctx_doc.themes('newsindex','34','CTX_THEMES',1,full_themes => TRUE,
    numthemes=>20);
end;
```

CTX_DOC.GIST Updated Syntax

Use the CTX_DOC.GIST procedure to generate a gist and theme summaries for a document. You can generate paragraph-level or sentence-level gists or theme summaries.

Syntax 1: In-Memory Storage

```
CTX_DOC.GIST(
    index_name    IN VARCHAR2,
    textkey       IN VARCHAR2,
    restab        IN OUT CLOB,
    glevel        IN VARCHAR2 DEFAULT 'P',
    pov           IN VARCHAR2 DEFAULT NULL,
    numParagraphs IN NUMBER DEFAULT 16,
    maxPercent    IN NUMBER DEFAULT 10,
    numthemes   IN NUMBER DEFAULT 50);
```

Syntax 2: Result Table Storage

```
CTX_DOC.GIST(
    index_name    IN VARCHAR2,
    textkey       IN VARCHAR2,
    restab        IN VARCHAR2,
    query_id      IN NUMBER DEFAULT 0,
    glevel        IN VARCHAR2 DEFAULT 'P',
    pov           IN VARCHAR2 DEFAULT NULL,
    numParagraphs IN NUMBER DEFAULT 16,
    maxPercent    IN NUMBER DEFAULT 10,
    numthemes     IN NUMBER DEFAULT 50);
```

index_name

Specify the name of the index associated with the text column containing the document identified by `textkey`.

textkey

Specify the unique identifier (usually the primary key) for the document.

The textkey parameter can be one of the following:

- a single column primary key value

- an encoded specification for a composite (multiple column) primary key. To encode a composite textkey, use the CTX_DOC.PKENCODE procedure.
- the rowid of the row containing the document

You toggle between primary key and rowid identification using CTX_DOC.SET_KEY_TYPE.

restab

You can specify that this procedure store the gist and theme summaries to either a table or to an in-memory CLOB.

To store results to a table specify the name of the table.

> **See Also:** For more information about the structure of the gist result table, see the gist table description in the *Oracle8i interMedia Text Reference*.

To store results in memory, specify the name of the CLOB locator. If restab is NULL, a temporary CLOB is allocated and returned. You must de-allocate the locator after using it.

If `restab` is not NULL, the CLOB is truncated before the operation.

query_id

Specify an identifier to use to identify the row(s) inserted into `restab`.

glevel

Specify the type of gist or theme summary to produce. The possible values are:

- *P* for paragraph
- *S* for sentence

The default is *P*.

pov

Specify whether a gist or a single theme summary is generated. The type of gist or theme summary generated (sentence-level or paragraph-level) depends on the value specified for `glevel`.

To generate a gist for the entire document, specify a value of 'GENERIC' for pov. To generate a theme summary for a single theme in a document, specify the theme as the value for pov.

When using result table storage and you do not specify a value for pov, this procedure returns the generic gist plus up to fifty theme summaries for the document.

When using in-memory result storage to a CLOB, you must specify a pov. However, if you do not specify pov, this procedure generates only a generic gist for the document.

Note: The pov parameter is case sensitive. To return a gist for a document, specify 'GENERIC' in all uppercase. To return a theme summary, specify the theme *exactly* as it is generated for the document.

Only the themes generated by CTX_DOC.THEMES Updated Syntax for a document can be used as input for pov.

---

numParagraphs

Specify the maximum number of document paragraphs (or sentences) selected for the document gist or theme summaries. The default is 16.

Note: The `numParagraphs` parameter is used only when this parameter yields a smaller gist or theme summary size than the gist or theme summary size yielded by the `maxPercent` parameter.

This means that the system always returns the smallest size gist or theme summary.

---

maxPercent

Specify the maximum number of document paragraphs (or sentences) selected for the document gist or theme summaries as a percentage of the total paragraphs (or sentences) in the document. The default is 10.

Note: The `maxPercent` parameter is used only when this parameter yields a smaller gist or theme summary size than the gist or theme summary size yielded by the `numParagraphs` parameter.

This means that the system always returns the smallest size gist or theme summary.

---

numthemes
Specify the number of theme summaries to produce when you do not specify a value for pov. For example, if you specify 10, this procedure returns the top 10 theme summaries. The default is 50.

If you specify 0 or NULL, this procedure returns all themes in a document. If the document contains more than 50 themes, only the top 50 themes show conceptual hierarchy.

Examples

In-Memory Gist

The following example generates a non-default size generic gist of at most 10 paragraphs. The result is stored in memory in a CLOB locator. The code then de-allocates the returned CLOB locator after using it.

```
declare
  gklob clob;
  amt number := 40;
  line varchar2(80);

begin
 ctx_doc.gist('newsindex','34','gklob',1,glevel => 'P',pov => 'GENERIC',
numParagraphs => 10);
  -- gklob is NULL when passed-in, so ctx-doc.gist will allocate a temporary
  -- CLOB for us and place the results there.

  dbms_lob.read(gklob, amt, 1, line);
  dbms_output.put_line('FIRST 40 CHARS ARE:'||line);
  -- have to de-allocate the temp lob
  dbms_lob.freetemporary(gklob);
 end;
```

Result Table Gists

The following example creates a gist table called CTX_GIST:

```
create table CTX_GIST (query_id  number,
                       pov       varchar2(80),
                       gist      CLOB);
```

Gists and Theme Summaries The following example returns a default sized paragraph level gist for document 34 as well as the top 10 theme summaries in the document:

```
begin
   ctx_doc.gist('newsindex','34','CTX_GIST', 1, glevel => 'P', numthemes=10);
end;
```

The following example generates a non-default size gist of at most 10 paragraphs:

```
begin
  ctx_doc.gist('newsindex','34','CTX_GIST',1,glevel => 'P',pov => 'GENERIC',
numParagraphs => 10);
end;
```

The following example generates a gist whose number of paragraphs is at most 10 percent of the total paragraphs in document:

```
begin
  ctx_doc.gist('newsindex','34','CTX_GIST',1, glevel =>'P',pov => 'GENERIC',
maxPercent => 10);
end;
```

Theme Summary The following example returns a paragraph level theme summary for *insects* for document 34. The default theme summary size is returned.

```
begin
   ctx_doc.gist('newsindex','34','CTX_GIST',1,glevel =>'P', pov => 'insects');
end;
```

3

New Character Set Scanner Utility

This chapter introduces the Character Set Scanner Utility, a National Language Support utility in release 8.1.7 for checking data before migrating character sets. The topics in this chapter include:

- Overview of Choosing and Migrating Character Sets
- Database Character Set Migration
- What is the Character Set Scanner Utility?
- Scan Modes in the Scanner
- Using The Scanner
- Scanner Parameters
- Sample Scanner Sessions
- Storage and Performance Considerations in the Scanner
- Reference Material

Overview of Choosing and Migrating Character Sets

Choosing the appropriate database character set for your database is an important decision and requires taking into account many factors. Some of these factors are:

- The type of data you need to store
- The number of languages the database character set can represent
- The different sizing requirements of each character set and their performance implications

A related topic is choosing a new character set for an existing database, which is called migrating character sets. Migrating from one database character set to another involves additional considerations beyond those of simply choosing a character set. In particular, it is a complex planning process with the goal of minimizing the possibility of losing data because of data truncation and character set conversions during the migration.

> **See Also:** *Oracle8i National Language Support Guide* for further details regarding choosing character sets

Data Truncation

The sizes of character data types `CHAR` and `VARCHAR2` are specified in bytes, not characters. Hence, the specification `CHAR(20)` in a table definition allows 20 bytes for storing character data. This is acceptable when the database character set uses a single-byte character encoding scheme because the number of characters will be equivalent to the number of bytes. If the database character set uses a multi-byte character encoding scheme, however, there is no such correspondence. That is, the number of bytes no longer equals the number of characters because a character can consist of one or more bytes. This situation can cause problems.

During migration to a new character set, it is important to verify the column widths of existing `CHAR` and `VARCHAR` columns because they might need to be extended to support encoding that requires multi-byte storage. If the character set width differs during the import process, truncation of data can occur if conversion causes expansion of data. Figure 3–1 shows a typical case of data expansion with single-byte characters becoming multibyte. In it, ä (a with an umlaut) is a single-byte character in WE8MSWIN1252, but it becomes a double-byte character in UTF8. Also, the Euro symbol goes from one byte to three bytes in this conversion.

Figure 3–1 Single-byte Character Sets Becoming Multibyte

Char	WE8MSWIN1252	UTF8
ä	E4	C3 A4
ö	F6	C3 B6
©	A9	C2 A9
€	80	E2 82 AC

The maximums for CHAR and VARCHAR2 data types are 2000 and 4000 bytes respectively. If the data columns in the new destination character set require more than 2000 and 4000 bytes, you will need to change your schema.

Restrictions

Here are some known restrictions caused by data truncation.

- Within the database data dictionary, schema object names cannot exceed 30 bytes in length. Schema objects are tables, clusters, views, indexes, synonyms, tablespaces, and usernames. Renaming schema objects is required if they exceed 30 bytes in the new database character set. For example, one Thai character in the Thai national character set requires 1 byte, but, in UTF8, it requires 3 bytes. So, if you have defined a table with 11 Thai characters, then this table name must be shortened to 10 or fewer Thai characters when changing the database character set to UTF8.
- If your existing Oracle usernames or passwords are created based on characters that will change in size on the target character set, these users will experience login difficulties due to authentication failures after the migration to a new character set. This is because the encrypted usernames and passwords stored in the data dictionary are not updated during migration to a new character set. For example, assuming the current database character set is WE8MSWIN1252 and the target database character set is UTF8, the username scött (o with an umlaut) will change from 5 bytes in WE8MSWIN1252 to 6 bytes. In UTF8, scött will no longer be able to log in because of the difference in the length of the username. Oracle recommends that usernames and passwords be based on ASCII characters. If they are not, you will need to reset the affected usernames and passwords after migrating to a new character set.

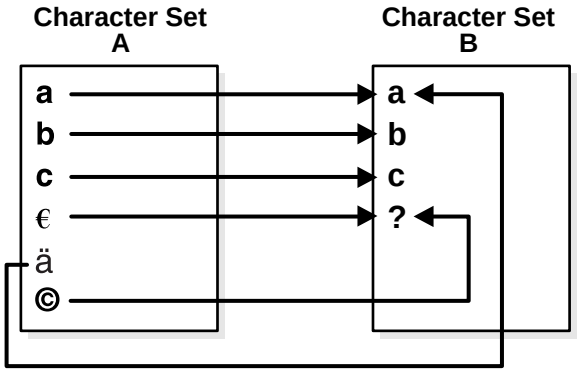
- When CHAR data contains characters that will be expanded after migration to a new character set, space padding will not be removed during database export by default. This means that these rows will be rejected upon import into the database with the new character set. The workaround is to set the initialization parameter BLANK_TRIMMING to TRUE prior to the import.

  See Also: *Oracle8i Reference* for further information about BLANK_TRIMMING

Character Set Conversions

When migrating to a new database character set, the Export and Import utilities can handle character set conversions from the original database character set to the new database character set. However, character set conversions can sometimes cause data loss or data corruption. For example, if you are migrating from character set A to character set B, the destination character set B should be a superset of character set A. Characters that are not available in character set B will be converted to replacement characters, which are usually specified as '?' or '¿' or other linguistically-related characters. For example, ä (a with an umlaut) will be converted to 'a'. Replacement characters are defined by the target character set. Figure 3–2 shows a sample conversion where the copyright and Euro symbols are converted to '?' and ä to 'a'.

Figure 3–2 Replacement Characters in Character Set Conversion

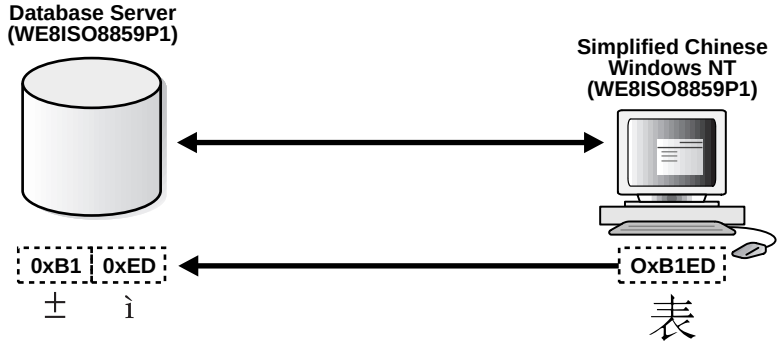


To reduce the risk of losing data, consider selecting a destination character set with similar character repertoires if possible. Migrating to Unicode can be an attractive option because UTF8 contains characters from most legacy character sets.

Another scenario that can cause the loss of data is migrating a database containing data of a different character set from that of the database character set. Users can insert data into the database from another character set if the client `NLS_LANG` character set setting is the same as the database character set. When these settings are the same, Oracle assumes that the data being sent or received is of the same character set, so no validations or conversions are performed.

This can lead to two possible data inconsistency problems. One is when a database contains data from another character set but the same codepoints exist in both character sets. For example, if the database character set is WE8ISO8859P1 and the end user Chinese Windows NT client's `NLS_LANG` setting is `SIMPLIFIED CHINESE_CHINA.WE8ISO8859P1`, then all multi-byte Chinese data (from the ZHS16GBK character set) is stored as multiples of single-byte WE8ISO8859P1 data. This means that Oracle will treat these characters as single-byte WE8ISO8859P1 characters, hence all SQL string manipulation functions such as `SUBSTR` or `LENGTH` will be based on bytes rather than characters. All bytes constituting ZHS16GBK data are legal WE8ISO8859P1 codes. If such a database is migrated to another character set, for example, UTF8, character codes will be converted as if they were in WE8ISO8859P1. This way, each of the two bytes of a ZHS16GBK character will be converted separately, yielding garbage values in UTF8. Figure 3–3 shows an example of this incorrect character set replacement.

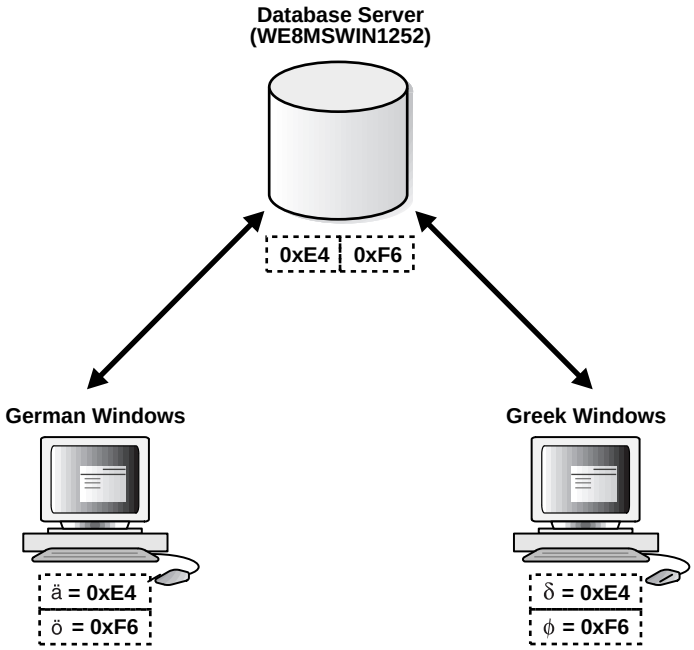
Figure 3–3 Incorrect Character Set Replacement



The second possibility is having data from mixed character sets inside the database. For example, if the data character set is WE8MSWIN1252, and two separate Windows clients using German and Greek are both using the `NLS_LANG` character set setting as WE8MSWIN1252, then the database will contain a mixture of German

and Greek characters. Figure 3–4 shows how different clients can use different character sets in the same database.

Figure 3–4 Mixed Character Sets



For database character set migration to be successful, both of these cases require manual intervention because Oracle cannot determine the character sets of the data being stored.

Database Character Set Migration

Database character set migration has two distinct stages:

- Data Scanning
- Conversion of Data

Data Scanning

Before you actually migrate your character set, you need to identify areas of possible database character set conversions and truncation of data. This step is called data scanning.

Data scanning identifies the amount of effort required to migrate data into the new character encoding scheme prior to the change of the database character set. Some examples of what are found during a data scan are the number of schema objects where the column widths need to be expanded and the extent of the data that does not exist in the target repertoire. This information will assist in determining the best approach for the conversion of the database character set.

Conversion of Data

There are generally three approaches in migrating data from one database character set to another, provided the database does not contain any of the inconsistencies described in "Character Set Conversions" on page 3-4. A description of methods to migrate databases with such inconsistencies is out of the scope of this document. For more information, contact Oracle Consulting Services for assistance.

In most cases, a full export or import is recommended to properly convert all data to a new character set. It is important to be aware of data truncation issues because character data type columns might need to be extended prior to import to handle the increase in size required. Existing PL/SQL code should be reviewed to ensure all byte-based SQL functions such as `LENGTHB`, `SUBSTRB`, and `INSTRB`, and PL/SQL `CHAR` and `VARCHAR2` declarations are still valid. However, if, and only if, the new character set is a strict superset of the current character set, you can use the `ALTER DATABASE CHARACTER SET` statement to expedite migration to a new database character set. The target character set is a strict superset if, and only if, each and every codepoint in the source character set is available in the target character set with the same corresponding codepoint value. For instance, because US7ASCII is a strict subset of UTF8, then an `ALTER DATABASE CHARACTER SET` statement can be used to upgrade the database character set from US7ASCII to UTF8. Refer to "Subsets and Supersets" on page 3-45 for a listing of all superset character sets in release 8.1.7.

The syntax is:

```
ALTER DATABASE [<db_name>] CHARACTER SET <new_character_set>;
ALTER DATABASE [<db_name>] NATIONAL CHARACTER SET <new_NCHAR_character_set>;
```

The database name is optional. The character set name should be specified without quotes. For example:

```
ALTER DATABASE CHARACTER SET UTF8;
```

To change the database character set, perform the following step:

```
SHUTDOWN IMMEDIATE;   -- or NORMAL
<do a full backup>
STARTUP MOUNT;
ALTER SYSTEM ENABLE RESTRICTED SESSION;
ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0;
ALTER SYSTEM SET IAQ_TM_PROCESSES=0;
ALTER DATABASE OPEN;
ALTER DATABASE CHARACTER SET <new_character_set_name>;
SHUTDOWN IMMEDIATE;   -- or NORMAL
STARTUP;
```

To change the national character set, replace the `ALTER DATABASE CHARACTER SET` statement with the `ALTER DATABASE NATIONAL CHARACTER SET` statement. You can issue both statements together if desired.

> **See Also:** *Oracle8i National Language Support Guide* and *Oracle8i SQL Reference* for the syntax of the `ALTER DATABASE [NATIONAL] CHARACTER SET` statement

When using Oracle Parallel Server (OPS), make sure no other Oracle background processes are running, with the exception of the one session through which a user is connected before attempting to issue the `ALTER DATABASE CHARACTER SET` statement. Use the following SQL statement to verify your environment:

```
SELECT SID, SERIAL#, PROGRAM FROM V$SESSION;
```

Setting the initialization parameter `PARALLEL_SERVER` to `FALSE` allows the character set change to go through. This is required in an OPS environment, as an exclusive startup is not sufficient.

> **Note:** It is essential to do a full backup of the database before using the `ALTER DATABASE [NATIONAL] CHARACTER SET` statement because the command cannot be rolled back.

The last approach is to perform an `ALTER DATABASE CHARACTER SET` statement followed by selective imports. This method is best suited for when the distributions

of convertible data are known and they are stored within a small number of tables. A full export and import will be too expensive in this scenario. For example, a 100GB database with over 300 tables but only 3 tables requires character set conversions and the rest of the data is of the same encoding as the destination character set. The 3 tables can be exported and imported back to the new database after issuing the ALTER DATABASE CHARACTER SET statement.

Incorrect data conversion can lead to the corruption of your data, so a full backup of your database must be performed before attempting to migrate your data to a new character set.

What is the Character Set Scanner Utility?

The Character Set Scanner provides an assessment of the feasibility and potential issues in migrating an Oracle database to a new database character set. The Scanner checks all character data in the database and tests for the effects and problems of changing the character set encoding. At the end of the scan, it generates a summary report of the database scan. This report provides estimates of the amount of work required to convert the database to a new character set.

Based on the information in the summary report, you will be able to decide on the most appropriate method to migrate the database's character set. The methods are:

- Export and Import Utilities
- ALTER DATABASE CHARACTER SET statement
- ALTER DATABASE CHARACTER SET with selective Export and Import

Note: If there are conversion exceptions reported by the Scanner, these problems must be fixed first before using any of the above methods to do the conversions. This may involve modifying the problem data to eliminate those exceptions. In extreme cases, both database and application might need to be modified. Oracle recommends you contact Oracle Consulting Services for services on database character set migration.

Conversion Tests on Character Data

The Scanner reads the character data and tests for the following conditions on each data cell:

- Do character codes of the data cells change when converted to the new character set?
- Can the data cells be successfully converted to the new character set?
- Will the post-conversion data fit into the current column size?

The Scanner reads and tests for data in CHAR, VARCHAR2, LONG, CLOB, NCHAR, NVARCHAR2, and NCLOB columns only. The Scanner does not perform post-conversion column size testing for LONG, CLOB, and NCLOB columns.

Access Privileges

To use the Scanner, you must have DBA privileges on the Oracle database.

Restrictions

All the character-based data in CHAR, VARCHAR2, LONG, and CLOB columns is stored in the same character set, which is the database character set specified with the CREATE DATABASE statement when the database was first created. However, in some configurations, it is possible to store data in a different character set from the database character set either intentionally or unintentionally. This happens most often when the NLS_LANG character set is the same as the database character set, because in such cases Oracle sends and receives data as is, without any conversion or validation. But it can also happen if one of the two character sets is a superset of the other, in which case many of the codes appear as if they were not converted. For example, if NLS_LANG is set to WE8ISO8859P1 and the database character set is WE8MSWIN1252, all codes except the range 128-159 are preserved through the client/server conversion.

Although a database that contains data not in its database character set cannot be converted to another character set by the three methods described in "What is the Character Set Scanner Utility?" on page 3-11, you can still use the Scanner in the way described below to test the effect of the conversion that would take place if the data was in the database character set.

Database Containing Data from Two or More Character Sets

If a database contains data from more than one character set, the Scanner cannot accurately test the effects of changing the database character set on the database because it cannot differentiate character sets properly. If the data can be divided into two separate tables, one for each language, then the Scanner can perform two single table scans to verify the validity of the data.

For each scan, a different value of the FROMCHAR parameter can be used to tell the Scanner to treat all CHAR, VARCHAR2, LONG, and CLOB columns in the table as if they were in the specified character set.

Database Containing Data not from the Database Character Set

If a database contains data not in the database character set, but still in only one character set, the Scanner can perform a full database scan. Use the FROMCHAR parameter to tell the Scanner what character set the data is in.

Scan Modes in the Scanner

The Character Set Scanner provides three modes of database scan:

- Full Database Scan
- User Tables Scan
- Single Table Scan

Full Database Scan

The Scanner reads and verifies the character data of all tables belonging to all users in the database including the data dictionary (SYS user), and it reports on the effects of the simulated migration to the new database character set. It scans all schema objects including stored packages, procedures and functions, and object names.

To understand the feasibility of migration to a new database character set, you need to perform a full database scan.

User Tables Scan

The Scanner reads and verifies character data of all tables belonging to the specified user and reports on the effects of changing the character set on them.

The Scanner does not test for table definitions such as table names and column names. To see the effects on the schema definitions, you need to perform a full database scan.

Single Table Scan

The Scanner reads and verifies the character data of the specified table, and reports the effects on changing the character set of them.

The Scanner does not test for table definitions such as table name and column name. To see the effects on the schema definitions, you need to perform a full database scan.

Using The Scanner

This section describes how to use the Scanner, including the steps you need to perform before scanning and the procedures on how to invoke the Scanner. The topics discussed are:

- Before Using the Scanner
- Compatibility
- Invoking the Scanner
- Getting Online Help
- The Parameter File

Before Using the Scanner

To use the Scanner, you must run the script `CSMINST.SQL` on the database that you plan to scan. `CSMINST.SQL` needs to be run only once, so it is not necessary to run it each time you scan the database. The script performs the following tasks to prepare the database for scanning:

- Creates a user named `CSMIG`
- Assigns the necessary privileges to `CSMIG`
- Assigns the default tablespace to `CSMIG`
- Connects as `CSMIG`
- Creates the Scanner system tables under `CSMIG`

The `SYSTEM` tablespace is assigned to `CSMIG` by default, so you need to ensure there is sufficient storage space available in the `SYSTEM` tablespace before scanning the database. The amount of space required depends on the type of scan and the nature of the data in the database. For information on storage considerations, refer to "Storage and Performance Considerations in the Scanner" on page 3-34.

You can modify the default tablespace for `CSMIG` by editing the script `CSMINST.SQL`. Modify the following statement in `CSMINST.SQL` to assign your preferred tablespace to `CSMIG` as follows:

```
SQL> ALTER USER CSMIG default tablespace PREFERRED_TABLESPACE;
```

Then run CSMINST.SQL using this command string:

```
% cd $ORACLE_HOME/rdbms/admin
% sqlplus
SQL> CONNECT system/manager
SQL> START csminst.sql
```

Compatibility

The Scanner is certified with Oracle databases on any platforms running under the same release except you cannot mix ASCII- and EBCDIC-based platforms. For example, the release 8.1.7 versions of the Scanner on any ASCII-based client platforms are certified to run with any release 8.1.7 Oracle databases on any ASCII-based platforms, while EBCDIC-based clients are certified to run with any release 8.1.7 Oracle database on EBCDIC platforms.

Oracle recommends you run the Scanner tool in the same Oracle Home as the database when possible.

Invoking the Scanner

You can invoke the Scanner by one of three methods:

4. Using the parameter file

   ```
   csscan system/manager PARFILE=filename
   ```

   PARFILE is a file containing the Scanner parameters you typically use.

5. Using the command line

   ```
   csscan system/manager full=y tochar=utf8 array=10240 process=3
   ```

6. Using an interactive session

   ```
   csscan system/manager
   ```

   In an interactive session, the Scanner prompts you for only the following parameters:

   - USERID
   - FULL
   - USER

- TABLE
- TOCHAR
- ARRAY
- PROCESS

If you want to specify parameters that are not listed above, you need to invoke the Scanner using either the parameter file or the command line.

Getting Online Help

The Scanner provides online help. Enter `csscan help=y` on the command line to invoke the help screen.

```
Character Set Scanner: Release 8.1.7.0.0 - Production

(c) Copyright 2000 Oracle Corporation. All rights reserved.


You can let Scanner prompt you for parameters by entering the CSSCAN command
followed by your username/password:
Example: CSSCAN SYSTEM/MANAGER

Or, you can control how Scanner runs by entering the CSSCAN command followed by
various parameters. To specify parameters, you use keywords:
Example: CSSCAN SYSTEM/MANAGER FULL=y TOCHAR=utf8 ARRAY=102400 PROCESS=3

Keyword    Default Prompt Description
---------- ------- ------ ------------------------------------------------
USERID             yes    username/password
FULL       N       yes    scan entire database
USER               yes    user name of the table to scan
TABLE              yes    table name to scan
TOCHAR             yes    new database character set name
FROMCHAR                  current database character set name
TONCHAR                   new NCHAR character set name
FROMNCHAR                 current NCHAR character set name
ARRAY      10240   yes    size of array fetch buffer
PROCESS    1       yes    number of scan process
MAXBLOCKS                 split table if larger than MAXBLOCKS
CAPTURE     N             capture convertible data
SUPPRESS                  suppress error log by N per table
FEEDBACK                  feedback progress every N rows
BOUNDARIES                list of column size boundaries for summary report
```

```
LASTRPT    N              generate report of the last database scan
LOG        scan           base name of log files
PARFILE                   parameter file name
HELP       N              show help screen (this screen)
---------- ------- ------ ------------------------------------------------
Scanner terminated successfully.
```

The Parameter File

The parameter file allows you to specify Scanner parameters in a file where they can be easily modified or reused. Create a parameter file using any flat file text editor. The command line option `PARFILE=filename` tells the Scanner to read the parameters from a specified file rather than from the command line. For example:

```
csscan parfile=filename
```

or

```
csscan username/password parfile=filename
```

The syntax for parameter file specifications is one of the following:

```
KEYWORD=value
KEYWORD=(value1, value2, ...)
```

The following is an example of a parameter file:

```
USERID=system/manager
USER=SCOTT    # scan SCOTT's tables
TOCHAR=utf8
ARRAY=40960
PROCESS=2  # use two concurrent scan processes
FEEDBACK=1000
```

You can add comments to the parameter file by preceding them with the pound (#) sign. All characters to the right of the pound sign are ignored.

Scanner Parameters

This section describes each of the Scanner parameters.

ARRAY

Default value: 10240

Minimum value: 4096

Maximum value: unlimited

Purpose: Specifies the size in bytes of the array buffer used to fetch data. The size of the array buffer determines the number of rows fetched by the Scanner at any one time.

The formula below gives an approximation of number of rows fetched at a time:

```
(rows in array) = (ARRAY buffer size) / (sum of the CHAR and VARCHAR2
    column sizes of a given table)
```

If the summation of CHAR and VARCHAR2 column sizes exceeds the array buffer size, the Scanner fetches only one row at a time. Tables with LONG, CLOB, or NCLOB columns are fetched only one row at a time.

This parameter affects the duration of a database scan. In general, the larger the size of the array buffer, the shorter the duration time. Each scan process will allocate the specified size of array buffer.

BOUNDARIES

Default value: none

Purpose: Specifies the list of column boundary sizes that are used for an application data conversion summary report. This parameter is used to locate the distribution of the application data for the datatypes CHAR, VARCHAR2, NCHAR, and NVARCHAR2.

For example, if you specify a BOUNDARIES value of (10, 100, 1000), the application data conversion summary report will produce a breakdown of the CHAR data into the following groups by their column length, CHAR(1..10), CHAR(11..100) and CHAR(101..1000), likewise for the VARCHAR2, NCHAR, and NVARCHAR2 datatypes.

CAPTURE

Default value: N

Range of values: Y or N

Purpose: Indicates whether to capture the information on the individual convertible rows as well as the default of storing the exception rows. The convertible rows information is written to the table CSM$ERRORS if the parameter CAPTURE is set to Y. This information can be used to deduce which records need to be converted to the target character set by selective export and import.

FEEDBACK

Default value: none

Minimum value: 100

Maximum value: 100000

Purpose: Specifies that the Scanner should display a progress meter in the form of a dot for every N number of rows scanned.

For example, if you specify FEEDBACK=1000, the Scanner displays a dot for every 1000 rows scanned. The FEEDBACK value applies to all tables being scanned, so it cannot be set on a per-table basis.

FROMCHAR

Default value: none

Purpose: Specifies the actual character set name for CHAR, VARCHAR2, LONG, and CLOB data types in the database. By default, the Scanner assumes the character set for the above data types to be the database character set.

Use this parameter to override the default database character set definition for CHAR, VARCHAR2, LONG, and CLOB data in the database.

FROMNCHAR

Default value: none

Purpose: Specifies the actual national database character set name for NCHAR, NVARCHAR2, and NCLOB data types in the database. By default, the Scanner assumes the character set for the above data types to be the database national character set.

Use this parameter to override the default database character set definition for NCHAR, NVARCHAR2, and NCLOB data in the database.

FULL

Default value: N

Range of values: Y or N

Purpose: Indicates whether to perform the full database scan (that is, to scan the entire database including the data dictionary). Specify FULL=Y to scan in full database mode.

For more information on full database scans, refer to "Scan Modes in the Scanner" on page 3-11.

HELP

Default value: N

Range of values: Y or N

Purpose: Displays a help message with descriptions of the Scanner parameters.

For more information, see "Getting Online Help" on page 3-14.

LASTRPT

Default value: N

Range of values: Y or N

Purpose: Indicates whether to regenerate the Scanner reports based on statistics gathered from the last database scan.

If `LASTRPT=Y` is specified, the Scanner does not scan the database, but creates the report files using the information left by the previous database scan session instead.

If `LASTRPT=Y` is specified, only the `USERID`, `BOUNDARIES`, and `LOG` parameters take effect.

LOG

Default value:	scan
Purpose:	Specifies a base file name for the following Scanner report files: Database Scan Summary Report file whose extension is `.txt` Individual Exception Report file whose extension is `.err` Screen log file whose extension is `.out`

By default, the Scanner generates the three text files, `scan.txt`, `scan.err`, and `scan.out` in the current directory.

MAXBLOCKS

Default value:	none
Minimum value:	1000
Maximum value:	unlimited
Purpose:	Specifies the maximum block size per table, so that large tables can be split into smaller chunks for the Scanner to process.

For example, if the `MAXBLOCKS` parameter is set to 1000, then any tables that are greater than 1000 blocks in size will be divided into n chunks, where `n=CEIL(table block size/1000)`.

Dividing large tables into smaller pieces will be beneficial only when the number of processes set with `PROCESS` is greater than 1. If the `MAXBLOCKS` parameter is not set, the Scanner attempts to split up large tables based on the its own optimization rules.

PARFILE

Default value: none

Purpose: Specifies a filename for a file that contains a list of Scanner parameters.

For more information on using a parameter file, see "The Parameter File" on page 3-15.

PROCESS

Default value: 1

Minimum value: 1

Maximum value: 32

Purpose: Specifies the number of concurrent scan processes to utilize for the database scan.

SUPPRESS

Default value: unlimited

Minimum value: 0

Maximum value: unlimited

Purpose: Specifies the maximum number of data exceptions being logged per table.

The Scanner inserts individual exceptional record information into the `CSM$ERRORS` table when an exception is found in a data cell. The table grows depending on the number of exceptions reported.

This parameter is used to suppress the logging of individual exception information after a specified number of exceptions are inserted per table. For example, if `SUPPRESS` is set to 100, then the Scanner records a maximum of 100 exception records per table.

For more information, see "Storage Considerations" on page 3-34.

TABLE

Default value: none

Purpose: Specifies the name of the table to scan.

When specified, Scanner scans the specified table only. For example, the command below scans an emp table that belongs to the user scott:

```
csscan system/manager USER=SCOTT TABLE=EMP ...
```

TOCHAR

Default value: none

Purpose: Specifies a target database character set name for the CHAR, VARCHAR2, LONG, and CLOB data.

TONCHAR

Default value: none

Purpose: Specifies a target database character set name for the NCHAR, NVARCHAR2, and NCLOB data.

If you do not specify a value for TONCHAR, the Scanner does not scan NCHAR, NVARCHAR2, and NCLOB data.

USER

Default value: none

Purpose: Specifies the owner of the tables to be scanned.

If the parameter USER is specified, the Scanner scans all tables belonging to the user. If TABLE is specified, the Scanner scans only the table specified by TABLE that belongs to the user. For example, the following statement scans all tables belonging to the user scott:

```
csscan system/manager USER=scott ...
```

USERID

Default value: none

Purpose: Specifies the username/password (and optional connect string) of the user who scans the database. If you omit the password, the Scanner prompts you for it.

The following examples are all valid:

```
username/password
username/password@connect_string
username
username@connect_string
```

Sample Scanner Sessions

The following examples show you how to use the command line and parameter file methods to use Full Database, User Tables, and Single Table scan modes.

Sample Session of Full Database Scan

The following example shows how to scan the full database to see the effects on migrating it to UTF8. This example assumes the current database character set is WE8ISO8859P1 (or anything other than UTF8).

Parameter File Method

```
% csscan system/manager parfile=param.txt
```

The `param.txt` file contains the following information:

```
full=y
tochar=utf8
array=40960
process=4
```

Command Line Method

```
% csscan system/manager full=y tochar=utf8 array=40960 process=4

Scanner Messages
Database Scanner: Release 8.1.7.0.0 - Production

(c) Copyright 2000 Oracle Corporation. All rights reserved.
```

```
Connected to:
Oracle8 Enterprise Edition Release 8.1.7.0.0 - Production
With the Objects option
PL/SQL Release 8.1.7.0.0 - Production

Enumerating tables to scan...

. process 1 scanning SYSTEM.REPCAT$_RESOLUTION
. process 1 scanning SYS.AQ$_MESSAGE_TYPES
. process 1 scanning SYS.ARGUMENT$
. process 2 scanning SYS.AUD$
. process 3 scanning SYS.ATTRIBUTE$
. process 4 scanning SYS.ATTRCOL$
. process 2 scanning SYS.AUDIT_ACTIONS
. process 2 scanning SYS.BOOTSTRAP$
. process 2 scanning SYS.CCOL$
. process 2 scanning SYS.CDEF$
 :
 :
. process 3 scanning SYSTEM.REPCAT$_REPOBJECT
. process 1 scanning SYSTEM.REPCAT$_REPPROP
. process 2 scanning SYSTEM.REPCAT$_REPSCHEMA
. process 3 scanning MDSYS.MD$DIM
. process 1 scanning MDSYS.MD$DICTVER
. process 2 scanning MDSYS.MD$EXC
. process 3 scanning MDSYS.MD$LER
. process 1 scanning MDSYS.MD$PTAB
. process 2 scanning MDSYS.MD$PTS
. process 3 scanning MDSYS.MD$TAB

Creating Database Scan Summary Report...

Creating Individual Exception Report...

Scanner terminated successfully.
```

Sample Session of User Tables Scan

The following example shows how to scan the user tables to see the effects on migrating them to UTF8. This example assumes the current database character set is US7ASCII, but the actual data stored is in Western European WE8MSWIN1252 encoding.

Parameter File Method

```
% csscan system/manager parfile=param.txt
```

The `param.txt` file contains the following information:

```
user=scott
fromchar=we8mswin1252
tochar=utf8
array=40960
process=1
```

Command Line Method

```
% csscan system/manager user=scott fromchar=we8mswin1252 tochar=utf8 array=40960
process=1
```

Scanner Messages

```
Database Scanner: Release 8.1.7.0.0 - Production

(c) Copyright 2000 Oracle Corporation. All rights reserved.

Connected to:
Oracle8 Enterprise Edition Release 8.1.7.0.0 - Production
With the Objects option
PL/SQL Release 8.1.7.0.0 - Production

Enumerating tables to scan...

. process 1 scanning SCOTT.BONUS
. process 1 scanning SCOTT.DEPT
. process 1 scanning SCOTT.EMP

Creating Database Scan Summary Report...

Creating Individual Exception Report...

Scanner terminated successfully.
```

Sample Session of Single Table Scan

The following example shows how to scan a single table to see the effects on migrating it to WE8MSWIN1252. This example assumes the current database character set is in US7ASCII.

Parameter File Method

```
% csscan system/manager parfile=param.txt
```

The `param.txt` file contains the following information:

```
user=scott
table=emp
tochar=we8mswin1252
array=40960
process=1
supress=100
```

Command Line Method

```
% csscan system/manager user=scott table=emp tochar=we8mswin1252 array=40960
process=1 supress=100

Scanner Messages
Database Scanner: Release 8.1.7.0.0 - Production

(c) Copyright 1999 Oracle Corporation. All rights reserved.

Connected to:
Oracle8 Enterprise Edition Release 8.1.7.0.0 - Production
With the Objects option
PL/SQL Release 8.1.7.0.0 - Production

. process 1 scanning SCOTT.EMP

Creating Database Scan Summary Report...

Creating Individual Exception Report...

Scanner terminated successfully.
```

Scanner Reports

The Scanner generates two reports per scan:

- Database Scan Summary Report
- Individual Exception Report

Database Scan Summary Report

A Database Scan Summary Report consists of the following sections. The information available for each section depends on the type of scans and the parameters you select.

- Database Scanner Parameters
- Database Size
- Scan Summary
- Data Dictionary Conversion Summary
- Application Data Conversion Summary
- Application Data Conversion Summary per Column Size Boundary
- Distribution of Convertible Data per Table
- Distribution of Convertible Data per Column
- Indexes To Be Rebuilt

Database Scanner Parameters

This section describes the parameters selected and the type of scan you chose. The following is an example:

```
Parameter                                Value
---------------------------------------- ------------------------------
Scan type                                Full database
Scan CHAR data?                          YES
Current database character set           WE8ISO8859P1
New database character set               UTF8
Scan NCHAR data?                         NO
Array fetch buffer size                  102400
Number of processes                      4
---------------------------------------- ------------------------------
```

Database Size

This section describes the current database size. The following is an example:

```
TABLESPACE                        Total(MB)        Used(MB)        Free(MB)
----------------------------- --------------- --------------- ---------------
APPS_DATA                         1,340.000       1,331.070           8.926
CTX_DATA                             30.000           3.145          26.852
INDEX_DATA                          140.000         132.559           7.438
```

```
RBS_DATA                              310.000        300.434          9.563
SYSTEM_DATA                           150.000        144.969          5.027
TEMP_DATA                             160.000                        159.996
TOOLS_DATA                             35.000         22.148         12.848
USERS_DATA                            220.000        142.195         77.801
---------------------------- --------------- --------------- ---------------
Total                               2,385.000      2,073.742        311.227
```

Scan Summary

This indicates the feasibility of the database character set migration. There are two basic criteria that determine the feasibility of the character set migration of the database. One is the condition of the data dictionary and the other is the condition of the application data.

The Scan Summary section consists of two status lines. Depending on the scan mode and the result returned, the following statuses are printed:

For the data dictionary

- All character-type data in the data dictionary remains the same in the new character set
- All character-type data in the data dictionary is convertible to the new character set
- Some character-type data in the data dictionary is not convertible to the new character set

For application data

- All character-type application data remains the same in the new character set
- All character-type application data is convertible to the new character set
- Some character-type application data is not convertible to the new character set

When all data remains the same in the new character set, it means that the data encoding of the original character set is identical to the target character set. In this case, the character set can be migrated using the ALTER DATABASE CHARACTER SET statement.

If all the data is convertible to the new character set, it means that the data can be safely migrated using the Export and Import utilities. However, the migrated data may or may not have the same encoding as the original character set.

> **See Also:** "Individual Exception Report" on page 3-32 for more information on non-convertible data

The following is sample output:

```
All character type data in the data dictionary remains the same in the new
character set
All character type application data remains the same in the new character set
```

Data Dictionary Conversion Summary

This section contains the statistics on the conversion summary of the data dictionary. The granularity of this report is per datatype. The following statuses are available:

Table 3–1 Data Conversion Summary for Data Dictionary

Status	Description
`Changeless`	Number of data cells that remain the same in the new character set
`Convertible`	Number of data cells that will be successfully converted to the new character set
`Exceptional`	Number of data cells that cannot be converted. If you choose to convert anyway, some characters will be lost or data will be truncated

If the numbers in both the `Convertible` and `Exceptional` columns are zero, it means that all the data in the data dictionary will remain the same in the new character set.

If the numbers in the `Exceptional` column are zero and some numbers in the `Convertible` columns are non-zero, it means all data in the data dictionary is convertible to the new character set. During import, the relevant data will be converted.

If the numbers in the `Exceptional` column are non-zero, it means there is data in the data dictionary that is not convertible. Therefore, it is not feasible to migrate the current database to the new character because the export and import process cannot convert the data into the new character set. For example, you might have a table name with invalid characters or a PL/SQL procedure where a comment line includes data that can not be mapped to the new character set. These changes to schema objects must be corrected manually prior to migration to a new character set.

This information is available only when a full database scan is performed. The following is an example:

```
Datatype        Changeless      Convertible      Exceptional            Total
--------- ---------------- ---------------- ---------------- ----------------
VARCHAR2           971,300                1                0          971,301
CHAR                     7                0                0                7
LONG                60,325                0                0           60,325
CLOB
--------- ---------------- ---------------- ---------------- ----------------
Total            1,031,632                1                0        1,031,633
```

Application Data Conversion Summary

This section contains the statistics on conversion summary of the application data. The granularity of this report is per datatype. The following statuses are available:

Table 3–2 Data Conversion Summary for Application Data

Status	Description
`Changeless`	Number of data cells that remain the same in the new character set
`Convertible`	Number of data cells that will be successfully converted to the new character set
`Exceptional`	Number of data cells that cannot be converted. If you choose to convert anyway, some characters will be lost or data will be truncated

The following is sample output:

```
Datatype         Changeless      Convertible      Exceptional            Total
---------- ---------------- ---------------- ---------------- ----------------
VARCHAR2         23,213,745            1,324                0       23,215,069
CHAR                423,430                0                0          423,430
LONG                  8,624               33                0            8,657
CLOB                 58,839           11,114               28           69,981
---------- ---------------- ---------------- ---------------- ----------------
Total            23,704,638           12,471               28       23,717,137
```

Application Data Conversion Summary per Column Size Boundary

This section contains the conversion summary of the `CHAR` and `VARCHAR2` application data. The granularity of this report is per column size boundaries specified by the `BOUNDARIES` parameter. The following status is available for each datatype and each boundary:

The granularity of this report is per datatype. The following statuses are available:

Table 3–3 Data Conversion Summary for Columns in Application Data

Status	Description
`Changeless`	Number of data cells that remain the same in the new character set
`Convertible`	Number of data cells that will be successfully converted to the new character set
`Exceptional`	Number of data cells that cannot be converted. If you choose to convert, some characters will be lost or data will be truncated

This information is available only when the `BOUNDARIES` parameter is specified.

The following is sample output:

```
Datatype                Changeless   Convertible     Exceptional           Total
------------------   ------------- ------------- --------------- ----------------
VARCHAR2(1..10)          1,474,825             0               0       1,474,825
VARCHAR2(11..100)        9,691,520            71               0       9,691,591
VARCHAR2(101..4000)     12,047,400         1,253               0      12,048,653
------------------   ------------- ------------- --------------- ----------------
CHAR(1..10)                423,413             0               0         423,413
CHAR(11..100)                   17             0               0              17
CHAR(101..4000)
------------------   ------------- ------------- --------------- ----------------
Total                   23,637,175         1,324               0      23,638,499
```

Distribution of Convertible Data per Table

This example show how `Convertible` and `Exceptional` data is distributed within the database. The granularity of this report is per table. If the list contains only a few rows, it means the `Convertible` data is localized. If the list contains many rows, it means the `Convertible` data is spread out in the database.

The following is sample output:

```
USER.TABLE                                          Convertible      Exceptional
------------------------------------------------ --------------- ----------------
SMG.SOURCE                                                     1                0
SMG.HELP                                                      12                0
SMG.CLOSE_LIST                                                16                0
SMG.ATTENDEES                                                  8                0
SGT.DR_010_I1T1                                                7                0
SGT.DR_011_I1T1                                                7                0
SGT.MRK_SRV_PROFILE                                            2                0
```

```
SGT.MRK_SRV_PROFILE_TEMP                                  2                0
SGT.MRK_SRV_QUESTION                                      3                0
---------------------------------------------- ---------------- ----------------
```

Distribution of Convertible Data per Column

This example shows how `Convertible` and `Exceptional` data is distributed within the database. The granularity of this report is per column. The following is an example:

```
USER.TABLE|COLUMN                                    Convertible      Exceptional
---------------------------------------------- ---------------- ----------------
SMG.SOURCE|SOURCE                                            1                0
SMG.HELP|INFO                                               12                0
SMG.CLOSE_LIST|FNAME                                         1                0
SMG.CLOSE_LIST|LNAME                                         1                0
SMG.CLOSE_LIST|COMPANY                                       1                0
SMG.CLOSE_LIST|STREET                                        8                0
SMG.CLOSE_LIST|CITY                                          4                0
SMG.CLOSE_LIST|STATE                                         1                0
SMG.ATTENDEES|ATTENDEE_NAME                                  1                0
SMG.ATTENDEES|ADDRESS1                                       3                0
SMG.ATTENDEES|ADDRESS2                                       2                0
SMG.ATTENDEES|ADDRESS3                                       2                0
SGT.DR_010_I1T1|WORD_TEXT                                    7                0
SGT.DR_011_I1T1|WORD_TEXT                                    7                0
SGT.MRK_SRV_PROFILE|FNAME                                    1                0
SGT.MRK_SRV_PROFILE|LNAME                                    1                0
SGT.MRK_SRV_PROFILE_TEMP|FNAME                               1                0
SGT.MRK_SRV_PROFILE_TEMP|LNAME                               1                0
SGT.MRK_SRV_QUESTION|ANSWER                                  3                0
---------------------------------------------- ---------------- ----------------
```

Indexes To Be Rebuilt

This generates a list of all the indexes that are affected by the database character set migration. These can be rebuilt upon the import of the data. The following is an example:

```
USER.INDEX on USER.TABLE(COLUMN)
--------------------------------------------------------------------------------
CD2000.COMPANY_IX_PID_BID_NNAME on CD2000.COMPANY(CO_NLS_NAME)
CD2000.I_MASHINE_MAINT_CONT on CD2000.MACHINE(MA_MAINT_CONT#)
CD2000.PERSON_NEWS_SABUN_CONT_CONT on
CD2000.PERSON_NEWS_SABUN_CONT(CONT_BID)
CD2000.PENEWSABUN3_PEID_CONT on CD2000.PE_NEWS_SABUN_3(CONT_BID)
PMS2000.CALLS_IX_STATUS_SUPPMGR on PMS2000.CALLS(SUPPMGR)
PMS2000.MAILQUEUE_CHK_SUB_TOM on PMS2000.MAIL_QUEUE(TO_MAIL)
PMS2000.MAILQUEUE_CHK_SUB_TOM on PMS2000.MAIL_QUEUE(SUBJECT)
```

```
PMS2000.TMP_IX_COMP on PMS2000.TMP_CHK_COMP(COMP_NAME)
-----------------------------------------------------------------------
```

Individual Exception Report

An Individual Exception Report consists of the following summaries:

- Database Scan Parameters
- Application Data Individual Exceptions

Database Scan Parameters

This section describes the parameters and the type of scan chosen. The following is an example:

```
Parameter                                Value
---------------------------------------- ------------------------------
Scan type                                Full database
Scan CHAR data?                          YES
Current database character set           we8mswin1252
New database character set               utf8
Scan NCHAR data?                         NO
Array fetch buffer size                  102400
Number of rows to heap up for insert     10
Number of processes                      1
---------------------------------------- ------------------------------
```

Application Data Individual Exceptions

This report identifies the data that has exceptions so that this data can then be modified if necessary.

There are two types of exceptions:

- Exceed Column Size

  The column size should be extended if the maximum column width has been surpassed. If not, data truncation occurs.

- Lossy Conversion

  The data must be corrected before migrating to the new character set, or else the invalid characters will be converted to a replacement character. Replacement characters are usually specified as '?' or '¿' or a similar linguistically-related character.

The following is an example of an individual exception report that illustrates some possible problems when changing the database character set from WE8ISO8859P1 to UTF8:

```
User:   SCOTT
Table:  PRODUCT
Column: NAME
Type:   VARCHAR2(10)
Number of Exceptions: 2
Max Post Conversion Data Size: 11

ROWID              Exception Type      Size Cell Data(first 30 bytes)
------------------ ------------------ ----- ------------------------------
AAAA2fAAFAABJwQAAg exceed column size    11 Ährenfeldt
AAAA2fAAFAABJwQAAu lossy conversion         órâclë8™
AAAA2fAAFAABJwQAAu exceed column size    11 órâclë8™
------------------ ------------------ ----- ------------------------------
```

The values Ährenfeldt and órâclë8™ exceed the column size (10 bytes) because each of the characters Ä, ó, â, and ë occupies one byte in WE8ISO8859P1 but two bytes in UTF8. The value órâclë8™ has lossy conversion to UTF8 because the trademark sign ™ (code 153) is not a valid WE8ISO8859P1 character. It is a WE8MSWIN1252 character, which is a superset of WE8ISO8859P1.

You can view the data that has an exception by issuing a SELECT statement:

```
SELECT name FROM scott.product
WHERE ROWID='AAAA2fAAFAABJwQAAu';
```

You can modify the data that has the exception by issuing an UPDATE statement:

```
UPDATE scott.emp SET ename = 'Oracle8 TM'
WHERE ROWID='AAAA2fAAFAABJwQAAu';
```

Storage and Performance Considerations in the Scanner

Storage Considerations

This section describes the sizing and the growth of the Scanner's system tables, and explains the approach to maintain them. There are three system tables that can increase rapidly depending on the nature of the data stored in the database.

- CSM$TABLES
- CSM$COLUMNS
- CSM$ERRORS

CSM$TABLES

The Scanner enumerates all tables that need to be scanned into the table `CSM$TABLES`. Therefore, `CSM$TABLES` contains as many rows as the number of tables in the database.

You might want to assign a large tablespace to the user `CSMIG` by amending the `CSMINST.SQL` script. By default, the `SYSTEM` tablespace is assigned to the user `CSMIG`.

You can look up the number of tables in the database by issuing the following SQL statement:

```
SELECT COUNT(*) FROM DBA_TABLES;
```

CSM$COLUMNS

The Scanner stores statistical information for each column scanned into the table `CSM$COLUMNS`. Therefore, `CSM$COLUMNS` contains as many rows as the number of character-type columns in the database.

You might want to assign a large tablespace to the user `CSMIG` by amending the `CSMINST.SQL` script. By default, the `SYSTEM` tablespace is assigned to `CSMIG` user.

You can look up the number of character type columns in the database by issuing the following SQL statement:

```
SELECT COUNT(*) FROM DBA_TAB_COLUMNS
WHERE DATA_TYPE IN ('CHAR', 'VARCHAR2', 'LONG', 'CLOB');
```

CSM$ERRORS

When exceptions are detected with cell data, the Scanner inserts individual exception information into the table CSM$ERRORS. This information then appears in the Individual Exception Report and facilitates identifying records to be modified if necessary.

If your database contains a lot of data that is signaled as Exceptional or Convertible (when the parameter CAPTURE=Y is set), the table CSM$ERRORS can grow too large. You can prevent the CSM$ERRORS table from growing unnecessarily large by using the SUPPRESS parameter.

The SUPPRESS parameter applies to each table. The Scanner suppresses inserting individual Exceptional information after the specified number of exceptions is inserted. Limiting the number of exceptions to be recorded may not be useful if the exceptions are spread over different tables.

You might want to assign a large tablespace to the user CSMIG by amending the CSMINST.SQL script.

Performance Consideration

This section describes ways to increase performance when scanning the database.

Utilizing Multiple Scan Processes

If you plan to scan a relatively large database, for example, over 50GB, you might want to consider using multiple scan processes. This shortens the duration time of database scans by utilizing hardware resources such as CPU and memory available on the machine.

Array Fetch Buffer Size

The Scanner fetches multiple rows at a time when an array fetch is allowed. Generally, you will improve performance by letting the Scanner use a bigger array fetch buffer.

Suppressing Exception and Convertible Log

The Scanner inserts individual `Exceptional` and `Convertible` (when `CAPTURE=Y`) information into the table `CSM$ERRORS`. In general, insertion into the `CSM$ERRORS` table is more costly than data fetching. If your database has a lot of data that is signaled as `Exceptional` or `Convertible`, the Scanner issues many insert statements, causing performance degradation. Oracle recommends setting a limit on the number of exception rows to be recorded using the `SUPRESS` parameter.

Reference Material

This section contains the following reference material:

- Scanner Tables
- Scanner Messages
- Subsets and Supersets

Scanner Tables

The Scanner uses the following tables.

CSM$PARAMETERS

This table contains Scanner parameters specified by the user.

Column	Datatype	NULL	Description
NAME	VARCHAR2(30)	NOT NULL	Parameter name
VALUE	VARCHAR2(30)	NOT NULL	Parameter value

CSM$TABLES

This table contains information about database tables to be scanned. The Scanner enumerates all tables to be scanned into this table.

Column	Datatype	NULL	Description
USR#	NUMBER	NOT NULL	Userid of the table owner
OBJ#	NUMBER	NOT NULL	Object id of the table
MINROWID	ROWID		Minimum rowid of the split range of the table
MAXROWID	ROWID		Maximum rowid of the split range of the table

Column	Datatype	NULL	Description
PROPERTY	NUMBER		Table property
WHO	NUMBER		Internal id of the scan process that scanned the table
SCNSTART	DATE		Time table scan started
SCNEND	DATE		Time table scan completed
SCNCOLS	NUMBER		Number of columns (to be) scanned
SCNROWS	NUMBER		Number of rows scanned

CSM$COLUMNS

This table contains statistical information of columns that were scanned.

Column	Datatype	NULL	Description
USR#	NUMBER	NOT NULL	Userid of the table owner
OBJ#	NUMBER	NOT NULL	Object id of the table
COL#	NUMBER	NOT NULL	Column id
INTCOL#	NUMBER	NOT NULL	Internal column id (for ADT)
DTY#	NUMBER	NOT NULL	Column datatype
FRM#	NUMBER	NOT NULL	Character set form
NUMROWS	NUMBER	NOT NULL	Number of rows in this column / table
NULCNT	NUMBER	NOT NULL	Number of NULL data cells
CNVCNT	NUMBER	NOT NULL	Number of data cells that need to be converted
ERRCNT	NUMBER	NOT NULL	Number of data cells that have exceptions
SIZERR	NUMBER	NOT NULL	Number of data cells that exceed column's size
CNVERR	NUMBER	NOT NULL	Number of data cells that undergo lossy conversion
MAXSIZ	NUMBER	NOT NULL	Maximum post-conversion data size

CSM$ERRORS

This table contains individual exception information of cell data and object definitions.

Column	Datatype	NULL	Description
ERR#	NUMBER	NOT NULL	Scanner internal exception code
USR#	NUMBER	NOT NULL	Userid of the object owner or data owner
OBJ#	NUMBER	NOT NULL	Object id
COL#	NUMBER		Column id
INTCOL#	NUMBER		Internal column id (for ADT)
TYP#	NUMBER		Column datatype or object type
FRM#	NUMBER		Character set form
CNVSIZE	DATE		Post-conversion data size
ID$	DATE		The rowid of the data or name that identifies the object

Scanner Messages

The Scanner has the following error messages:

CSC-00100 failed to allocate memory size of number

Cause: An attempt was made to allocate memory with size 0 or bigger than the maximum size.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00101 failed to release memory

Cause: An attempt was made to release memory with invalid pointer.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00102 failed to release memory, null pointer given

Cause: An attempt was made to release memory with null pointer.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00105 failed to parse BOUNDARIES parameter

Cause: BOUNDARIES parameter was specified in an invalid format.

Action: Refer to the manual for the correct syntax.

CSC-00106 failed to parse SPLIT parameter

Cause: SPLIT parameter was specified in an invalid format.

Action: Refer to the manual for the correct syntax.

CSC-00107 CSM$* tables not found

Cause: CSM$VERSION table not found in the database.

Action: Run CSMINST.SQL on the database.

CSC-00108 incompatible CSM$* tables

Cause: Incompatible CSM$* tables found in the database.

Action: Run CSMINST.SQL on the database.

CSC-00110 failed to parse userid

Cause: USERID parameter was specified in an invalid format.

Action: Refer to the manual for the correct syntax.

CSC-00111 failed to get RDBMS version

Cause: Failed to retrieve the value of the Version of the database.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00112 database version not supported

Cause: The database version is older than release 8.0.5.0.0.

Action: Upgrade the database to release 8.0.5.0.0 or later, then try again.

CSC-00113 user %s is not allowed to access data dictionary

Cause: The specified user cannot access the data dictionary.

Action: Set O7_DICTIONARY_ACCESSIBILITY parameter to TRUE, or use SYS user.

CSC-00114 failed to get database character set name

Cause: Failed to retrieve value of NLS_CHARACTERSET or NLS_NCHAR_CHARACTERSET parameter from NLS_DATABASE_PARAMETERS view.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00115 invalid character set name %s

Cause: The specified character set is not a valid Oracle character set.

Action: Refer to the National Language Support Guide for the correct character set name.

CSC-00116 failed to reset NLS_LANG/NLS_NCHAR parameter

Cause: Failed to force `NLS_LANG` character set to be same as database character set.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00117 failed to clear previous scan log

Cause: Failed to delete all rows from `CSM$*` tables.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00118 failed to save command parameters

Cause: Failed to insert rows into `CSM$PARAMETERS` table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00119 failed to save scan start time

Cause: Failed to insert a row into `CSM$PARAMETERS` table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00120 failed to enumerate tables to scan

Cause: Failed to enumerate tables to scan into `CSM$TABLES` table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00121 failed to save scan complete time

Cause: Failed to insert a row into `CSM$PARAMETERS` table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00122 failed to create scan report

Cause: Failed to create database scan report.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00123 failed to check if user %s exist

Cause: Select statement that checks if the specified user exists in the database failed.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00124 user %s not found

Cause: The specified user does not exist in the database.

Action: Check the user name.

CSC-00125 failed to check if table %s.%s exist

Cause: Select statement that checks if the specified table exists in the database failed.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00126 table %s.%s not found

Cause: The specified table does not exist in the database.

Action: Check the user name and table name.

CSC-00127 user %s does not have DBA privilege

Cause: The specified user does not have DBA privileges, which are required to scan the database.

Action: Choose a user with DBA privileges.

CSC-00128 failed to get server version string

Cause: Failed to retrieve the version string of the database.

Action: None.

CSC-00130 failed to initialize semaphore

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00131 failed to spawn scan process %d

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00132 failed to destroy semaphore

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00133 failed to wait semaphore

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00134 failed to post semaphore

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00140 failed to scan table (tid=%d, oid=%d)

Cause: Data scan on this particular table failed.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00141 failed to save table scan start time

Cause: Failed to update a row in the `CSM$TABLES` table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00142 failed to get table information

Cause: Failed to retrieve various information from user id and object id of the table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00143 failed to get column attributes

Cause: Failed to retrieve column attributes of the table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00144 failed to scan table %s.%s

Cause: Data scan on this particular table was not successful.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00145 failed to save scan result for columns

Cause: Failed to insert rows into `CSM$COLUMNS` table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00146 failed to save scan result for table

Cause: Failed to update a row of `CSM$TABLES` table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00147 unexpected data truncation

Cause: Scanner allocates the exactly same size of memory as the column byte size for fetch buffer, resulting in unexpected data truncation.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00150 failed to retrieve table information

Cause: Failed to retrieve the specified table information.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00151 failed to enumerate user tables

Cause: Failed to enumerate all tables that belong to the specified user.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00152 failed to enumerate all tables

Cause: Failed to enumerate all tables in the database.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00153 failed to enumerate character type columns

Cause: Failed to enumerate all `CHAR`, `VARCHAR2`, `LONG`, and `CLOB` columns of tables to scan.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00154 failed to create list of tables to scan

Cause: Failed to enumerate the tables into `CSM$TABLES` table.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00155 failed to split tables for scan

Cause: Failed to split the specified tables.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00156 failed to get total number of tables to scan

Cause: Select statement that retrieves the number of tables to scan failed.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00157 failed to retrieve list of tables to scan

Cause: Failed to read all table ids into the scanner memory.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00158 failed to retrieve index defined on column

Cause: Select statement that retrieves index defined on the column fails.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00160 failed to open summary report file

Cause: File open function returned error.

Action: Check if you have create/write privilege on the disk and check if the file name specified for the `LOG` parameter is valid.

CSC-00161 failed to report scan elapsed time

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00162 failed to report database size information

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00163 failed to report scan parameters

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00164 failed to report Scan summary

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00165 failed to report conversion summary

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00166 failed to report convertible data distribution

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00167 failed to open exception report file

Cause: File open function returned error.

Action: Check if you have create/write privilege on the disk and check if the file name specified for `LOG` parameter is valid.

CSC-00168 failed to report individual exceptions

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00170 failed to retrieve used size of tablespace %s

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00171 failed to retrieve free size of tablespace %s

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00172 failed to retrieve total size of tablespace %s

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00173 failed to retrieve used size of the database

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00174 failed to retrieve free size of the database

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

CSC-00175 failed to retrieve total size of the database

Cause: Unknown.

Action: This is an internal error. Contact Oracle Customer Support.

Subsets and Supersets

Table 3–4 lists common subset/superset relationships.

Table 3–4 Subset-Superset Pairs

Subset	Superset
AR8ADOS710	AR8ADOS710T
AR8ADOS720	AR8ADOS720T
AR8ADOS720T	AR8ADOS720
AR8APTEC715	AR8APTEC715T
AR8ARABICMACT	AR8ARABICMAC
AR8ISO8859P6	AR8ASMO708PLUS
AR8ISO8859P6	AR8ASMO8X
AR8MUSSAD768	AR8MUSSAD768T
AR8MUSSAD768T	AR8MUSSAD768

Table 3–4 Subset-Superset Pairs

Subset	Superset
AR8NAFITHA711	AR8NAFITHA711T
AR8NAFITHA721	AR8NAFITHA721T
AR8SAKHR707	AR8SAKHR707T
AR8SAKHR707T	AR8SAKHR707
D7DEC	D7SIEMENS9780X
D7SIEMENS9780X	D7DEC
DK7SIEMENS9780X	N7SIEMENS9780X
I7DEC	I7SIEMENS9780X
I7SIEMENS9780X	IW8EBCDIC424
IW8EBCDIC424	IW8EBCDIC1086
KO16KSC5601	KO16MSWIN949
LV8PC8LR	LV8RST104090
N7SIEMENS9780X	DK7SIEMENS9780X
US7ASCII	See Table 3-5 for a complete list
WE16DECTST	WE16DECTST2
WE16DECTST2	WE16DECTST
WE8DEC	TR8DEC
WE8DEC	WE8NCR4970
WE8ISO8859P1	WE8MSWIN1252
WE8NCR4970	TR8DEC
WE8NCR4970	WE8DEC
WE8PC850	WE8PC858

US7ASCII is a special case because so many other character sets are supersets of it. Table 3–5 lists supersets for US7ASCII.

Table 3–5 US7ASCII Supersets

Supersets	Supersets	Supersets
AL24UTFFSS	EE8PC852	RU8BESTA
AR8ADOS710	EL8DEC	RU8PC855
AR8ADOS710T	EL8ISO8859P7	RU8PC866
AR8ADOS720	EL8MACGREEKS	SE8ISO8859P3
AR8ADOS720T	EL8MSWIN1253	TH8MACTHAIS
AR8APTEC715	EL8PC437S	TH8TISASCII
AR8APTEC715T	EL8PC851	TR8DEC
AR8ARABICMACS	EL8PC869	TR8MACTURKISHS
AR8ASMO708PLUS	ET8MSWIN923	TR8MSWIN1254
AR8ASMO8X	HU8ABMOD	TR8PC857
AR8HPARABIC8T	HU8CWI2	US8PC437
AR8ISO8859P6	IN8ISCII	UTF8
AR8MSAWIN	IS8PC861	VN8MSWIN1258
AR8MUSSAD768	IW8ISO8859P8	VN8VN3
AR8MUSSAD768T	IW8MACHEBREWS	WE8DEC
AR8NAFITHA711	IW8MSWIN1255	WE8DG
AR8NAFITHA711T	IW8PC1507	WE8ISO8859P1
AR8NAFITHA721	JA16EUC	WE8ISO8859P15
AR8NAFITHA721T	JA16SJIS	WE8ISO8859P9
AR8SAKHR706	JA16TSTSET	WE8MACROMAN8S
AR8SAKHR707	JA16TSTSET2	WE8MSWIN1252
AR8SAKHR707T	JA16VMS	WE8NCR4970
BG8MSWIN	KO16KSC5601	WE8NEXTSTEP
BG8PC437S	KO16KSCCS	WE8PC850
BLT8CP921	KO16MSWIN949	WE8PC858
BLT8MSWIN1257	KO16TSTSET	WE8PC860

Table 3–5 US7ASCII Supersets

Supersets	Supersets	Supersets
BLT8PC775	LA8ISO6937	WE8ROMAN8
BN8BSCII	LA8PASSPORT	ZHS16CGB231280
CDN8PC863	LT8MSWIN921	ZHS16GBK
CL8ISO8859P5	LT8PC772	ZHT16BIG5
CL8KOI8R	LT8PC774	ZHT16CCDC
CL8MACCYRILLICS	LV8PC1117	ZHT16DBT
CL8MSWIN1251	LV8PC8LR	ZHT16HKSCS
EE8ISO8859P2	LV8RST104090	ZHT16MSWIN950
EE8MACCES	N8PC865	ZHT32EUC
EE8MACCROATIANS	NE8ISO8859P10	ZHT32SOPS
EE8MSWIN1250	NEE8ISO8859P4	ZHT32TRIS

4

Oracle Parallel Server

This chapter describes the changes to Oracle Parallel Server for release 8.1.7. This chapter contains the following topics:

- Raw Partition Tablespace Size Requirements
- Connecting to Secondary Instances
- Recovery Manager Procedures for Oracle Parallel Server
- New Parameter OPS_INTERCONNECTS for Solaris

Raw Partition Tablespace Size Requirements

Some of the raw partition tablespace size requirements have changed for Oracle Parallel Server release 8.1.7 as shown in Table 4–1. These tablespaces require slightly greater capacities than the values that were published in the release 2 (8.1.6) documentation.

Table 4–1 Raw Partition Tablespace Size Requirements

Create a Raw Device for	With File Size
SYSTEM	275MB
TEMP	78MB for Online Transaction Processing environments and 520MB for Decision Support Systems
DRSYS	95MB

Connecting to Secondary Instances

The process of configuring Oracle Parallel Server to connect to secondary instances is simplified for release 8.1.7. Use the `INSTANCE_ROLE` parameter in the Connect Data portion of the connect descriptor to configure explicit secondary instance connections. For more detailed information and examples of `INSTANCE_ROLE` refer to Chapter 5, "Net8".

Recovery Manager Procedures for Oracle Parallel Server

The procedures for connecting Recovery Manager (RMAN) to a target database in an Oracle Parallel Server cluster have changed for release 8.1.7. For more information refer to Chapter 7 "Connecting RMAN to a Target Database in an OPS Cluster".

New Parameter OPS_INTERCONNECTS for Solaris

`OPS_INTERCONNECTS` provides information about additional cluster interconnects for use in Oracle Parallel Server environments. Oracle uses the information from this parameter to distribute traffic among the various interfaces. You would normally use `OPS_INTERCONNECTS` when a single interconnect is insufficient to meet the bandwidth requirements of large Oracle Parallel Server databases.

OPS_INTERCONNECTS is an optional parameter. If you do not set it, the current semantics that determine the appropriate interconnect for Oracle Parallel Server inter-node communication are preserved.

The syntax of the parameter is:

```
OPS_INTERCONNECTS = <if1>:<if2>:...:<ifn>
```

Where <if*n*> is an IP address in standard dotted-decimal format, for example, 144.25.16.214. Subsequent platform implementations may specify interconnects with different syntaxes.

> **Note:** When you set OPS_INTERCONNECTS in Sun Cluster 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.

5

Net8

This chapter describes new and changed features for Net8 in release 8.1.7. This chapter contains these topics:

- Configuring Directory Server Access for Oracle8i Feature Integration
- Configuring Session Data Unit for Net8 Performance
- INSTANCE_ROLE Parameter for Primary and Secondary Instance Configurations
- Oracle Names Control Utility Command Changes

Configuring Directory Server Access for Oracle8*i* Feature Integration

Directory access configuration, described on pages 6-17 through 6-18 of the *Net8 Administrator's Guide* has changed in release 8.1.7. The configuration method is described in this section.

Net8 directory naming and Oracle Advanced Security enterprise user features use a centralized directory server to store entries. If you want to use these features, you must establish a directory for them, as well as enable computers to access the directory.

You can configure directory access during or after installation. This section contains these topics:

- Configuring Directory Access During Installation
- Configuring Directory Access After Installation
- Adding Users to and Removing Users from the OracleNetAdmins Group

Configuring Directory Access During Installation

Oracle Universal Installer launches Net8 Configuration Assistant after software installation. Net8 Configuration Assistant enables you to configure access to a directory. Directory access configuration varies depending on the installation mode you selected during installation, as described in these topics:

- Directory Access Configuration During a Custom Installation on the Server
- Directory Access Configuration During a Client Installation

Directory Access Configuration During a Custom Installation on the Server

After a custom installation on the server, Net8 Configuration Assistant prompts you to configure access to a directory. Directory access configuration enables:

- Oracle Database Configuration Assistant to register a database service
- Net8 Assistant to create net service names in the directory, as well as to modify Net8 attributes of the database service entry and the net service name entries
- The server to look up database service and net service name entries in the directory

---

Note: Directory access configuration is not available during a typical or minimal installation on the server.

During directory access configuration, Net8 Configuration Assistant prompts you to configure the following directory access settings:

- Select the type of directory, that is, Oracle Internet Directory, Microsoft Active Directory, or Novell Directory Services
- Identify the location of the directory
- Select an administrative context from which this server can look up, create, and modify connect identifiers

  The administrative context is a directory entry that contains an Oracle Context (`cn=OracleContext`). An Oracle Context is the root of a directory subtree under which all Oracle software-related information is kept. A published directory entry for the administrative context must exist; otherwise, you cannot select an administrative context.

This configuration information is stored in an `ldap.ora` file that the server reads to locate the directory and to access Oracle entries.

If an Oracle Context does not exist in the directory under the selected administrative context, Net8 Configuration Assistant prompts you to create it. During Oracle Context creation, you are prompted for directory authentication credentials. If the Oracle Context is created successfully, the authenticated user is added to the following groups in the directory:

- OracleDBCreators (`cn=OracleDBCreators,cn=OracleContext`)

  Being a member of OracleDBCreators enables a user to use Oracle Database Configuration Assistant to register a database service entry.

- OracleNetAdmins (`cn=OracleNetAdmins,cn=OracleContext`)

  Being a member of OracleNetAdmins enables a user to use Net8 Assistant to create, modify, and delete net service names, and modify Net8 attributes of database services.

- OracleSecurityAdmins (`cn=OracleSecurityAdmins,cn=OracleContext`)

  Being a member of the OracleSecurityAdmins group gives the user full control over the Oracle Context. Such a user can perform the same operations as users who are members of both OracleDBCreators and OracleNetAdmins, as well as being able to perform additional enterprise security operations.

A directory administrator can add other users to these groups.

In addition, Net8 Configuration Assistant verifies that the Oracle schema was created. The Oracle schema defines the Oracle entries and their attributes. If the

schema does not exist or is an older version, you are prompted to create it. During Oracle schema creation, you are prompted for directory authentication credentials.

After Net8 Configuration Assistant completes configuration, Oracle Database Configuration Assistant creates the database. The service name for the database is automatically created under the Oracle Context.

See Also:

- "Adding Users to and Removing Users from the OracleNetAdmins Group" on page 5-8 to add users to the NetAdmins group
- *Oracle Advanced Security Administrator's Guide* for further information about adding users to the OracleDBCreators group
- Oracle installation guide

Directory Access Configuration During a Client Installation

During client installation, Net8 Configuration Assistant prompts you to configure access to a directory. Directory access configuration enables the client to look up connect identifier entries in the directory. If directory access is not configured, the client cannot use directory naming.

Net8 Configuration Assistant typically performs the necessary directory access configuration during client installation and stores the following in a read-only `ldap.ora` file.

During directory access configuration, Net8 Configuration Assistant prompts you to configure the following directory access settings:

- Specify the type of directory
- Identify the location of the directory
- Select an administrative context (that already exists in the directory) from which this client can look up connect identifiers

This configuration information is stored in a `ldap.ora` file that the client reads to locate the directory and to access Oracle entries.

In addition, Net8 Configuration Assistant verifies that the Oracle schema was installed. If an Oracle Context or the Oracle schema was not configured by the server, you cannot complete directory access configuration on the client.

See Also: "Directory Access Configuration During a Custom Installation on the Server" on page 5-2

Configuring Directory Access After Installation

Directory access can be configured with Net8 Configuration Assistant at any time.

To configure directory access:

1. Start Net8 Configuration Assistant:
   - On UNIX, run `netca` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Configuration Assistant.

   The Welcome page appears.
2. Select "Directory Service Access configuration," and then choose Next.

   The Directory Service Access page appears.

The Directory Service Access page options are as follows:

Option	Description
Configure this Oracle Home to use a directory server that is already set up for Oracle features	Choose this option to enable this computer to use a directory server that is already configured to use directory naming or enterprise user security features. This option is ideal for clients that use a directory server that has already been configured for these features. Once configuration is complete, this option enables this computer to look up entries in the directory. This option prompts you to: ■ Select the type of directory ■ Identify the location of the directory ■ Select or enter an administrative context (that already exists in the directory) from which this client can look up connect identifiers **Note:** If no Oracle Context or Oracle schema exists, you cannot configure this computer to use the directory.

Option	Description
Configure a directory server for Oracle features, and configure this Oracle Home to use the directory	Choose this option to configure a directory server for directory naming and enterprise user security features, and to enable this computer to use that directory. This option is designed for administrators who first configure these features. Once configuration is complete, this computer can then look up entries in the directory. This option prompts you to: ■ Select the type of directory ■ Identify the location of the directory ■ Select or enter an administrative context (that already exists in the directory) from which this server can access and create Oracle entries A published directory entry for the administrative context must exist; otherwise, you cannot select an administrative context. If an Oracle Context does not exist under the administrative context, you are prompted to create one. Likewise, if the Oracle schema does not exist or is an older version, you are prompted to create it. During Oracle Context or Oracle schema creation, you are prompted for directory authentication credentials. If the Oracle Context is created successfully, the authenticated user is added to the following groups in the directory: ■ OracleDBCreators (`cn=OracleDBCreators,cn=OracleContext`) ■ OracleNetAdmins (`cn=OracleNetAdmins,cn=OracleContext`) ■ OracleSecurityAdmins (`cn=OracleSecurityAdmins,cn=OracleContext`) **See Also:** ■ Directory specific documentation for directory entry configuration instructions ■ "Adding Users to and Removing Users from the OracleNetAdmins Group" on page 5-8 to add users to the OracleNetAdmins group ■ *Oracle Advanced Security Administrator's Guide* for further information about adding users to the OracleDBCreators group

Option	Description
Create a new Oracle Context	Choose this option to create an additional Oracle Context in the directory. To create an Oracle Context, the following must exist in the directory: ■ A directory entry under which you want to create the Oracle Context ■ An Oracle schema During Oracle Context creation, you are prompted for directory authentication credentials. If the Oracle Context is created successfully, the authenticated user is added to the following groups in the directory: ■ OracleDBCreators (`cn=OracleDBCreators,cn=OracleContext`) ■ OracleNetAdmins (`cn=OracleNetAdmins,cn=OracleContext`) ■ OracleSecurityAdmins (`cn=OracleSecurityAdmins,cn=OracleContext`)
Create or update Oracle schema objects	Choose this option to create the Oracle schema in the directory, or update the Oracle schema to the current release. During Oracle schema creation, you are prompted for directory authentication credentials.

3. Select the appropriate option, and then follow the prompts in the wizard and online help to complete directory access configuration.

Adding Users to and Removing Users from the OracleNetAdmins Group

The user that creates the Oracle Context is a member of the OracleNetAdmins (`cn=OracleNetAdmins,cn=OracleContext`) group. Using directory tools, such as `ldapmodify`, a directory administrator or the directory user who created the Oracle Context can add users to this group.

To add a user to the OracleNetAdmins group with `ldapmodify`:

1. Create an LDAP Data Interchange Format (LDIF) file that specifies that you want to add a user to the OracleNetAdmins group. You can use the following sample LDIF file with your own settings for the Distinguished Name (DN) for `cn=OracleNetAdmins` and the user that you want to add.

   ```
   dn: cn=OracleNetAdmins,cn=OracleContext,...
   changetype: modify
   add: uniquemember
   uniquemember: <DN of user being added to group>
   ```

2. Use the following `ldapmodify` syntax to add the user:

   ```
   ldapmodify -h host -p port -D binddn
   -w password -f ldif_file
   ```

Argument	Description
`-h` *`host`*	Specify the directory server host.
`-p` *`port`*	Specify the listening TCP/IP port for the directory server. If you do not specify this option, the default port (389) is used.
`-D` *`binddn`*	Specify the directory administrator or user DN.
`-w` *`password`*	Specify the password for the directory administrator or directory user.
`-f` *`ldif_file`*	Specify the input file name.

Configuring Session Data Unit for Net8 Performance

Before sending data across the network, Net8 buffers and encapsulates data into the session data unit (SDU). Net8 sends the data stored in this buffer when the buffer is full, flushed, or when RDBMS tries to read data. When large amounts of data are being transmitted or when the message size is consistent, adjusting the size of the SDU buffers can improve performance, network utilization, or memory consumption.

The SDU size can range from 512 bytes to 32 KB. The default SDU for the client and the database is 2 KB.

Optimal SDU size depends on the maximum segment size (MSS) and message fragmentation. For TTC connections, configuring an SDU size larger than the 2 KB default requires configuring the SDU on both the client and server computers. When the configured values do not match, the lower of the two values will be used.

To minimize packet header overhead and message fragmentation, set the SDU size as a multiple of the MSS. When Oracle Advanced Security encryption is not used, increase the SDU size by one (1). For example, the TCP/IP version 4 MSS on Ethernet is 1460 bytes. Use a multiple of 1460 for the SDU size if encryption is used. If encryption is not used, increase the SDU size to 1461.

The packet header overhead and message fragmentation can be measured using a network sniffer or by analyzing Net8 trace files.

SDU Configuration on Clients

To configure the client, set the SDU size with the `SDU` parameter in a connect descriptor as follows:

```
net_service_name=
 (DESCRIPTION=
   (SDU=2920)
   (ADDRESS=...)
   (ADDRESS=...)
   (CONNECT_DATA=
     (SERVER_NAME=sales.us.acme.com)))
```

SDU Configuration on the Server

Database server configuration depends upon whether or not the database is configured to use MTS or dedicated servers.

MTS Configuration with SDU

If using MTS, set the SDU size in the `MTS_DISPATCHERS` parameter as follows:

```
MTS_DISPATCHERS="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp))(SDU=2920))"
```

Ensure the SDU size matches the value configured for the client.

Dedicated Server Configuration with SDU

If using dedicated servers for a database that is dynamically registered with the listener through service registration, then the SDU size cannot be set. Instead, the 2 KB default is used.

If using dedicated servers for a database that is registered with the listener through static configuration in the `listener.ora` file, then set the SDU size in the `SID_DESC` section of the `listener.ora` file as follows:

```
SID_LIST_listener_name=
  (SID_LIST=
    (SID_DESC=
     (SDU=2920)
     (SID_NAME=sales)))
```

Ensure the SDU size matches the value configured for the client.

INSTANCE_ROLE Parameter for Primary and Secondary Instance Configurations

The INSTANCE_ROLE parameter is an optional parameter for the CONNECT_DATA section of a connect descriptor. It enables you to specify a connection to the primary or secondary instance in Oracle Parallel Server and Oracle Parallel Fail Safe primary/secondary configurations.

This parameter is useful when:

- You want to explicitly connect to a primary or secondary instance. The default is the primary instance.
- You want to use Transparent Application Failover (TAF) to preconnect to a secondary instance.

INSTANCE_ROLE supports the following values:

primary — Specifies a connection to the primary instance

secondary — Specifies a connection to the secondary instance

any — Specifies a connection to whichever instance has the lowest load, regardless of primary or secondary instance role

Example: Connection to Instance Role Type

In the following example, net service name sales_primary enables connections to the primary instance, and net service name sales_secondary enables connections to the secondary instance.

```
sales_primary=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales1-server)
       (PORT=1521))
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales2-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.acme.com)
     (INSTANCE_ROLE=primary)))
```

```
sales_secondary=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales1-server)
       (PORT=1521))
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales2-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.acme.com)
     (INSTANCE_ROLE=secondary)))
```

Example: Connection To a Specific Instance

There are times when Oracle Enterprise Manager and other system management products need to connect to a specific instance regardless of its role to perform administrative tasks. For these types of connections, configure (`INSTANCE_NAME=`*`instance_name`*`)` and (`INSTANCE_ROLE=any`) to connect to the instance regardless of its role.

In the following example, net service name `sales1` enables connections to the instance on `sales1-server` and `sales2` enables connections to the instance on `sales2-server`. (`SERVER=dedicated`) is specified to force a dedicated server connection.

```
sales1=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales1-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.acme.com)
     (INSTANCE_ROLE=any)
     (INSTANCE_NAME=sales2)
     (SERVER=dedicated)))
sales2=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales2-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.acme.com)
     (INSTANCE_ROLE=any)
     (INSTANCE_NAME=sales2)
     (SERVER=dedicated)))
```

Example: TAF Pre-Establishing a Connection

If Transparent Application Failover (TAF) is configured, a backup connection can be pre-established to the secondary instance. The initial and backup connections must be explicitly specified. In the following example, Net8 connects to the listener on `sales1-server` and preconnects to `sales2-server`, the secondary instance. If `sales1-server` fails after the connection, the TAF application fails over to `sales2-server`, the secondary instance, preserving any `SELECT` statements in progress.

```
sales1.acme.com=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales1-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.acme.com)
     (INSTANCE_ROLE=primary)
     (FAILOVER_MODE=
       (BACKUP=sales2.acme.com)
       (TYPE=select)
       (METHOD=preconnect))))
sales2.acme.com=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales2-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.acme.com)
     (INSTANCE_ROLE=secondary)))
```

Oracle Names Control Utility Command Changes

In this release of 8.1.7, the following changes affect the Oracle Names Control utility:

- DOMAIN_HINT Command Not Supported
- REGISTER_NS and UNREGISTER_NS Commands for Oracle Names Server Registration

DOMAIN_HINT Command Not Supported

The Oracle Names Control utility command `DOMAIN_HINT` to create a domain hint is no longer supported in this release. Instead, configure the `NAMES.DOMAIN_HINTS` parameter in the `names.ora` file.

A domain hint contains the name of the domain and at least one address of an Oracle Names server in that domain. This enables an Oracle Names server to forward client requests to a specific address, reducing network traffic.

It is necessary for all Oracle Names servers in delegated administrative regions to be configured with a domain hint to an Oracle Names server in the root administration region. This enables Oracle Names server to forward requests anywhere outside their own subtrees, because all name lookups can be found by forwarding through the root. If a domain hint to the root administrative region is not specified, then the local Oracle Names server may be unable to forward requests to other regions.

If you want to forward requests directly to other administrative regions, configure additional domain hints. This can ensure that clients do not have to wait for a request to be forwarded from the root administrative region. If domain hints to other administrative regions are not specified, the local Oracle Names server may go through unnecessary routing.

The local Oracle Names server will forward a client request on to whatever remote Oracle Names servers it knows, who then forward the request to the root Oracle Names server in its region. The root Oracle Names server will forward the request to the Oracle Names servers that have information on the domain that the request refers to.

To configure a domain hint, manually configure the `names.ora` file with the `NAMES.DOMAIN_HINTS` parameter. The syntax for this parameter follows:

```
NAMES.DOMAIN_HINTS=
 (HINT_DESC=
   (HINT_LIST=
    (HINT=(NAME=onames_server)(ADDRESS=...)))
   (DOMAIN_LIST=
    (DOMAIN=domain)))
```

Note: Specify the root domain with a dot (.) or a null value.

The `HINT_LIST` specifies the list of Oracle Names servers to forward the initial set of queries for the domains listed in the `DOMAIN_LIST`. A hint contains the name and address of an Oracle Names server in the remote administrative region. The Oracle Names server caches the results of those queries in its memory. If the queries fail, the Oracle Names servers listed in the `HINT_LIST` will not be cached and the local Oracle Names server continues to run without information about the root administrative region. The `HINT_LIST` should include enough Oracle Names servers to guarantee that the local Oracle Names server can resolve the query for the root.

Example: Domain Hint for the Root Administrative Region

In the following example, `NAMES.DOMAINS_HINTS` contains a domain hint for Oracle Names server `rootsvr.com` that is located in the root domain of the remote administrative region. The `DOMAIN` parameter is left null, meaning that the hint is for the root domain.

```
NAMES.DOMAIN_HINTS=
 (HINT_DESC=
   (HINT_LIST=
    (HINT=(NAME=rootsvr.com)(ADDRESS=(PROTOCOL=tcp)(HOST=rootsvr)(PORT=1575))))
   (DOMAIN_LIST=
    (DOMAIN=)))
```

When the local Oracle Names server is started:

1. It reads the NAMES.DOMAIN_HINTS parameter. For each domain listed in the DOMAIN_LIST, it calls the Oracle Name server listed in the HINT_LIST and queries for all Oracle Names servers in the domain.
2. Based on the time-to-live (TTL) of the answer, the local Oracle Names server sets up queries that are automatically issued before the remote Oracle Names server data expires.

Example: Domain Hint for Multiple Oracle Names Servers

The following example shows a hint to query two domains, the root domain and the acme.com domain, for Oracle Names servers rootsvr1.com and rootsvr2.com.

```
NAMES.DOMAIN_HINTS=
 (HINT_DESC=
   (HINT_LIST=
    (HINT=(NAME=rootsvr1.com)(ADDRESS=(PROTOCOL=tcp)(HOST=sales-pc)(PORT=1575)))
    (HINT=(NAME=rootsvr2.com)(ADDRESS=(PROTOCOL=tcp)(HOST=hr-pc)(PORT=1575))))
   (DOMAIN_LIST=
    (DOMAIN=)
    (DOMAIN=acme.com)))
```

In this query, the local Oracle Names server:

1. Sends a query for the root administrative region, using the addresses given in the HINT
2. Sends a query for the acme.com domain, using the same addresses from the HINT descriptor

REGISTER_NS and UNREGISTER_NS Commands for Oracle Names Server Registration

Two new Oracle Names Control utility commands, REGISTER_NS and UNREGISTER _NS, enable you to register and unregister an Oracle Names server as an authoritative server for a given domain. These commands replace Oracle Names server registration capabilities that were previously available with the REGISTER and UNREGISTER commands.

REGISTER_NS

Purpose

Use the REGISTER_NS command to define an Oracle Names server and its authoritative domain.

Prerequisites

None

Password required if one has been set

No

Syntax

From the operating system:

```
NAMESCTL REGISTER_NS {onames_server}{(ADDRESS=...)}{domain}
```

From Oracle Names Control utility:

```
NAMESCTL> REGISTER_NS {onames_server}{(ADDRESS=...)}{domain}
```

Arguments

{*onames_server*}—Specify the Oracle Names server name.

{(ADDRESS=...)}—Specify the Oracle Names server protocol address.

{*domain*}—Specify the domain name.

Usage Notes

This command provides a mechanism for registering an Oracle Names server as an authoritative server for a given domain. The command adds a network session record type, NS.SMD, for the Oracle Names server to the domain, and provides the Oracle Names server with an address record, A.SMD.

This command will fail if either the domain exists and has non-NS records or the server exists and has a type of service record that is other than 'ORACLE_NAMESERVER'.

Ordinarily, the Oracle Names servers will maintain their own data by registering themselves when they start. This command is provided as a manual way to manage domain and Oracle Names server data if for some reason the Oracle Names server cannot. This may occur if the region database tables are set up as read-only for security reasons.

If the Oracle Names servers are not registering themselves, this command should be used to define the region topology data. Each Oracle Names server in the region should be defined using this command for each top-level domain in the region. Usually, the top level consists of a single parent domain, for example, acme.com. However, a region may also have multiple sibling parent domains, for example, a region covering North America would have US, CA, and MX as its top-level parent domains.

Note the regions which were defined using the Oracle Network Manager in SQL*Net version 2 have NS.SMD records defined for every domain in the administrative region, but in Net8 only the top-level parent domains need to have NS.SMD records defined for each server in the region.

Use the Oracle Names Control utility DELEGATE DOMAIN command to define Oracle Names servers which are delegation points for subregions.

Use the NAMES.DOMAIN_HINTS parameter in the names.ora file to provide data about any other Oracle Names servers in foreign regions.

Example

```
NAMESCTL> REGISTER_NS  namesrv1
(ADDRESS=(PROTOCOL=tcp)(HOST=namesvr)(PORT=1575))
Total response time:   7 minutes 59.14 seconds
Response status:       normal, successful completion
```

UNREGISTER_NS

Purpose

Use the `UNREGISTER_NS` command to undefine an Oracle Names server and its authoritative domain.

Prerequisites

None

Password required if one has been set

No

Syntax

From the operating system:

`NAMESCTL UNREGISTER_NS {`*`onames_server`*`}{(ADDRESS=...)}{`*`domain`*`}`

From Oracle Names Control utility:

`NAMESCTL> UNREGISTER_NS {`*`onames_server`*`}{(ADDRESS=...)}{`*`domain`*`}`

Arguments

`{`*`onames_server`*`}`—Specify the Oracle Names server name.

`{(ADDRESS=...)}`—Specify the Oracle Names server protocol address.

`{`*`domain`*`}`—Specify the domain name.

Usage Notes

This command provides a mechanism for unregistering an Oracle Names server as an authoritative server for a given domain. This command removes the `NS.SMD` record for the Oracle Names from the domain, and deletes the Oracle Names server and its `A.SMD` address record.

This command will fail if either the domain exists and has non-NS records or the server exists and has a type of service record that is other than '`ORACLE_NAMESERVER`'.

Ordinarily, the Oracle Names servers will maintain their own data by registering themselves when they start. This command is provided as a manual way to manage domain and Oracle Names server data if for some reason the Oracle Names server cannot. This can occur if the region database tables are set up as read-only for security reasons.

If the Oracle Names servers are not registering themselves, this command should be used to define the region topology data. Each Oracle Names server in the region should be defined using this command for each top-level domain in the region. Usually, the top level consists of a single parent domain, for example, `acme.com`. However, a region may also have multiple sibling parent domains, for example, a region covering North America would have US, CA and MX as its top-level parent domains.

Note the regions which were defined using the Oracle Network Manager in SQL*Net version 2 have `NS.SMD` records defined for every domain in the administrative region, but in Net8 only the top-level parent domains need to have `NS.SMD` records defined for each server in the region.

Example

```
NAMESCTL> UNREGISTER_NS  namesrv1
(ADDRESS=(PROTOCOL=tcp)(HOST=namesvr)(PORT=1575))
Total response time:   7 minutes 59.14 seconds
Response status:       normal, successful completion
```

6

PL/SQL Supplied Packages

This chapter contains a description of the new DBMS_LDAP package and the DBMS_OBFUSCATION_TOOLKIT which had significant changes for the 8.1.7 release.

This addendum contains these topics:

- Package Overview
- DBMS_LDAP
- DBMS_OBFUSCATION_TOOLKIT

Note: Oracle provides packages with various products, such as Oracle Developer and the Oracle Application Server. This manual, however, only covers the packages that Oracle provides with the database server.

See Also: For detailed information on how to create your own packages, see *Oracle8i Application Developer's Guide - Fundamentals*.

Package Overview

A *package* is an encapsulated collection of related program objects stored together in the database. Program objects are procedures, functions, variables, constants, cursors, and exceptions.

Packages have many advantages over stand-alone procedures and functions. For example, they:

- Let you organize your application development more efficiently.
- Let you grant privileges more efficiently.
- Let you modify package objects without recompiling dependent schema objects.
- Enable Oracle to read multiple package objects into memory at once.
- Let you *overload* procedures or functions. Overloading means creating multiple procedures with the same name in the same package, each taking arguments of different number or datatype.
- Can contain global variables and cursors that are available to all procedures and functions in the package.

Using Oracle Supplied Packages

Most Oracle supplied packages are automatically installed when the database is created and the `CATPROC.SQL` script is run. For example, to create the `DBMS_ALERT` package, the `DBMSALRT.SQL` and `PRVTALRT.PLB` scripts must be run when connected as the user `SYS`. These scripts, however, are run automatically by the `CATPROC.SQL` script.

Certain packages are not installed automatically. Special installation instructions for these packages are documented in the individual chapters.

To call a PL/SQL function from SQL, you must either own the function or have `EXECUTE` privileges on the function. To select from a view defined with a PL/SQL function, you must have `SELECT` privileges on the view. No separate `EXECUTE` privileges are needed to select from the view. Instructions on special requirements for packages are documented in the individual chapters.

Creating New Packages

There are two distinct steps to creating a new package:

1. Create the package *specification* with the CREATE PACKAGE statement.

   You can declare program objects in the package specification. Such objects are called *public* objects. Public objects can be referenced outside the package, as well as by other objects in the package.

   Note: It is often more convenient to add the OR REPLACE clause in the CREATE PACKAGE statement.

2. Create the package *body* with the CREATE PACKAGE BODY statement.

   You can declare and define program objects in the package body:

   - You must define public objects declared in the package specification
   - You can declare and define additional package objects. Such objects are called *private* objects. Private objects are declared in the package body rather than in the package specification, so they can be referenced only by other objects in the package: They cannot be referenced outside the package

   See Also: For more information on creating new packages, see *PL/SQL User's Guide and Reference* and *Oracle8i Application Developer's Guide - Fundamentals.* For more information on storing and executing packages, see *Oracle8i Concepts.*

Separation of Specification and Body

The specification of a package *declares* the public types, variables, constants, and subprograms that are visible outside the immediate scope of the package. The body of a package *defines* the objects declared in the specification, as well as private objects that are not visible to applications outside the package.

Oracle stores the specification and body of a package separately in the database. Other schema objects that call or reference public program objects depend only on the package specification, not on the package body. This distinction allows you to change the definition of a program object in the package body without causing Oracle to invalidate other schema objects that call or reference the program object. Oracle invalidates dependent schema objects only if you change the declaration of the program object in the package specification.

Example The following example shows a package specification for a package named EMPLOYEE_MANAGEMENT. The package contains one stored function and two stored procedures.

```
CREATE PACKAGE employee_management AS
   FUNCTION hire_emp (name VARCHAR2, job VARCHAR2,
      mgr NUMBER, hiredate DATE, sal NUMBER, comm NUMBER,
      deptno NUMBER) RETURN NUMBER;
   PROCEDURE fire_emp (emp_id NUMBER);
   PROCEDURE sal_raise (emp_id NUMBER, sal_incr NUMBER);
END employee_management;
```

The body for this package defines the function and the procedures:

```
CREATE PACKAGE BODY employee_management AS
   FUNCTION hire_emp (name VARCHAR2, job VARCHAR2,
      mgr NUMBER, hiredate DATE, sal NUMBER, comm NUMBER,
      deptno NUMBER) RETURN NUMBER IS
```

The function accepts all arguments for the fields in the employee table except for the employee number. A value for this field is supplied by a sequence. The function returns the sequence number generated by the call to this function.

```
       new_empno    NUMBER(10);

   BEGIN
      SELECT emp_sequence.NEXTVAL INTO new_empno FROM dual;
      INSERT INTO emp VALUES (new_empno, name, job, mgr,
         hiredate, sal, comm, deptno);
      RETURN (new_empno);
   END hire_emp;

   PROCEDURE fire_emp(emp_id IN NUMBER) AS
```

The procedure deletes the employee with an employee number that corresponds to the argument emp_id. If no employee is found, then an exception is raised.

```
   BEGIN
      DELETE FROM emp WHERE empno = emp_id;
      IF SQL%NOTFOUND THEN
      raise_application_error(-20011, 'Invalid Employee
         Number: ' || TO_CHAR(emp_id));
   END IF;
END fire_emp;

PROCEDURE sal_raise (emp_id IN NUMBER, sal_incr IN NUMBER) AS
```

The procedure accepts two arguments. `Emp_id` is a number that corresponds to an employee number. `Sal_incr` is the amount by which to increase the employee's salary.

```
   BEGIN

   -- If employee exists, then update salary with increase.

      UPDATE emp
         SET sal = sal + sal_incr
         WHERE empno = emp_id;
      IF SQL%NOTFOUND THEN
         raise_application_error(-20011, 'Invalid Employee
            Number: ' || TO_CHAR(emp_id));
      END IF;
   END sal_raise;
END employee_management;
```

Note: If you want to try this example, then first create the sequence number `emp_sequence`. You can do this using the following SQL*Plus statement:

```
SQL> EXECUTE CREATE SEQUENCE emp_sequence
   > START WITH 8000 INCREMENT BY 10;
```

Referencing Package Contents

To reference the types, items, and subprograms declared in a package specification, use the dot notation (you might need to specify the schema also). For example:

```
package_name.type_name
package_name.item_name
package_name.subprogram_name
```

DBMS_LDAP

The PL/SQL package DBMS_LDAP contains the functions and procedures which can be used by PL/SQL programmers to access data from LDAP servers. This section explains all of the API functions in detail. End users (client programmers) are expected to have read through the users guide and the examples (use cases). The information presented in this section should be used as a programming reference.

Summary of Subprograms

Table 6–1 DBMS_LDAP API Subprograms

Function or Procedure	Description
FUNCTION init	init() initializes a session with an LDAP server. This actually establishes a connection with the LDAP server.
FUNCTION simple_bind_s	The function simple_bind_s can be used to perform simple username/password based authentication to the directory server.
FUNCTION bind_s	The function bind_s can be used to perform complex authentication to the directory server.
FUNCTION unbind_s	The function unbind_s is used for closing an active LDAP session.
FUNCTION compare_s	The function compare_s can be used to test if a particular attribute in a particular entry has a particular value.
FUNCTION search_s	The function search_s performs a synchronous search in the LDAP server. It returns control to the PL/SQL environment only after all of the search results have been sent by the server or if the search request is 'timed-out' by the server.
FUNCTION search_st	The function search_st performs a synchonous search in the LDAP server with a client side timeout. It returns control to the PL/SQL environment only after all of the search results have been sent by the server or if the search request is 'timed-out' by the client or the server.
FUNCTION first_entry	The function first_entry is used to retrieve the first entry in the result set returned by either search_s or search_st.
FUNCTION next_entry	The function next_entry() is used to iterate to the next entry in the result set of a search operation.
FUNCTION count_entries	This function is used to count the number of entries in the result set. It can also be used to count the number of entries remaining during a traversal of the result set using a combination of the functions first_entry() and next_entry().
FUNCTION first_attribute	The function first_attribute() fetches the first attribute of a given entry in the result set.

Table 6–1 DBMS_LDAP API Subprograms

Function or Procedure	Description
FUNCTION next_attribute	The function next_attribute() fetches the next attribute of a given entry in the result set.
FUNCTION get_dn	The function get_dn() retrieves the X.500 distinguished name of given entry in the result set.
FUNCTION get_values	The function get_values() can be used to retrieve all of the values associated for a given attribute in a given entry.
FUNCTION get_values_ len	The function get_values_len() can be used to retrieve values of attributes that have a binary' syntax.
FUNCTION delete_s	This function can be used to remove a leaf entry in the LDAP Directory Information Tree.
FUNCTION modrdn2_s	The function modrdn2_s() can be used to rename the relative distinguished name of an entry.
FUNCTION err2string	The function err2string() can be used to convert an LDAP error code to string in the local language in which the API is operating.
FUNCTION create_mod_ array	The function create_mod_array() allocates memory for array modification entries that will be applied to an entry using the modify_s() functions.
PROCEDURE populate_ mod_array (String Version)	Populates one set of attribute information for add or modify operations.
PROCEDURE populate_ mod_array (Binary Version)	Populates one set of attribute information for add or modify operations. This procedure call has to happen after DBMS_ LDAP.create_mod_array() is called.
FUNCTION modify_s	Performs a sychronous modification of an existing LDAP directory entry.
FUNCTION add_s	Adds a new entry to the LDAP directory synchronously. Before calling `add_s`, you have to call `DBMS_LDAP.creat_mod_ array` () and `DBMS_LDAP.populate_mod_array`() first.
PROCEDURE free_mod_ array	Frees the memory allocated by `DBMS_LDAP.create_mod_ array`().
FUNCTION count_values	Counts the number of values returned by `DBMS_LDAP.get_ values` ().
FUNCTION count_ values_len	Counts the number of values returned by DBMS_LDAP.get_ values_len ().

Table 6–1 DBMS_LDAP API Subprograms

Function or Procedure	Description
FUNCTION rename_s	Renames an LDAP entry synchronously.
FUNCTION explode_dn	Breaks a dn up into its components.
FUNCTION open_ssl	Establishes an SSL (Secure Sockets Layer) connection over an existing LDAP connection.

Exception Summary

The DBMS_LDAP package shipped with RDBMS 8.1.7 can generate the following exceptions

Table 6–2 DBMS_LDAP Exception Summary

Exception Name	Oracle Error Number	Cause of Exception
general_error	31202	Raised anytime an error is encountered that does not have a specific PL/SQL exception associated with it. The error string contains the description of the problem in the local language of the user.
init_failed	31203	Raised by DBMS_LDAP.init() if there are some problems.
invalid_session	31204	Raised by all functions and procedures in the DBMS_LDAP package if they are passed an invalid session handle.
invalid_auth_method	31205	Raised by DBMS_LDAP.bind_s() if the authentication method requested is not supported.
invalid_search_scope	31206	Raised by all of the search functions if the scope of the search is invalid.
invalid_search_time_val	31207	Raised by time based search function: DBMS_LDAP.search_st() if it is given an invalid value for the time limit.
invalid_message	31208	Raised by all functions that iterate through a result-set for getting entries from a search operation if the message handle given to them is invalid.
count_entry_error	31209	Raised by DBMS_LDAP.count_entries if it cannot count the entries in a given result set.
get_dn_error	31210	Raised by DBMS_LDAP.get_dn if the dn of the entry it is retrieving is NULL.
invalid_entry_dn	31211	Raised by all the functions that modify add or rename an entry if they are presented with an invalid entry dn.

Table 6–2 DBMS_LDAP Exception Summary

Exception Name	Oracle Error Number	Cause of Exception
invalid_mod_array	31212	Raised by all functions that take a modification array as an argument if they are given an invalid modification array.
invalid_mod_option	31213	Raised by DBMS_LDAP.populate_mod_array if the modification option given is anything other than MOD_ADD, MOD_DELETE or MOD_REPLACE.
invalid_mod_type	31214	Raised by DBMS_LDAP.populate_mod_array if the attribute type that is being modified is NULL.
invalid_mod_value	31215	Raised by DBMS_LDAP.populate_mod_array if the modification value parameter for a given attribute is NULL.
invalid_rdn	31216	Raised by all functions and procedures that expect a valid RDN if the value of the RDN is NULL.
invalid_newparent	31217	Raised by DBMS_LDAP.rename_s if the new parent of an entry being renamed is NULL.
invalid_deleteoldrdn	31218	Raised by DBMS_LDAP.rename_s if the deleteoldrdn parameter is invalid.
invalid_notypes	31219	Raised by DBMS_LDAP.explode_dn if the notypes parameter is invalid.
invalid_ssl_wallet_loc	31220	Raised by DBMS_LDAP.open_ssl if the wallet location is NULL but the SSL authentication mode requires a valid wallet.
invalid_ssl_wallet_password	31221	Raised by DBMS_LDAP.open_ssl if the wallet password given is NULL.
invalid_ssl_auth_mode	31222	Raised by DBMS_LDAP.open_ssl if the SSL authentication mode is not one of 1, 2 or 3.
mts_mode_not_supported	31398	Raised by the functions init(), bind_s() or simple_bind_s() if they are ever invoked in MTS mode.

Data-Type Summary

The DBMS_LDAP package uses the following data-types.

Table 6–3 DBMS_LDAP Data-Type Summary

Data-Type	Purpose
SESSION	Used to hold the handle of the LDAP session. Nearly all of the functions in the API require a valid LDAP session to work.
MESSAGE	Used to hold a handle to the message retrieved from the result set. This is used by all functions that work with entries attributes and values.
MOD_ARRAY	Used to hold a handle into the array of modifications being passed into either modify_s() or add_s().
TIMEVAL	Used to pass time limit information to the LDAP API functions that require a time limit.
BER_ELEMENT	Used to hold a handle to a BER structure used for decoding incoming messages.
STRING_COLLECTION	Used to hold a list of VARCHAR2 strings which can be passed on to the LDAP server.
BINVAL_COLLECTION	Used to hold a list of RAW data which represent binary data.
BERVAL_COLLECTION	Used to hold a list of BERVAL values that are used for populating a modification array.

FUNCTION init

init() initializes a session with an LDAP server. This actually establishes a connection with the LDAP server.

Syntax

```
FUNCTION init      (hostname IN VARCHAR2,
                    portnum  IN PLS_INTEGER )
      RETURN SESSION;
```

Parameters

Table 6–4 INIT Function Parameters

Parameter	Description
hostname	Contains a space-separated list of hostnames or dotted strings representing the IP address of hosts running an LDAP server to connect to. Each hostname in the list MAY include a port number which is separated from the host itself with a colon (:) character. The hosts will be tried in the order listed, stopping with the first one to which a successful connection is made.
portnum	Contains the TCP port number to connect to. If a host includes a port number then this parameter is ignored. If this parameter is not specified and the hostname also does not contain the port number, a default port number of 389 is assumed.

Return Values

Table 6–5 INIT Function Return Values

Value	Description
SESSION (function return)	A handle to an LDAP session which can be used for further calls into the API.

Exceptions

Table 6–6 INIT Function Exceptions

Exception	Description
init_failed	Raised when there is a problem contacting the LDAP server.
ts_mode_not_supported	Raised if DBMS_LDAP.init() is invoked from a user session that is logged onto the database using an MTS service.
general_error	For all other errors. The error string associated with the exception describes the error in detail.

Usage Notes

DBMS_LDAP.init() is the first function that should be called in order to establish a session to the LDAP server. Function DBMS_LDAP.`init()` returns a "session handle," a pointer to an opaque structure that MUST be passed to subsequent calls pertaining to the session. This routine will return NULL and raise the “INIT_FAILED” exception if the session cannot be initialized.Subsequent to the call to init(), the connection has to be authenticated using DBMS_LDAP.bind_s or DBMS_LDAP.simple_bind_s().

See Also

DBMS_LDAP.simple_bind_s(), DBMS_LDAP.bind_s().

FUNCTION simple_bind_s

The function simple_bind_s can be used to perform simple username/password based authentication to the directory server.

Syntax

```
FUNCTION simple_bind_s (    ld     IN SESSION,
                            dn     IN VARCHAR2,
                            passwd IN VARCHAR2)
      RETURN PLS_INTEGER;
```

Parameters

Table 6–7 SIMPLE_BIND_S Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
dn	The distinguished name of the user who is attempting to login
passwd	A text string containing the password.

Return Values

Table 6–8 SIMPLE_BIND_S Function Return Values

Value	Description
PLS_INTEGER (function return)	DBMS_LDAP SUCCESS on a successful completion. If there was a problem, one of the following exceptions will be raised.

Exceptions

Table 6–9 SIMPLE_BIND_S Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
mts_mode_not_supported	Raised if DBMS_LDAP.init() is invoked from a user session that is logged onto as an MTS service.
general_error	For all other errors. The error string associated with this exception will explain the error in detail.

Usage Notes

DBMS_LDAP.simple_bind_s() can be used to authenticate a user whose directory distinguished name and directory password are known. It can be called only after a valid LDAP session handle is obtained from a call to DBMS_LDAP.init().

FUNCTION bind_s

The function bind_s can be used to perform complex authentication to the directory server.

Syntax

```
FUNCTION  bind_s       (    ld     IN SESSION,
                            dn     IN VARCHAR2,
                            passwd IN VARCHAR2,
                            meth IN PLS_INTEGER )
      RETURN PLS_INTEGER;
```

Parameters

Table 6–10 BIND_S Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
dn	The distinguished Name of the User that you are trying to login as.
cred	A text string containing the credentials used for authentication.
meth	The authentication method.

Return Values

Table 6–11 BIND_S Function Return Values

Value	Description
PLS_INTEGER (function return)	DBMS_LDAP.SUCCESS on a successful completion. One of the following exceptions is raised if there was a problem.

Exceptions

Table 6–12 BIND_S Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_auth_method	Raised if the authentication method requested is not supported.
mts_mode_not_supported	Raised if invoked from a user session that is logged onto an MTS service.
general_error	For all other errors. The error string associated with this exception will explain the error in detail.

Usage Notes

DBMS_LDAP.bind_s() can be used to authenticate a user. It can be called only after a valid LDAP session handle is obtained from a call to DBMS_LDAP.init().

See Also

DBMS_LDAP.init(), DBMS_LDAP.simple_bind_s().

FUNCTION unbind_s

The function unbind_s is used for closing an active LDAP session.

Syntax

```
FUNCTION unbind_s (ld IN SESSION )  RETURN PLS_INTEGER;
```

Parameters

Table 6–13 UNBIND_S Function Parameters

Parameter	Description
ld	A valid LDAP session handle

Return Values

Table 6–14 UNBIND_S Function Return Values

Value	Description
PLS_INTEGER (function return)	SUCCESS on proper completion. One of the following exceptions is raised otherwise.

Exceptions

Table 6–15 UNBIND_S Function Exceptions

Exception	Description
invalid_session	Raised if the sessions handle ld is invalid.
general error	For all other errors. The error string associated with this exception will explain the error in detail.

Usage Notes

The `unbind_s()` function, will send an unbind request to the server, close all open connections associated with the LDAP session and dispose of all resources associated with the session handle before returning. After a call to this function, the session handle ld is invalid and it is illegal to make any further LDAP API calls using ld.

See Also

DBMS_LDAP.bind_s(), DBMS_LDAP.simple_bind_s().

FUNCTION compare_s

The function compare_s can be used to test if a particular attribute in a particular entry has a particular value.

Syntax

```
FUNCTION compare_s (    ld    IN SESSION,
                        dn    IN VARCHAR2,
                        attr  IN VARCHAR2,
                        value IN VARCHAR2)
      RETURN PLS_INTEGER;
```

Parameters

Table 6–16 COMPARE_S Function Parameters

Parameter	Description
ld	A valid LDAP session handle
dn	The name of the entry to compare against
attr	The attribute to compare against.
value	A string attribute value to compare against

Return Values

Table 6–17 COMPARE_S Function Return Values

Value	Description
PLS_INTEGER (function return)	COMPARE_TRUE is the given attribute has a matching value. COMPARE_FALSE if the value of the attribute does not match the value given.

Exceptions

Table 6–18 COMPARE_S Function Exceptions

Exception	Description
invalid_session	Raised if hte session handle ld is invalid.
general_error	For all other errors. The error string associated with this exception will explain the error in detail.

Usage Notes

The function compare_s can be used to determine if the value of a given attribute stored in the directory server matches a certain value.This operation can only be performed on attributes whose syntax definition allows them to be compared. The compare_s function can only be called after a valid LDAP session handle has been obtained from the init() function and authenticated using the bind_s() or simple_bind_s() functions.

See Also

DBMS_LDAP.bind_s()

FUNCTION search_s

The function search_s performs a synchronous search in the LDAP server. It returns control to the PL/SQL environment only after all of the search results have been sent by the server or if the search request is timed-out by the server.

Syntax

```
FUNCTION search_s  (    ld       IN  SESSION,
                        base     IN  VARCHAR2,
                        scope    IN  PLS_INTEGER,
                        filter   IN  VARCHAR2,
                        attrs    IN  STRING_COLLECTION,
                        attronly IN  PLS_INTEGER,
                        res      OUT MESSAGE)
      RETURN PLS_INTEGER;
```

Parameters

Table 6–19 SEARCH_S Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
base	The dn of the entry at which to start the search.
scope	One of SCOPE_BASE (0x00), SCOPE_ONELEVEL (0x01), or SCOPE_SUBTREE (0x02), indicating the scope of the search.
filter	A character string representing the search filter. The value NULL can be passed to indicate that the filter "(objectclass=*)" which matches all entries is to be used.
attrs	A collection of strings indicating which attributes to return for each matching entry. Passing NULL for this parameter causes all available user attributes to be retrieved. The special constant string NO_ATTRS ("1.1") MAY be used as the only string in the array to indicate that no attribute types are to be returned by the server. The special constant string ALL_USER_ATTRS ("*") can be used in the attrs array along with the names of some operational attributes to indicate that all user attributes plus the listed operational attributes are to be returned.
attrsonly	A boolean value that must be zero if both attribute types and values are to be returned, and non-zero if only types are wanted.
res	This is a result parameter which will contain the results of the search upon completion of the call. If no results are returned, *res is set to NULL.

Return Values

Table 6–20 SEARCH_S Function Return Value

Value	Description
PLS_INTEGER (function return)	DBMS_LDAP.SUCCESS if the search operation succeeded. An exception is raised in all other cases.
res (OUT parameter)	If the search succeeded and there are entries, this parameter is set to a NON-NULL value which can be used to iterate through the result set.

Exceptions

Table 6–21 SEARCH_S Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_search_scope	Raised if the search scope is not one of SCOPE_BASE, SCOPE_ONELEVEL, or SCOPE_SUBTREE.
general_error	For all other errors. The error string associated with this exception will explain the error in detail.

Usage Notes

The function search_s() issues a search operation and does not return control to the user environment until all of the results have been returned from the server. Entries returned from the search (if any) are contained in the res parameter. This parameter is opaque to the caller. Entries, attributes, values, etc., can be extracted by calling the parsing routines described below.

See Also

DBMS_LDAP.search_st(), DBMS_LDAP.first_entry(), DBMS_LDAP.next_entry.

FUNCTION search_st

The function search_st performs a synchronous search in the LDAP server with a client-side timeout. It returns control to the PL/SQL environment only after all of the search results have been sent by the server or if the search request is timed-out by the client or the server.

Syntax

```
FUNCTION search_st  (ld       IN  SESSION,
                        base     IN  VARCHAR2,
                        scope    IN  PLS_INTEGER,
                        filter   IN  VARCHAR2,
                        attrs    IN  STRING_COLLECTION,
                        attronly IN  PLS_INTEGER,
                        tv       IN  TIMEVAL,
                        res      OUT MESSAGE)
     RETURN PLS_INTEGER;
```

Parameters

Table 6–22 SEARCH_ST Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
base	The dn of the entry at which to start the search.
scope	One of SCOPE_BASE (0x00), SCOPE_ONELEVEL (0x01), or SCOPE_SUBTREE (0x02), indicating the scope of the search.
filter	A character string representing the search filter. The value NULL can be passed to indicate that the filter "(objectclass=*)" which matches all entries is to be used.
attrs	A collection of strings indicating which attributes to return for each matching entry. Passing NULL for this parameter causes all available user attributes to be retrieved. The special constant string NO_ATTRS ("1.1") may be used as the only string in the array to indicate that no attribute types are to be returned by the server. The special constant string ALL_USER_ ATTRS ("*") can be used in the attrs array along with the names of some operational attributes to indicate that all user attributes plus the listed operational attributes are to be returned.
attrsonly	A boolean value that must be zero if both attribute types and values are to be returned, and non-zero if only types are wanted.
tv	The timeout value expressed in se conds and microseconds that should be used for this search.
res	This is a result parameter which will contain the results of the search upon completion of the call. If no results are returned, *res is set to NULL.

Return Values

Table 6–23 SEARCH_ST Function Return Values

Value	Description
PLS_INTEGER (function return)	DBMS_LDAP.SUCCESS if the search operation succeeded. An exception is raised in all other cases.
res (OUT parameter)	If the search succeeded and there are entries, this parameter is set to a NON_NULL value which can be used to iterate through the result set.

Exceptions

Table 6–24 SEARCH_ST Function Exceptions

Exception	Description
invalid_session	Raised if the session handle "ld" is invalid.
invalid_search_scope	Raised if the search scope is not one of SCOPE_BASE, SCOPE_ONELEVEL or SCOPE_SUBTREE.
invalid_search_time_value	Raised if the time value specified for the timeout is invalid.
general_error	For all other errors. The error string associated with this exception will explain the error in detail.

Usage Notes

This function is very similar to DBMS_LDAP.search_s() except that it requires a timeout value to be given.

See Also

DBMS_LDAP.search_s(), DBML_LDAP.first_entry(), DBMS_LDAP.next_entry.

FUNCTION first_entry

The function first_entry is used to retrieve the first entry in the result set returned by either search_s() or search_st()

Syntax

```
FUNCTION first_entry (ld  IN SESSION,
                          msg IN MESSAGE )
      RETURN MESSAGE;
```

Parameters

Table 6–25 FIRST_ENTRY Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
msg	The search result, as obtained by a call to one of the synchronous search routines.

Return Values

Table 6–26 FIRST_ENTRY Return Values

Value	Description
MESSAGE (function return)	A handle to the first entry in the list of entries returned from the LDAP server. It is set to NULL if there was an error and an exception is raised.

Exceptions

Table 6–27 FIRST_ENTRY Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_message	Raised if the incoming msg handle is invalid.

Usage Notes

The function first_entry() should always be the first function used to retrieve the results from a search operation.

See Also

DBMS_LDAP.next_entry(), DBMS_LDAP.search_s(), DBMS_LDAP.search_st()

FUNCTION next_entry

The function next_entry() is used to iterate to the next entry in the result set of a search operation.

Syntax

```
FUNCTION next_entry (ld  IN SESSION,
                         msg IN MESSAGE )
      RETURN MESSAGE;
```

Parameters

Table 6–28 NEXT_ENTRY Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
msg	The search result, as obtained by a call to one of the synchronous search routines.

Return Values

Table 6–29 NEXT_ENTRY Function Return Values

Value	Description
MESSAGE	A handle to the next entry in the list of entries returned from the LDAP server. It is set to null if there was an error and an exception is raised.

Exceptions

Table 6–30 NEXT_ENTRY Function Exceptions

Exception	Description
invalid_session	Raised if the session handle, ld is invalid.
invalid_message	Raised if the incoming msg handle is invalid.

Usage Notes

The function next_entry() should always be called after a call to the function first_entry(). Also, the return value of a successful call to next_entry() should be used as 'msg' argument used in a subsequent call to the function next_entry() to fetch the next entry in the list.

See Also

DBMS_LDAP.first_entry(), DBMS_LDAP.search_s(), DBMS_LDAP.search_st()

FUNCTION count_entries

This function is used to count the number of entries in the result set. It can also be used to count the number of entries remaining during a traversal of the result set using a combination of the functions first_entry() and next_entry().

Syntax

```
FUNCTION count_entries (ld  IN SESSION,
                            msg IN MESSAGE )
      RETURN PLS_INTEGER;
```

Parameters

Table 6–31 COUNT_ENTRY Function Parameters

Parameter	Description
ld	A valid LDAP session handle
msg	The search result, as obtained by a call to one of the synchronous search routines

Return Values

Table 6–32 COUNT_ENTRY Function Return Values

Value	Description
PLS INTEGER (function return)	Non-zero if there are entries in the result set -1 if there was a problem.

Exceptions

Table 6–33 COUNT_ENTRY Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_message	Raised if the incoming msg handle is invalid.
count_entry_error	Raised if there was a problem in counting the entries.

Usage Notes

`count_entries()` returns the number of entries contained in a chain of entries; if an error occurs such as the res parameter being invalid, -1 is returned. The `count_`

`entries()` call can also be used to count the number of entries that remain in a chain if called with a message, entry or reference returned by `first_message()`, `next_message()`, `first_entry()`, `next_entry()`, `first_reference()`, `next_reference()`.

See Also

DBMS_LDAP.first_entry(), DBMS_LDAP.next_entry().

FUNCTION first_attribute

The function first_attribute() fetches the first attribute of a given entry in the result set.

Syntax

```
FUNCTION first_attribute     (ld       IN  SESSION,
                              msg      IN  MESSAGE,
                              ber_elem OUT BER_ELEMENT)
      RETURN VARCHAR2;
```

Parameters

Table 6–34 FIRST_ATTRIBUTE Function Parameter

Parameter	Description
ld	A valid LDAP session handle
msg	The entry whose attributes are to be stepped through, as returned by `first_entry()` or `next_entry()`
ber_elem	A handle to a BER ELEMENT that is used to keep track of which attribute in the entry has been read

Return Values

Table 6–35 FIRST_ATTRIBUTE Function Return Values

Value	Description
VARCHAR2 (function return)	The name of the attribute if it exists. NULL if no attribute exists or if an error occurred.
ber_elem	A handle used by DBMS_LDAP.next_attribute() to iterate over all of the attributes

Exceptions

Table 6–36 FIRST_ATTRIBUTE Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_message	Raised if the incoming msg handle is invalid

Usage Notes

The handle to the BER_ELEMENT returned as a function parameter to first_attribute() should be used in the next call to next_attribute() to iterate through the various attributes of an entry. The name of the attribute returned from a call to first_attribute() can in turn be used in calls to the functions get_values() or get_values_len() to get the values of that particular attribute.

See Also

DBMS_LDAP.next_attribute(), DBMS_LDAP.get_values(), DBMS_LDAP.get_values_len(), DBMS_LDAP.first_entry(), DBMS_LDAP.next_entry().

FUNCTION next_attribute

The function next_attribute() fetches the next attribute of a given entry in the result set.

Syntax

```
FUNCTION next_attribute (    ld       IN SESSION,
                             msg      IN MESSAGE,
                             ber_elem IN BER_ELEMENT)
      RETURN VARCHAR2;
```

Parameters

Table 6–37 NEXT_ATTRIBUTE Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
msg	The entry whose attributes are to be stepped through, as returned by `first_entry()` or `next_entry()`.
ber_elem	A handle to a BER ELEMENT that is used to keep track of which attribute in the entry has been read.

Return Values

Table 6–38 NEXT_ATTRIBUTE Function Return Values

Value	Description
VARCHAR2 (function return)	The name of the attribute if it exists.

Exceptions

Table 6–39 NEXT_ATTRIBUTE Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_message	Raised if the incoming msg handle is invalid.

Usage Notes

The handle to the BER_ELEMENT returned as a function parameter to first_attribute() should be used in the next call to next_attribute() to iterate through the

various attributes of an entry. The name of the attribute returned from a call to next_attribute() can in turn be used in calls to the functions get_values() or get_values_len() to get the values of that particular attribute.

See Also

DBMS_LDAP.first_attribute(), DBMS_LDAP.get_values(), DBMS_LDAP.get_values_len(), DBMS_LDAP.first_entry(), DBMS_LDAP.next_entry().

FUNCTION get_dn

The function get_dn() retrieves the X.500 distinguished name of given entry in the result set.

The function first_attribute() fetches the first attribute of a given entry in the result set

Syntax

```
FUNCTION get_dn(    ld  IN SESSION,
                   msg IN MESSAGE)
     RETURN VARCHAR2;
```

Parameters

Table 6–40 GET_DN Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
msg	The entry whose dn is to be returned.

Return Values

Table 6–41 GET_DN Function Return Values

Value	Description
VARCHAR2 (function return)	The X.500 distinguished name of the entry as a PL/SQL string. NULL if there was a problem.

Exceptions

Table 6–42 GET_DN Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_message	Raised if the incoming msg handle is invalid.
get_dn_error	Raised if there was a problem in determining the dn

Usage Notes

The function get_dn() can be used to retrieve the dn of an entry as the program logic is iterating through the result set. This can in turn be used as an input to explode_dn() to retrieve the individual components of the dn.

See Also

DBMS_LDAP.explode_dn().

FUNCTION get_values

The function get_values() can be used to retrieve all of the values associated for a given attribute in a given entry.

Syntax

```
FUNCTION get_values(    ld   IN SESSION,
                        ldapentry IN MESSAGE,
                        attr IN VARCHAR2)
      RETURN STRING_COLLECTION;
```

Parameters

Table 6–43 GET_VALUES Function Parameters

Parameter	Description
ld	A valid LDAP session handle
ldapentry	A valid handle to an entry returned from a search result
attr	The name of the attribute for which values are being sought

Return Values

Table 6–44 GET_VALUES Function Return Values

Value	Description
STRING_COLLECTION (function return)	A PL/SQL string collection containing all of the values of the given attribute NULL if there are no values associated with the given attribute

Exceptions

Table 6–45 GET_VALUES Function Exceptions

Exception	Description
invalid session	Raised if the session handle ld is invalid.
invalid message	Raised if the incoming "entry handle' is invalid.

Usage Notes

The function get_values() can only be called after the handle to entry has been first retrieved by call to either first_entry() or next_entry(). The name of the attribute may be known beforehand or can also be determined by a call to first_attribute() or

next_attribute().The function get_values() always assumes that the data-type of the attribute it is retrieving is String. For retrieving binary data-types, get_values_len() should be used.

See Also

DBMS_LDAP.first_entry(), DBMS_LDAP.next_entry(), DBMS_LDAP.count_values(), DBMS_LDAP.get_values_len().

FUNCTION get_values_len

The function get_values_len() can be used to retrieve values of attributes that have a binary syntax.

Syntax

```
FUNCTION get_values_len(    ld   IN SESSION,
                            ldapentry IN MESSAGE,
                            attr IN VARCHAR2)
      RETURN BINVAL_COLLECTION;
```

Parameters

Table 6–46 GET_VALUES_LEN Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
ldapentrymsg	A valid handle to an entry returned from a search result.
attr	The string name of the attribute for which values are being sought.

Return Values

Table 6–47 GET_VALUES_LEN Function Return Values

Value	Description
BINVAL_COLLECTION (function return)	A PL/SQL raw collection containing all the values of the given attribute. NULL if there are no values associated with the given attribute.

Exceptions

Table 6–48 GET_VALUES_LEN Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_message	Raised if the incoming entry handle is invalid

Usage Notes

The function get_values_len() can only be called after the handle to entry has been first retrieved by call to either first_entry() or next_entry().The name of the attribute may be known beforehand or can also be determined by a call to first_attribute() or next_attribute().This function can be used to retrieve both binary and non-binary attribute values.

See Also

DBMS_LDAP.first_entry(), DBMS_LDAP.next_entry(), DBMS_LDAP.count_values_ len(), DBMS_LDAP.get_values().

FUNCTION delete_s

The function delete_s() can be used to remove a leaf entry in the LDAP Directory Information Tree.

Syntax

```
FUNCTION delete_s(ld      IN SESSION,
                  entrydn IN VARCHAR2)
RETURN PLS_INTEGER;
```

Table 6–49 DELETE_S Function Parameters

Parameter Name	Description
ld	A valid LDAP session
entrydn	The X.500 distinguished name of the entry to delete

Return Values

Table 6–50 DELETE_S Function Return Values

Value	Description
PLS_INTEGER (function return)	DBMS_LDAP.SUCCESS if the delete operation was successful. And exception is raised otherwise.

Exceptions

Table 6–51 DELETE_S Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_entry_dn	Raised if the distinguished name of the entry is invalid
general_error	For all other errors. The error string associated with this exception will explain the error in detail.

Usage Notes

The function delete_s() can be used to remove only leaf level entries in the LDAP DIT. A leaf level entry is an entry that does not have any children or LDAP entries under it. It cannot be used to delete non-leaf entries.

See Also

DBMS_LDAP.modrdn2_s()

FUNCTION modrdn2_s

The function modrdn2_s() can be used to rename the relative distinguished name of an entry.

Syntax

```
FUNCTION modrdn2_s (ld IN SESSION,
           entrydn in VARCHAR2
           newrdn in VARCHAR2
           deleteoldrdn IN PLS_INTEGER)
   RETURN PLS_INTEGER;
```

Parameters

Table 6–52 MODRDN2_S Function Parameters

Parameter	Description
ld	A valid LDAP session handle.
entrydn	The distinguished name of the entry. This entry must be a leaf node in the DIT.
newrdn	The new relative distinguished name of the entry.
deleteoldrdn	A boolean value that if non-zero indicates that the attribute values from the old name should be removed from the entry.

Return Values

Table 6–53 MODRDN2_S Function Return Values

Value	Description
PLS_INTEGER (function return)	DBMS_LDAP.SUCCESS if the operation was successful. An exception is raised otherwise.

Exceptions

Table 6–54 MODRDN2_S Function Exceptions

Exception	Description
invalid_session	Raised if the session handle ld is invalid.
invalid_entry_dn	Raised if the distinguished name of the entry is invalid.
invalid_rdn	Invalid LDAP RDN.
invalid_deleteoldrdn	Invalid LDAP deleteoldrdn.
general error	For all other errors. The error string associated with this exception will explain the error in detail.

Usage Notes

The function nodrdn2_s() can be used to rename the leaf nodes of a DIT. It simply changes the relative distinguished name by which they are known. The use of this function is being deprecated in the LDAP v3 standard. Please use rename_s() which can achieve the same foundation.

See Also

DBMS_LDAP.rename_s().

FUNCTION err2string

The function err2string() can be used to convert an LDAP error code to string in the local language in which the API is operating.

Syntax

```
FUNCTION err2string( ldap_err IN PLS_INTEGER )
    RETURN VARCHAR2;
```

Parameters

Table 6–55 ERR2STRING Function Parameters

Parameter	Description
ldap_err	An error number returned from one the API calls.

Return Values

Table 6–56 ERR2STRING Function Returrn Values

Value	Description
VARCHAR@ (function return)	A character string appropriately translated to the local language which describes the error in detail.

Exceptions

Table 6–57 ERR2STRING Function Exceptions

Exception	Description
N / A	None

Usage Notes

In this release, the exception handling mechanism automatically invokes this if any of the API calls encounter an error.

See Also

N / A

FUNCTION create_mod_array

The function create_mod_array() allocates memory for array modification entries that will be applied to an entry using the modify_s() or add_s() functions.

Syntax

```
FUNCTION create_mod_array (num IN PLS_INTEGER)
    RETURN MOD_ARRAY;
```

Parameters

Table 6–58 CREATE_MOD_ARRAY Function Parameters

Parameter	Description
num	The number of the attributes that you want to add or modify.

Return Values

Table 6–59 CREATE_MOD_ARRAY Function Return Values

Value	Description
MOD_ARRAY (function return)	The data structure holds a pointer to an LDAP mod array. NULL if there was a problem.

Exceptions

Table 6–60 CREATE_MOD_ARRAY Function Exceptions

Exception	Description
N / A	No LDAP specific exception will be raised

Usage Notes

This function is one of the preparation steps for `DBMS_LDAP.add_s` and `DBMS_LDAP.modify_s`. It is required to call `DBMS_LDAP.free_mod_array` to free memory after the calls to add_s or modify_s have completed.

See Also

`DBMS_LDAP.populate_mod_array()`, `DBMS_LDAP.modify_s()`, `DBMS_LDAP.add_s()`, and `DBMS_LDAP.free_mod_array()`.

PROCEDURE populate_mod_array (String Version)

Populates one set of attribute information for add or modify operations.

Syntax

```
PROCEDURE populate_mod_array
    (modptr   IN DBMS_LDAP.MOD_ARRAY,
     mod_op   IN PLS_INTEGER,
     mod_type IN VARCHAR2,
     modval   IN DBMS_LDAP.STRING_COLLECTION);
```

Parameters

Table 6–61 POPULATE_MOD_ARRAY (String Version) Procedure Parameters

Parameter	Description
modptr	The data structure holds a pointer to an LDAP mod array.
Mod_op	This field specifies the type of modification to perform.
Mod_type	This field indicates the name of the attribute type to which the modification applies.
Modval	This field specifies the attribute values to add, delete, or replace. It is for the string values only.

Return Values

Table 6–62 POPULATE_MOD_ARRAY (String Version) Procedure Return Values

Value	Description
N / A	

Exceptions

Table 6–63 POPULATE_MOD_ARRAY (String Version) Procedure Exceptions

Exception	Description
invalid_mod_array	Invalid LDAP mod array
invalid_mod_option	Invalid LDAP mod option
invalid_mod_type	Invalid LDAP mod type
invalid_mod_value	Invalid LDAP mod value

Usage Notes

This function is one of the preparation steps for `DBMS_LDAP.add_s` and `DBMS_LDAP.modify_s`. It has to be called after `DBMS_LDAP.create_mod_array` is called.

See Also

`DBMS_LDAP.create_mod_array()`, `DBMS_LDAP.modify_s()`, `DBMS_LDAP.add_s()`, and `DBMS_LDAP.free_mod_array()`.

PROCEDURE populate_mod_array (Binary Version)

Populates one set of attribute information for add or modify operations. This procedure call has to happen after `DBMS_LDAP.create_mod_array()` is called.

Syntax

```
PROCEDURE populate_mod_array
    (modptr   IN DBMS_LDAP.MOD_ARRAY,
     mod_op   IN PLS_INTEGER,
     mod_type IN VARCHAR2,
     modval   IN DBMS_LDAP.BERVAL_COLLECTION);
```

Parameters

Table 6–64 POPULATE_MOD_ARRAY (Binary Version) Procedure Parameters

Parameter	Description
modptr	The data structure holds a pointer to an LDAP mod array
Mod_op	This field specifies the type of modification to perform
Mod_type	This field indicates the name of the attribute type to which the modification applies
Modval	This field specifies the attribute values to add, delete, or replace. It is for the binary values

Return Values

Table 6–65 POPULATE_MOD_ARRAY (Binary Version) Procedure Return Values

Value	Description
N / A	

Exceptions

Table 6–66 POPULATE_MOD_ARRAY (Binary Version) Procedure Exceptions

Exception	Description
invalid_mod_array	Invalid LDAP mod array
invalid_mod_option	Invalid LDAP mod option
invalid_mod_type	Invalid LDAP mod type
invalid_mod_value	Invalid LDAP mod value

Usage Notes

This function is one of the preparation steps for `DBMS_LDAP.add_s` and `DBMS_LDAP.modify_s`. It has to happen after `DBMS_LDAP.create_mod_array` is called.

See Also

`DBMS_LDAP.create_mod_array()`, `DBMS_LDAP.modify_s()`, `DBMS_LDAP.add_s()`, and `DBMS_LDAP.free_mod_array()`.

FUNCTION modify_s

Performs a synchronous modification of an existing LDAP directory entry.

Syntax

```
FUNCTION modify_s(ld      IN DBMS_LDAP.SESSION,
                  entrydn IN VARCHAR2,
                  modptr  IN DBMS_LDAP.MOD_ARRAY)
        RETURN PLS_INTEGER;
```

Parameters

Table 6–67 MODIFY_S Function Parameters

Parameter	Description
ld	This parameter is a handle to an LDAP session, as returned by a successful call to `DBMS_LDAP.init()`.
entrydn	This parameter specifies the name of the directory entry whose contents are to be modified.
modptr	This parameter is the handle to an LDAP mod structure, as returned by successful call to `DBMS_LDAP.create_mod_array()`.

Return Values

Table 6–68 MODIFY_S Function Return Values

Value	Description
PLS_INTEGER	The indication of the success or failure of the modification operation

Exceptions

Table 6–69 MODIFY_S Function Exceptions

Exception	Description
invalid_session	Invalid LDAP session
invalid_entry_dn	Invalid LDAP entry dn
invalid_mod_array	Invalid LDAP mod array

Usage Notes

This function call has to follow successful calls of `DBMS_LDAP.create_mod_array()` and `DBMS_LDAP.populate_mod_array()`.

See Also

DBMS_LDAP.create_mod_array(),DBMS_LDAP.populate_mod_array(), DBMS_LDAP.add_s(), and DBMS_LDAP.free_mod_array().

FUNCTION add_s

Adds a new entry to the LDAP directory synchronously. Before calling `add_s`, you have to call `DBMS_LDAP.create_mod_array()` and `DBMS_LDAP.populate_mod_array()`.

Syntax

```
FUNCTION add_s(ld      IN DBMS_LDAP.SESSION,
                  entrydn IN VARCHAR2,
                  modptr  IN DBMS_LDAP.MOD_ARRAY)
RETURN PLS_INTEGER;
```

Parameters

Table 6–70 ADD_S Function Parameters

Parameter	Description
ld	This parameter is a handle to an LDAP session, as returned by a successful call to `DBMS_LDAP.init()`.
Entrydn	This parameter specifies the name of the directory entry to be created.
Modptr	This parameter is the handle to an LDAP mod structure, as returned by successful call to `DBMS_LDAP.create_mod_array()`.

Return Values

Table 6–71 ADD_S Function Return Values

Value	Description
PLS_INTEGER	The indication of the success or failure of the modification operation.

Exceptions

Table 6–72 ADD_S Function Exceptions

Exception	Description
invalid_session	Invalid LDAP session.
invalid_entry_dn	Invalid LDAP entry dn.
invalid_mod_array	Invalid LDAP mod array.

Usage Notes

The parent entry of the entry to be added must already exist in the directory. This function call has to follow successful calls of `DBMS_LDAP.create_mod_array()` and `DBMS_LDAP.populate_mod_array()`.

See Also

`DBMS_LDAP.create_mod_array()`,`DBMS_LDAP.populate_mod_array()`, `DBMS_LDAP.modify_s()`, and `DBMS_LDAP.free_mod_array()`.

PROCEDURE free_mod_array

Frees the memory allocated by `DBMS_LDAP.create_mod_array()`.

Syntax

```
PROCEDURE free_mod_array(modptr IN DBMS_LDAP.MOD_ARRAY);
```

Parameters

Table 6–73 FREE_MOD_ARRAY Procedure Parameters

Parameter	Description
modptr	This parameter is the handle to an LDAP mod structure, as returned by successful call to `DBMS_LDAP.create_mod_array()`.

Return Values

Table 6–74 FREE_MOD_ARRAY Procedure Return Value

Value	Description
N / A	

Exceptions

Table 6–75 FREE_MOD_ARRAY Procedure Exceptions

Exception	Description
N / A	No LDAP specific exception will be raised.

Usage Notes

N / A

See Also

`DBMS_LDAP.populate_mod_array()`, `DBMS_LDAP.modify_s()`, `DBMS_LDAP.add_s()`, and `DBMS_LDAP.create_mod_array()`.

FUNCTION count_values

Counts the number of values returned by `DBMS_LDAP.get_values()`.

Syntax

```
FUNCTION count_values
          (values IN DBMS_LDAP.STRING_COLLECTION)
    RETURN PLS_INTEGER;
```

Parameters

Table 6–76 COUNT_VALUES Function Parameters

Parameter	Description
values	The collection of string values.

Return Values

Table 6–77 COUNT_VALUES Function Return Values

Value	Description
PLS_INTEGER	The indication of the success or failure of the operation.

Exceptions

Table 6–78 COUNT_VALUES Function Exceptions

Exception	Description
N / A	No LDAP specific exception will be raised.

Usage Notes

N / A

See Also

`DBMS_LDAP.count_values_len()`, `DBMS_LDAP.get_values()`.

FUNCTION count_values_len

Counts the number of values returned by `DBMS_LDAP.get_values_len()`.

Syntax

```
FUNCTION count_values_len (values IN DBMS_LDAP.BINVAL_COLLECTION)
    RETURN PLS_INTEGER;
```

Parameters

Table 6–79 COUNT_VALUES_LEN Function Parameters

Parameter	Description
values	The collection of binary values.

Return Values

Table 6–80 COUNT_VALUES_LEN Function Return Values

Value	Description
PLS_INTEGER	The indication of the success or failure of the operation.

Exceptions

Table 6–81 COUNT_VALUES_LEN Function Exceptions

Exception	Description
N / A	No LDAP specific exception will be raised.

Usage Notes

N / A

See Also

DBMS_LDAP.count_values(), DBMS_LDAP.get_values_len().

FUNCTION rename_s

Renames an LDAP entry synchronously.

Syntax

```
FUNCTION rename_s(ld           IN SESSION,
                  dn           IN VARCHAR2,
                  newrdn       IN VARCHAR2,
                  newparent    IN VARCHAR2,
                  deleteoldrdn IN PLS_INTEGER,
                  serverctrls  IN LDAPCONTROL,
                  clientctrls  IN LDAPCONTROL)
   RETURN PLS_INTEGER;
```

Parameters

Table 6–82 RENAME_S Function Parameters

Parameter	Description
ld	This parameter is a handle to an LDAP session, as returned by a successful call to `DBMS_LDAP.init()`.
Dn	This parameter specifies the name of the directory entry to be renamed or moved.
newrdn	This parameter specifies the new RDN.
Newparent	This parameter specifies the dn of the new parent.
Deleteoldrdn	This parameter specifies if the old RDN should be retained. If this value is 1, then the old RDN will be removed.
Serverctrls	Currently not supported.
Clientctrls	Currently not supported.

Return Values

Table 6–83 RENAME_S Function Return Values

Value	Description
PLS_INTEGER	The indication of the success or failure of the operation.

Exceptions

Table 6–84 RENAME_S Function Exceptions

Exception	Description
invalid_session	Invalid LDAP Session.
invalid_entry_dn	Invalid LDAP dn.
invalid_rdn	Invalid LDAP Rdn.
invalid_newparent	Invalid LDAP newparent.
invalid_deleteoldrdn	Invalid LDAP deleteoldrdn.

Usage Notes

N / A

See Also

`DBMS_LDAP.modrdn2_s()`.

FUNCTION explode_dn

Breaks a DN up into its components.

Syntax

```
FUNCTION explode_dn (dn      IN VARCHAR2,
                             notypes IN PLS_INTEGER)
    RETURN STRING_COLLECTION;
```

Parameters

Table 6–85 EXPLODE_DN Function Parameters

Parameter	Description
dn	This parameter specifies the name of the directory entry to be broken up.
Notypes	This parameter specifies if the attribute tags will be returned. If this value is not 0, then there will be no attribute tags will be returned.

Return Values

Table 6–86 EXPLODE_DN Function Return Values

Value	Description
STRING_COLLECTION	An array of strings. If the dn can not be broken up, NULL will be returned.

Exceptions

Table 6–87 EXPLODE_DN Function Exceptions

Exception	Description
invalid_entry_dn	Invalid LDAP dn.
invalid_notypes	Invalid LDAP notypes value.

Usage Notes

N/A

See Also

`DBMS_LDAP.get_dn()`.

FUNCTION open_ssl

Establishes an SSL(Secure Sockets Layer) connection over an existing LDAP connection.

Syntax

```
FUNCTION open_ssl(ld              IN SESSION,
    sslwrl          IN VARCHAR2,
    sslwalletpasswd IN VARCHAR2,
    sslauth         IN PLS_INTEGER)
RETURN PLS_INTEGER;
```

Parameters

Table 6–88 OPEN_SSL Function Parameters

Parameter	Description
ld	This parameter is a handle to an LDAP session, as returned by a successful call to `DBMS_LDAP.init()`.
Sslwrl	This parameter specifies the wallet location (Required for one-way or two-way SSL connection.)
sslwalletpasswd	This parameter specifies the wallet password (Required for one-way or two-way SSL connection.)
sslauth	This parameter specifies the SSL Authentication Mode (1 for no authentication required, 2 for one way authentication required, 3 for two way authentication required.

Return Values

Table 6–89 OPEN_SSL Function Return Values

Value	Description
PLS_INTEGER	The indication of the success or failure of the operation.

Exceptions

Table 6–90 OPEN_SSL Function Exceptions

Exception	Description
invalid_session	Invalid LDAP Session.
invalid_ssl_wallet_loc	Invalid LDAP SSL wallet location.
invalid_ssl_wallet_ passwd	Invalid LDAP SSL wallet passwd.
invalid_ssl_auth_mode	Invalid LDAP SSL authentication mode.

Usage Notes

Need to call `DBMS_LDAP.init()` first to acquire a valid LDAP session.

See Also

`DBMS_LDAP.init()`.

DBMS_OBFUSCATION_TOOLKIT

The DBMS_OBFUSCATION_TOOLKIT package allows an application to encrypt data using either the Data Encryption Standard (DES) or the Triple DES algorithms.

The Data Encryption Standard (DES), also known as the Data Encryption Algorithm (DEA) by the American National Standards Institute (ANSI) and DEA-1 by the International Standards Organization (ISO), has been a worldwide encryption standard for over twenty years. The banking industry has also adopted DES-based standards for transactions between private financial institutions, and between financial institutions and private individuals.

DES is a symmetric key cipher; that is, the same key is used to encrypt data as well as decrypt data. DES encrypts data in 64-bit blocks using a 56-bit key. The DES algorithm ignores 8 bits of the 64-bit key that is supplied; however, developers must supply a 64-bit key to the algorithm.

Triple DES (3DES) is a far stronger cipher than DES; the resulting ciphertext (encrypted data) is much harder to break using an exhaustive search: 2**112 or 2**168 attempts instead of 2**56 attempts. Triple DES is also not as vulnerable to certain types of cryptanalysis as is DES.

The DES procedures are the following:

- DESEncrypt Procedure
- DESDecrypt Procedure

Oracle installs this package in the SYS schema. You can then grant access the package to existing users and roles as needed. The package also grants access to the PUBLIC role so no explicit grant needs to be done.

Overview of Key Management

Key management, including both generation and secure storage of cryptographic keys, is one of the most important aspects of encryption. If keys are poorly chosen or stored improperly, then it is far easier for a malefactor to break the encryption. Rather than using an exhaustive key search attack (that is, cycling through all the possible keys in hopes of finding the correct decryption key), cryptanalysts typically seek weaknesses in the choice of keys, or the way in which keys are stored.

Key generation is an important aspect of encryption. Typically, keys are generated automatically through a random-number generator. Provided that the random number generation is cryptographically secure, this can be an acceptable form of key generation. However, if random numbers are not cryptographically secure, but

have elements of predictability, the security of the encryption may be easily compromised.

The DBMS_OBFUSCATION_TOOLKIT package does not generate encryption keys nor does it maintain them. Care must be taken by the application developer to ensure the secure generation and storage of encryption keys used with this package. Furthermore, the encryption and decryption done by the DBMS_OBFUSCATION_TOOLKIT takes place on the server, not the client. If the key is passed over the connection between the client and the server, the connection must be protected using Oracle Advanced Security, otherwise the key is vulnerable to capture over the wire.

Key storage is one of the most important, yet difficult aspects of encryption and one of the hardest to manage properly. To recover data encrypted with a symmetric key, the key must be accessible to the application or user seeking to decrypt data. The key needs to be easy enough to retrieve that users can access encrypted data when they need to without significant performance degradation. The key also needs to be secure enough that it is not easily recoverable by unauthorized users trying to access encrypted data they are not supposed to see.

The three options available to a developer are:

- store the key in the database
- store the key in the operating system
- have the user manage the key

Store the Key in the Database

Storing the keys in the database cannot always provide “bullet-proof” security if you are trying to protect data against the DBA accessing encrypted data (since an all-privileged DBA could access tables containing encryption keys), but it can provide security against the casual snooper, or against someone compromising the database files on the operating system. Furthermore, the security you can obtain by storing keys in the database does not have to be bullet-proof in order to be extremely useful.

For example, suppose you want to encrypt an employee’s security number, one of the columns in table EMP. You could encrypt each employee’s security number using a key which is stored in a separate column in EMP. However, anyone with SELECT access on the EMP table could retrieve the encryption key and decrypt the matching social security number. Alternatively, you could store the encryption keys in another table, and use a package to retrieve the correct key for the encrypted data item, based on a primary key-foreign key relationship between the tables.

A developer could envelope both the DBMS_OBFUSCATION_TOOLKIT package and the procedure to retrieve the encryption keys supplied to the package. Furthermore, the encryption key itself could be transformed in some way (for example, XORed with the foreign key to the EMP table) so that the key itself is not stored in easily recoverable form.

Oracle recommends using the wrap utility of PL/SQL to obfuscate the code within a PL SQL package itself that does the encryption. That prevents people from breaking the encryption by looking at the PL/SQL code that handles keys, calls encrypting routines, etc.. In other words, use the wrap utility to obfuscate the PL/SQL packages themselves.

This scheme is secure enough to prevent users with SELECT access to EMP from reading unencrypted sensitive data, and a DBA from easily retrieving encryption keys and using them to decrypt data in the EMP table. It can be made more secure by changing encryption keys regularly, or having a better key storage algorithm (so the keys themselves are encrypted, for example).

Storing the Key in the Operating System

Storing keys in the operating system (e.g. in a flat file) is another option. Oracle8*i* allows you to make callouts from PL/SQL, which you could use to retrieve encryption keys. If you store keys in the O/S and make callouts to retrieve the keys, then the security of your encrypted data is only as secure as the protection of the key file on the O/S. Of course, a user retrieving keys from the operating system would have to be able to either access the Oracle database files (to decrypt encrypted data), or be able to gain access to the table in which the encrypted data is stored as a legitimate user.

User Managed Keys

If you ask user to supply the key, it is crucial that you use network encryption, such as that provided by Oracle Advanced Security, so the key is not passed from client to server in the clear. Furthermore, you have to rely on the user to remember the key, or your data is nonrecoverable.

DESEncrypt Procedure

The DESEncrypt procedure generates the encrypted form of the input data. An example of the DESEncrypt procedure appears at the end of this chapter.

The DES algorithm encrypts data in 64-bit blocks using a 56-bit key. The DES algorithm throws away 8 bits of the supplied key (the particular bits which are thrown away is beyond the scope of this documentation). However, developers using the algorithm must supply a 64-bit key or the package will raise an error.

Parameter Descriptions

Table 6–91 and Table 6–92 list the parameters for the DESEncrypt syntax as well as their modes, types, and descriptions.

Table 6–91 DESEncrypt parameters for raw data

Parameter Name	Mode	Type	Description
input	IN	RAW	data to be encrypted
key	IN	RAW	encryption key
encrypted_data	OUT	RAW	encrypted data

Table 6–92 DESEncrypt parameters for string data

Parameter Name	Mode	Type	Description
input_string	IN	VARCHAR2	string to be encrypted
key_string	IN	VARCHAR2	encryption key string
encrypted_string	OUT	VARCHAR2	encrypted string

If the input data or key given to the PL/SQL DESEncrypt procedure is empty, then the procedure raises the error ORA-28231 "Invalid input to Obfuscation toolkit".

If the input data given to the DESEncrypt procedure is not a multiple of 8 bytes, the procedure raises the error ORA-28232 "Invalid input size for Obfuscation toolkit".

If the user tries to double encrypt data using the DESEncrypt procedure, then the procedure raises the error ORA-28233 "Double encryption not supported".

Encryption Procedure Restriction

The DESEncryption procedure has two restrictions. The first is that the DES key length for encryption is fixed at 56 bits; you cannot alter this key length.

The second is that you cannot execute multiple passes of encryption. That is, you cannot re-encrypt previously encrypted data by calling the function twice.

Note: Both the key length limitation and the prevention of multiple encryption passes are requirements of US regulations governing the export of cryptographic products.

DESDecrypt Procedure

The purpose of the DESDecrypt procedure is to generate the decrypted form of the input data. An example of the DESDecrypt procedure appears at the end of this chapter.

Parameter Descriptions

Table 6–93 and Table 6–94 list the parameters for the DESDecrypt syntax, their modes, types, and descriptions.

Table 6–93 DESDecrypt parameters for raw data

Parameter Name	Mode	Type	Description
input	IN	RAW	Data to be decrypted
key	IN	RAW	Decryption key
decrypted_data	OUT	RAW	Decrypted data

Table 6–94 DESDecrypt parameters for string data

Parameter Name	Mode	Type	Description
input_string	IN	VARCHAR2	String to be decrypted
key_string	IN	VARCHAR2	Decryption key string
decrypted_string	OUT	VARCHAR2	Decrypted string

If the input data or key given to the PL/SQL DESDecrypt function is empty, then Oracle raises ORA error 28231 "Invalid input to Obfuscation toolkit".

If the input data given to the DESDecrypt function is not a multiple of 8 bytes, Oracle raises ORA error 28232 "Invalid input size for Obfuscation toolkit".

Note: ORA-28233 is not applicable for the DESDecrypt function.

DESDecryption Procedure Restriction

The DES key length for encryption is fixed at 64 bits (of which 56 bits are used); you cannot alter this key length.

Note: The key length limitation is a requirement of US regulations governing the export of cryptographic products.

DES Encryption Code Example

Following is a sample PL/SQL program for your reference. Segments of the code are numbered and contain narrative text explaining portions of the code.

```
DECLARE
   input_string        VARCHAR2(16) := 'tigertigertigert';
   raw_input           RAW(128) := UTL_RAW.CAST_TO_RAW(input_string);
   key_string          VARCHAR2(8)  := 'scottsco';
   raw_key             RAW(128) := UTL_RAW.CAST_TO_RAW(key_string);
   encrypted_raw               RAW(2048);
   encrypted_string            VARCHAR2(2048);
   decrypted_raw               RAW(2048);
   decrypted_string            VARCHAR2(2048);
   error_in_input_buffer_length EXCEPTION;
   PRAGMA EXCEPTION_INIT(error_in_input_buffer_length, -28232);
   INPUT_BUFFER_LENGTH_ERR_MSG VARCHAR2(100) :=
    '*** DES INPUT BUFFER NOT A MULTIPLE OF 8 BYTES - IGNORING
EXCEPTION ***';
   double_encrypt_not_permitted EXCEPTION;
   PRAGMA EXCEPTION_INIT(double_encrypt_not_permitted, -28233);
   DOUBLE_ENCRYPTION_ERR_MSG VARCHAR2(100) :=
    '*** CANNOT DOUBLE ENCRYPT DATA - IGNORING EXCEPTION ***';

-- 1. Begin testing raw data encryption and decryption
   BEGIN
   dbms_output.put_line('> ========= BEGIN TEST RAW DATA =========');
   dbms_output.put_line('> Raw_input                    : ' ||
                 UTL_RAW.CAST_TO_VARCHAR2(raw_input));
   BEGIN
      dbms_obfuscation_toolkit.DESEncrypt(input => raw_input,
                key => raw_key, encrypted_data => encrypted_raw );
      dbms_output.put_line('> encrypted hex value          : ' ||
                rawtohex(encrypted_raw));
      dbms_obfuscation_toolkit.DESDecrypt(input => encrypted_raw,
                key => raw_key, decrypted_data => decrypted_raw);
      dbms_output.put_line('> Decrypted raw output         : ' ||
                   UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw));
      dbms_output.put_line('>  ');
      if UTL_RAW.CAST_TO_VARCHAR2(raw_input) =
                   UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw) THEN
         dbms_output.put_line('> Raw DES Encyption and Decryption successful');
```

```
      END if;
   EXCEPTION
      WHEN error_in_input_buffer_length THEN
             dbms_output.put_line('> ' || INPUT_BUFFER_LENGTH_ERR_MSG);
   END;
   dbms_output.put_line('>  ');


-- 2. Begin testing string data encryption and decryption
   dbms_output.put_line('> ========= BEGIN TEST STRING DATA =========');

   BEGIN
      dbms_output.put_line('> input string                    : '
                           || input_string);
      dbms_obfuscation_toolkit.DESEncrypt(
               input_string => input_string,
               key_string => key_string,
               encrypted_string => encrypted_string );
      dbms_output.put_line('> encrypted hex value             : ' ||
                  rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_string)));
      dbms_obfuscation_toolkit.DESDecrypt(
               input_string => encrypted_string,
               key_string => key_string,
               decrypted_string => decrypted_string );
      dbms_output.put_line('> decrypted string output         : ' ||
                 decrypted_string);
      if input_string = decrypted_string THEN
         dbms_output.put_line('> String DES Encyption and Decryption success-
ful');
      END if;
   EXCEPTION
      WHEN error_in_input_buffer_length THEN
             dbms_output.put_line('  ' || INPUT_BUFFER_LENGTH_ERR_MSG);
   END;
   dbms_output.put_line('>  ');
```

DES3Encrypt Procedure

The DES3Encrypt procedure generates the encrypted form of the input data by passing it through the Triple DES encryption algorithm. An example of the DESEncrypt procedure appears at the end of this chapter.

Oracle's implementation of 3DES supports either a 2-key or 3-key implementation, in outer cipher-block-chaining (CBC) mode.

A developer choosing to use Oracle's 3DES interface with a 2-key implementation must supply a *single* key of 128 bits as an argument to the DES3Encrypt procedure. Oracle then breaks the supplied key into two 64-bit keys. As with DES, the 3DES algorithm throws away 8 bits of each derived key (the particular bits which are thrown away is beyond the scope of this documentation). However, developers using the algorithm must supply a single 128-bit key for the 2-key 3DES implementation or the package will raise an error. The DES3Encrypt procedure uses the 2-key implementation by default.

A developer using Oracle’s 3DES interface with a 3-key implementation must supply a *single* key of 192 bits as an argument to the DES3Encrypt procedure. Oracle then breaks the supplied key into three 64-bit keys. As with DES, the 3DES algorithm throws away 8 bits of each derived key (the particular bits which are thrown away is beyond the scope of this documentation). However, developers using the algorithm must supply a single 192-bit key for the 3-key 3DES implementation or the package will raise an error.

Parameter Descriptions

Table 6–95 and Table 6–96 list the parameters for the DES3Encrypt syntax, their modes, types, and descriptions.

Table 6–95 DES3Encrypt parameters for raw data

Parameter Name	Mode	Type	Description
input	IN	RAW	data to be encrypted
key	IN	RAW	encryption key
encrypted_data	OUT	RAW	encrypted data

Table 6–96 DES3Encrypt parameters for string data

Parameter Name	Mode	Type	Description
input_string	IN	VARCHAR2	string to be encrypted

Table 6–96 DES3Encrypt parameters for string data

Parameter Name	Mode	Type	Description
key_string	IN	VARCHAR2	encryption key string
encrypted_string	OUT	VARCHAR2	encrypted string

If the input data or key given to the PL/SQL DES3Encrypt procedure is empty, then the procedure raises the error ORA-28231 "Invalid input to Obfuscation toolkit".

If the input data given to the DES3Encrypt procedure is not a multiple of 8 bytes, the procedure raises the error ORA-28232 "Invalid input size for Obfuscation toolkit".

If the user tries to double encrypt data using the DES3Encrypt procedure, then the procedure raises the error ORA-28233 "Double encryption not supported".

Encryption Procedure Restriction

The DES3Encrypt procedure has two restrictions. The first is that the DES key length for encryption is fixed at 128 bits (for 2-key DES) or 192 bits (for 3-key DES); you cannot alter these key lengths.

The second is that you cannot execute multiple passes of encryption using 3DES. The 3DES algorithm itself encrypts data multiple times; however, you cannot call the 3DESencrypt function itself more than once to encrypt the same data using 3DES.

Note: Both the key length limitation and the prevention of multiple encryption passes are requirements of US regulations governing the export of cryptographic products.

DES3Decrypt Procedure

The purpose of the DES3Decrypt procedure is to generate the decrypted form of the input data. An example of the DES3Decrypt procedure appears at the end of this chapter.

Parameter Descriptions

Table 6–97 and Table 6–98 list the parameters for the DES3Decrypt syntax, their modes, types, and descriptions.

Table 6–97 DES3Decrypt parameters for raw data

Parameter Name	Mode	Type	Description
input	IN	RAW	Data to be decrypted
key	IN	RAW	Decryption key
decrypted_data	OUT	RAW	Decrypted data

Table 6–98 DES3Decrypt parameters for string data

Parameter Name	Mode	Type	Description
input_string	IN	VARCHAR2	String to be decrypted
key_string	IN	VARCHAR2	Decryption key string
decrypted_string	OUT	VARCHAR2	Decrypted string

If the input data or key given to the DES3Decrypt procedure is empty, then the procedure raises the error ORA-28231 "Invalid input to Obfuscation toolkit".

If the input data given to the DES3Decrypt procedure is not a multiple of 8 bytes, the procedure raises the error ORA-28232 "Invalid input size for Obfuscation toolkit".

Note: ORA-28233 is NOT applicable for the DES3Decrypt function.

DES3Decrypt Procedure Restriction

As stated above, a developer must supply a single key of either 128 bits for a 2-key implementation (of which only 112 are used), or a single key of 192 bits for a 3-key implementation (of which 168 bits are used). Oracle automatically truncates the

supplied key into 56-bit lengths for decryption. This keylength is fixed and cannot be altered.

Note: Both the key length limitation and the prevention of multiple encryption passes are requirements of US regulations governing the export of cryptographic products.

DES3 Encryption Code Example

Following is a sample PL/SQL program for your reference. Segments of the code are numbered and contain narrative text explaining portions of the code.

```
DECLARE
   input_string        VARCHAR2(16) := 'tigertigertigert';
   raw_input           RAW(128) := UTL_RAW.CAST_TO_RAW(input_string);
   key_string          VARCHAR2(8)  := 'scottsco';
   raw_key             RAW(128) := UTL_RAW.CAST_TO_RAW(key_string);
encrypted_raw               RAW(2048);
   encrypted_string            VARCHAR2(2048);
decrypted_raw               RAW(2048);
   decrypted_string            VARCHAR2(2048);
   error_in_input_buffer_length EXCEPTION;
   PRAGMA EXCEPTION_INIT(error_in_input_buffer_length, -28232);
   INPUT_BUFFER_LENGTH_ERR_MSG VARCHAR2(100) :=
    '*** DES INPUT BUFFER NOT A MULTIPLE OF 8 BYTES - IGNORING EXCEPTION ***';
   double_encrypt_not_permitted EXCEPTION;
   PRAGMA EXCEPTION_INIT(double_encrypt_not_permitted, -28233);
   DOUBLE_ENCRYPTION_ERR_MSG VARCHAR2(100) :=
    '*** CANNOT DOUBLE ENCRYPT DATA - IGNORING EXCEPTION ***';

-- 1. Begin testing raw data encryption and decryption
   BEGIN
   dbms_output.put_line('> ========= BEGIN TEST RAW DATA =========');
   dbms_output.put_line('> Raw input                        : ' ||
                 UTL_RAW.CAST_TO_VARCHAR2(raw_input));
   BEGIN
      dbms_obfuscation_toolkit.DES3Encrypt(input => raw_input,
               key => raw_key, encrypted_data => encrypted_raw );
      dbms_output.put_line('> encrypted hex value              : ' ||
               rawtohex(encrypted_raw));
```

```
      dbms_obfuscation_toolkit.DES3Decrypt(input => encrypted_raw,
               key => raw_key, decrypted_data => decrypted_raw);
      dbms_output.put_line('> Decrypted raw output             : ' ||
                    UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw));
      dbms_output.put_line('>  ');
      if UTL_RAW.CAST_TO_VARCHAR2(raw_input) =
                    UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw) THEN
         dbms_output.put_line('> Raw DES3 Encyption and Decryption successful');
      END if;
   EXCEPTION
      WHEN error_in_input_buffer_length THEN
            dbms_output.put_line('> ' || INPUT_BUFFER_LENGTH_ERR_MSG);
   END;
   dbms_output.put_line('>  ');

-- 2. Begin testing string data encryption and decryption
   dbms_output.put_line('> ========= BEGIN TEST STRING DATA =========');

   BEGIN
      dbms_output.put_line('> input string                     : '
                           || input_string);
      dbms_obfuscation_toolkit.DES3Encrypt(
               input_string => input_string,
               key_string => key_string,
               encrypted_string => encrypted_string );
      dbms_output.put_line('> encrypted hex value              : ' ||
                  rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_string)));
      dbms_obfuscation_toolkit.DES3Decrypt(
               input_string => encrypted_string,
               key_string => key_string,
               decrypted_string => decrypted_string );
      dbms_output.put_line('> decrypted string output          : ' ||
                 decrypted_string);
      if input_string = decrypted_string THEN
         dbms_output.put_line('> String DES3 Encyption and Decryption
successful');
      END if;
   EXCEPTION
      WHEN error_in_input_buffer_length THEN
            dbms_output.put_line(' ' || INPUT_BUFFER_LENGTH_ERR_MSG);
   END;
   dbms_output.put_line('>  ');

/
```

7

Recovery Manager

This chapter describes new and changed features for RMAN in Oracle8*i* Release 3 (8.1.7). This chapter contains these topics:

- Preparing a Standby Database Using RMAN
- Creating a Standby Database Using RMAN
- Backing Up Files at the Standby Site Using RMAN
- Performing a Test Backup Using RMAN
- Setting the Default Location of the RMAN Snapshot Control File
- Crosschecking and Deleting on Multiple RMAN Channels
- Connecting RMAN to a Target Database in an OPS Cluster
- Selected RMAN Maintenance Commands No Longer Require a Catalog
- Troubleshooting RMAN Character Set Errors: Scenario
- Setting Recovery Catalog Compatibility Level No Longer Required
- Recovery Catalog Compatibility Changes
- RMAN Syntax Diagram Changes
- Recovery Catalog View Changes
- Backup and Recovery Dynamic Performance View Changes

Note: In release 8.1.7, the formatting of RMAN commands is changed. In previous releases, RMAN commands appeared in lowercase bold, for example, **backup**. Now, RMAN commands appear in uppercase monospace, for example, BACKUP.

Preparing a Standby Database Using RMAN

The procedure for preparing a standby database using RMAN is basically the same as for preparing a duplicate database. The duplication procedure is documented in "Duplicating a Database with RMAN" from *Oracle8i Recovery Manager User's Guide and Reference*. Nevertheless, you need to amend the duplication procedure to account for the issues specific to a standby database.

The documentation for the preparation and maintenance of standby databases is located in *Oracle8i Standby Database Concepts and Administration*. Familiarize yourself with what a standby database is and how to create one *before* you attempt the RMAN creation procedures in this chapter.

This section contains these topics:

- About Standby Database Preparation Using RMAN
- Creating the Standby Control File Using RMAN
- Making Image Copies of Standby Control Files
- Creating the Standby Control File by Cataloging a SQL-Generated Control File
- Naming the Standby Database Datafiles When Using RMAN
- Naming the Standby Database Online Redo Logs When Using RMAN

About Standby Database Preparation Using RMAN

You can either use manual methods or the Recovery Manager `DUPLICATE` command to create a standby database. Before you perform the creation procedure, you must prepare the standby instance. You can use RMAN to do the preparation tasks described in Table 7–2.

Table 7–1 Standby Database Preparation Using RMAN

Task	Procedure
Make a backup of the database to use for creation of standby database.	Use normal backup procedure as documented in *Oracle8i Recovery Manager User's Guide and Reference.*
Create a standby control file (if you do not have one).	"Creating the Standby Control File Using RMAN" on page 7-3
Choose filenames for the standby datafiles.	"Naming the Standby Database Datafiles When Using RMAN" on page 7-6

Table 7–1 Standby Database Preparation Using RMAN

Task	Procedure
Choose filenames for the standby database online redo logs.	"Naming the Standby Database Online Redo Logs When Using RMAN" on page 7-8

You cannot use RMAN to perform all necessary standby database preparation. You must manually perform these tasks:

- Set all necessary initialization parameters in the primary initialization parameter file.
- Create an initialization parameter file for the standby database and set all necessary parameters.
- Perform any Net8 setup and configuration.
- Start the standby instance without mounting the control file.

> **See Also :** *Oracle8i Standby Database Concepts and Administration* for a complete discussion of standby database preparation, including initialization parameter settings. You must perform all necessary preparation tasks described in this document before RMAN can successfully create the standby database files and mount the standby database.

Creating the Standby Control File Using RMAN

In pre-8.1.7 releases, you were required to create the standby control file using the SQL ALTER DATABASE statement. Now, RMAN allows you to create the standby control file using an RMAN command.

Using RMAN, you can create a standby control file in any of the ways described in the following sections:

- Creating the Standby Control File Using the BACKUP Command
- Creating the Standby Control File Using the COPY Command
- Creating the Standby Control File by Cataloging a SQL-Generated Control File

Creating the Standby Control File Using the BACKUP Command

You have these options for creating the standby control file using the BACKUP command:

- You can create a standby control file by running the BACKUP CURRENT CONTROLFILE command with the FOR STANDBY option.
- You can create a standby control file by using the BACKUP . . . INCLUDE CURRENT CONTROLFILE command with the FOR STANDBY option.

To create a backup set containing only a standby control file:

1. Start RMAN and connect to the target database and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET / CATALOG rman/rman@rcat
   ```

   To write the output to a log file, specify the file at startup. For example, enter:

   ```
   % rman TARGET / CATALOG rman/rman@rcat LOG "/oracle/log/mlog.f"
   ```

2. Mount or open the target database. For example, enter:

   ```
   RMAN> STARTUP MOUNT
   ```

3. Create the standby control file using the BACKUP . . . FOR STANDBY command. This example backs up the standby control file to tape:

   ```
   RUN {
        ALLOCATE CHANNEL ch1 TYPE 'sbt_tape';
        BACKUP CURRENT CONTROLFILE FOR STANDBY;
   }
   ```

   You cannot specify a tag for a standby control file.

4. If desired, issue a LIST command to see a listing of backup sets and pieces.

To include the standby control file within another backup:

1. Start RMAN and connect to the target database and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET / CATALOG rman/rman@rcat
   ```

   To write the output to a log file, specify the file at startup. For example, enter:

   ```
   % rman TARGET / CATALOG rman/rman@rcat LOG /oracle/log/mlog.f
   ```

2. Back up the standby database and include a standby control file. This example backs up all the datafiles as well as the standby control file to disk:

```
RUN {
     ALLOCATE CHANNEL ch1 TYPE DISK;
     BACKUP DATABASE
      INCLUDE CURRENT CONTROLFILE FOR STANDBY;
}
```

3. Issue a `LIST` command to see a listing of backup sets and pieces.

Creating the Standby Control File Using the COPY Command

You can create a standby control file by using the `COPY CURRENT CONTROLFILE` command with the `FOR STANDBY` option.

To create a control file copy that is usable as a standby control file:

1. Start RMAN and connect to the target database and, if desired, the recovery catalog database. For example, enter:

```
% rman TARGET / CATALOG rman/rman@rcat
```

2. Use the `FOR STANDBY` option of the `COPY CURRENT CONTROLFILE` command to make a copy of the current control file that is usable as a standby control file. For example, enter:

```
RUN {
     ALLOCATE CHANNEL ch1 TYPE DISK;
     # create standby control file by copying current control file of
     # target database
     COPY CURRENT CONTROLFILE FOR STANDBY TO '/oracle/backup/standbycf.f';
}
```

3. Issue a `LIST COPY` command to see a listing of image copies.

Creating the Standby Control File by Cataloging a SQL-Generated Control File

You can record a standby control file generated using the `ALTER DATABASE` statement in the repository.

To catalog a standby control file generated using ALTER DATABASE:

1. Create a standby control file using the SQL `ALTER DATABASE` statement. This example creates a standby control file in SQL*Plus:

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/oracle/dbs/stbycf.f';
```

2. Use RMAN to connect to the target database and, if desired, the recovery catalog database. For example, enter:

```
% rman TARGET / CATALOG rman/rman@rcat
```

3. Issue the CATALOG command to add metadata about the standby control file to the recovery catalog. For example, enter:

```
RMAN> CATALOG CONTROLFILECOPY '/oracle/dbs/stbycf.f';
```

RMAN considers a control file generated using ALTER DATABASE as a control file copy.

Making Image Copies of Standby Control Files

You can use RMAN to make an image copy of a control file copy that was generated using:

- The RMAN COPY command
- The SQL ALTER DATABASE statement

To copy an RMAN-generated control file copy or SQL-generated control file:

1. Start RMAN and connect to the target database and, if desired, the recovery catalog database. For example, enter:

```
% rman TARGET / CATALOG rman/rman@rcat
```

2. Copy the standby control file using the COPY CONTROLFILECOPY command. RMAN treats SQL-generated control files and RMAN-generated standby control file copies exactly the same. It inspects the header of the control file and determines whether it is a standby or a normal control file.

   For example, run the following command:

```
RUN {
     ALLOCATE CHANNEL ch1 TYPE DISK;
     # copy standby control file created using ALTER DATABASE
     COPY CONTROLFILECOPY '/oracle/dbs/sql_generated_cf.f' TO
                          '/backup/standbycf_sql.f';
     # copy standby control file created using RMAN
     COPY CONTROLFILECOPY '/oracle/dbs/rman _generated_cf.f' TO
                          '/backup/standbycf_rman.f';
}
```

3. Issue a LIST COPY command to see a listing of image copies.

Naming the Standby Database Datafiles When Using RMAN

A standby database can reside either on the same host as the primary database or on a different host. The following table illustrates the implications for renaming the

standby database datafiles depending on whether the directory structures on the hosts are the same or different.

Standby Database Host	Directory Structure	Renaming
Same host as primary	Different from primary host	Necessary
Same host as primary	Same as primary host	Illegal. The standby database cannot exist in the same directories as the primary database on the same host.
Different host from primary	Same as primary host	Not necessary
Different host from primary	Different from primary host	Necessary

When the directory structures are *different* for the primary and standby hosts, you have these options for naming the standby datafiles:

- Configuring the standby database initialization parameter `DB_FILE_NAME_CONVERT`
- Using the `SET AUXNAME` or `SET NEWNAME` commands in RMAN when creating the standby database

When the directory structures are the *same* for the primary and standby hosts, then you have these naming options:

- Leaving the standby filenames the same as the primary filenames (that is, not setting `DB_FILE_NAME_CONVERT` or issuing a `SET AUXNAME` or `SET NEWNAME` command) and specifying the `NOFILENAMECHECK` option of the `DUPLICATE` command
- Using `DB_FILE_NAME_CONVERT`, `SET AUXNAME`, or `SET NEWNAME` to rename the standby datafiles

Because you can specify datafile filenames in the standby control file in multiple ways, a method for prioritizing settings is necessary. Table 7–2 specifies the hierarchy for the naming of datafiles in the standby database.

Table 7–2 Order of Precedence for Naming Datafiles in Standby Database

	Method of Standby Datafile Naming	Requirement
1	Issue `SET AUXNAME` command.	You must be connected to a recovery catalog, and a not-`NULL` `AUXNAME` must be stored in the catalog for the datafile.

Table 7–2 Order of Precedence for Naming Datafiles in Standby Database

	Method of Standby Datafile Naming	Requirement
2	Issue SET NEWNAME command.	You must issue this command in the RUN block for the creation of the standby database.
3	Datafile filename as currently specified in the standby control file. The standby filename is identical to the primary filename or is renamed using DB_FILE_NAME_CONVERT.	If the filename is different, then the DB_FILE_NAME_CONVERT parameter must be set in the standby initialization parameter file. If the filename is the same, then you must specify the NOFILENAMECHECK clause of the DUPLICATE command.

Naming the Standby Database Online Redo Logs When Using RMAN

Until a standby database is opened read-only or activated, it does not have online redo logs. Because a standby database is updated through archived logs received from the primary database, the online logs are not needed. The online redo logs are created at standby activation, *not* standby creation. When you perform failover to a standby database, thereby making the standby database a primary database, Oracle accesses the standby control file to obtain the names for the online redo logs, and then creates them.

The only option when naming the online redo logs on the standby database is the filename for the logs as specified in the standby control file. If the standby online log filenames must be different from the primary online log filenames, then specify filenames for the online redo logs by setting LOG_FILE_NAME_CONVERT in the standby initialization parameter file. If you do not set this parameter, then the filenames in the standby control file for the online logs are the same as the log filenames in the primary database.

Note these restrictions when specifying filenames for the standby online redo logs:

- You must use LOG_FILE_NAME_CONVERT to name the online redo logs if the primary and standby databases use different naming conventions for the logs.
- You cannot use SET NEWNAME or SET AUXNAME to rename the online redo logs.
- You cannot use the LOGFILE clause of the DUPLICATE command to specify filenames for the online redo logs.
- If you want the standby online log filenames the same as the primary online log filenames, then you must specify the NOFILENAMECHECK clause of the DUPLICATE command. Otherwise, RMAN signals an error even if the standby database is created in a different host.

Creating a Standby Database Using RMAN

When you create a standby database, the procedure differs depending on whether the standby database is on the same host as the primary database or on a different host. The procedures in this section assume that you have already completed the standby setup and preparation as outlined in *Oracle8i Standby Database Concepts and Administration*. Do not attempt these procedures until you have made all necessary initialization parameter settings and network configuration.

This section contains these topics:

- About Standby Creation Using RMAN
- Starting RMAN and the Standby Instance
- Restrictions When Creating a Standby Database Using RMAN
- Creating a Standby Database on a Remote Host with the Same Directory Structure
- Creating a Standby Database on a Remote Host with a Different Directory Structure
- Creating a Standby Database on the Local Host
- Using Image Copies to Create a Standby Database

Note: These procedures assume that you are using RMAN backups to create the standby database. If you are using RMAN image copies, then see "Using Image Copies to Create a Standby Database" on page 7-20.

About Standby Creation Using RMAN

After you have performed the steps necessary for preparing the standby instance, use the Recovery Manager `DUPLICATE` command to create the standby database. The steps differ depending on whether you specify that RMAN should recover the standby database after creating it.

RMAN Standby Creation Without Recovery

If you do not specify the `DORECOVER` option of the `DUPLICATE` command, then RMAN automates these steps of the standby creation procedure during duplication:

1. RMAN establishes connections both to the primary and standby databases, and the recovery catalog (if used).
2. RMAN queries the repository, which is either the primary control file or the recovery catalog, to identify the backups of primary database datafiles and the standby control file.
3. If you use a media manager, then RMAN contacts the media manager on the standby host to request the backup data.
4. RMAN restores the standby control file to the standby host, thereby creating the standby control file.
5. RMAN restores the primary datafile backups and copies to the standby host, creating the standby database files.
6. RMAN leaves the standby database mounted, but does *not* place the standby database in manual or managed recovery mode. RMAN disconnects and does not perform media recovery of the standby database.

RMAN Standby Creation with Recovery

If you do specify the `DORECOVER` option of the `DUPLICATE` command, then RMAN automates these steps of the standby creation procedure during duplication:

1. RMAN establishes connections both to the primary and standby databases, and the recovery catalog (if used).
2. RMAN queries the repository, which is either the primary control file or the recovery catalog, to identify the backups of primary database datafiles and the standby control file.
3. If you use a media manager, then RMAN contacts the media manager on the standby host to request the backup data.
4. RMAN restores the standby control file to the standby host, thereby creating the standby control file.
5. RMAN restores the primary datafile backups and copies to the standby host, creating the standby database datafiles.

6. After all data is restored, then RMAN initiates media recovery. If recovery requires archived redo logs, then RMAN attempts to locate them on the standby host disk drives. If the logs are not there, then RMAN attempts to restore them.
7. RMAN recovers the standby database to the specified time, system change number (SCN), or log sequence number, or to the latest archived redo log generated if none of the above are specified.
8. RMAN leaves the standby database mounted after media recovery is complete, but does *not* place the standby database in manual or managed recovery mode.

> **Note:** After RMAN creates the standby database, you must resolve any gap sequence before placing it in manual or managed recovery mode, or opening it in read-only mode. *Oracle8i Standby Database Concepts and Administration* discusses gap sequence resolution in detail.

If you want RMAN to recover the standby database after creating it, then the standby control file must be usable for the desired recovery. Thus, these conditions must be met:

- The end recovery time of the standby database must be greater than or equal to the checkpoint SCN of the standby control file.
- An archived redo log containing the checkpoint SCN of the standby control file must be available at the standby site for recovery.

One way to ensure that these conditions are met is to issue the `ALTER SYSTEM ARCHIVE LOG CURRENT` statement after creating the standby control file. This statement archives the online logs of the primary database. Then, either back up the most recent archived log using RMAN or move the archived log to the standby site.

Whether or not you perform recovery, RMAN does *not* activate the standby database after creating it. The only way to activate a standby database is to issue the `ALTER DATABASE ACTIVATE STANDBY DATABASE` statement. After a standby database is activated and the redo logs are reset, all backups and archived logs of the old primary database are invalid for the new incarnation of the database.

> **See Also :** *Oracle8i Standby Database Concepts and Administration* to learn how to manage a standby database.

Restrictions When Creating a Standby Database Using RMAN

For the complete list of DUPLICATE restrictions for creating a standby database using RMAN, see "DUPLICATE" on page 7-47.

Starting RMAN and the Standby Instance

No matter which standby creation scenario you choose, you must first start the standby instance and then connect RMAN to this instance. The details of this procedure vary depending on whether the standby and primary sites have a different directory structure.

To start the standby instance:

1. Use an operating system utility to copy the initialization parameter file from the target host to the standby host. Set all required parameters as described in *Oracle8i Standby Database Concepts and Administration*. For example, if you are creating the standby on a separate host with a different directory structure, make sure to edit:
   - All initialization parameters that end with _DEST and _PATH and specify a pathname
   - DB_FILE_NAME_CONVERT so that it captures *all* the target datafiles and converts them appropriately, for example, from tbs_* to sbytbs_*
   - LOG_FILE_NAME_CONVERT so that it captures *all* the online redo logs and converts them appropriately, for example, log_* to sbylog_*
2. Use SQL*Plus to start the standby instance without mounting it. For example, enter the following to connect to sbdb1 as SYS (who has SYSDBA privileges) and start the database:

   ```
   SQL> CONNECT SYS/sys_pwd@sbdb1 AS SYSDBA
   SQL> STARTUP NOMOUNT PFILE=initSBDB1.ora
   ```

3. Use SQL*Plus to mount or open the target database if it is not already mounted or open. For example, enter the following to connect to prod1 as SYS and start the database:

   ```
   SQL> CONNECT SYS/sys_pwd@prod1 AS SYSDBA
   SQL> STARTUP PFILE=initPROD1.ora
   ```

   If you use a recovery catalog, make sure that the catalog database is open. For example, enter the following to connect to rcat as SYS and start the database:

   ```
   SQL> CONNECT SYS/sys_pwd@rcat AS SYSDBA
   SQL> STARTUP PFILE=initRCAT.ora
   ```

4. The auxiliary instance must be accessible via Net8. Before proceeding, use SQL*Plus to ensure that you can establish a connection to the auxiliary instance. Note that you must connect to the auxiliary instance with SYSDBA privileges, so a password file must exist.

5. Use RMAN to connect to the target database, the standby instance, and (if you use one) the recovery catalog database. In this example, connection is established without a recovery catalog using operating system authentication:

   ```
   % rman TARGET / AUXILIARY SYS/sys_pwd@sbdb1 NOCATALOG
   ```

   In this example, the user SYS is on both the target and standby databases and has SYSDBA privileges. A net service name is used for the target as follows:

   ```
   % rman AUXILIARY sys/sys_pwd@sbdb1 TARGET SYS/sys_pwd@prod1 NOCATALOG
   ```

   In this example, RMAN connects to the primary, catalog, and auxiliary databases, all using net service names:

   ```
   % rman CATALOG rman/rman@rcat TARGET SYS/sys_pwd@prod1 AUXILIARY SYS/sys_pwd@sbdb1
   ```

Creating a Standby Database on a Remote Host with the Same Directory Structure

The simplest case is to create the standby database on a different host and to use the same directory structure. In this case, you do not need to set the DB_FILE_NAME_CONVERT or LOG_FILE_NAME_CONVERT parameters in the standby initialization parameter file or set new filenames for the standby datafiles. The primary and standby datafiles and logs have the same filenames.

To create a standby database on a different host with the same directory structure:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 7-12. Make sure to set all necessary parameters in the standby initialization parameter file.

2. Perform this step only if you are *not* using RMAN to recover the standby database (the default option). If you *are* using RMAN to recover the standby database, then skip to the next step.

   Follow these steps during the duplication to create but not recover the standby datafiles:

   - Allocate at least one auxiliary channel. This channel performs the work of duplication.
   - Specify the NOFILENAMECHECK option in the DUPLICATE command. The NOFILENAMECHECK option is required when the standby and primary

datafiles and logs have identical filenames. Otherwise, RMAN signals an error.

For example, enter the following:

```
RUN {
     ALLOCATE AUXILIARY CHANNEL ch1 TYPE 'sbt_tape';
     DUPLICATE TARGET DATABASE FOR STANDBY
       NOFILENAMECHECK;
}
```

3. Perform this step only if you *are* using RMAN to recover the standby database after creating it. Otherwise, skip this step.

   Follow these steps during duplication to restore and recover the standby datafiles:

   - Ensure that the end recovery time is greater than or equal to the checkpoint SCN of the standby control file and that a log containing the checkpoint SCN is available for recovery.
   - If desired, issue a `SET` command to specify the end time, SCN, or log sequence number for recovery.
   - Allocate at least one auxiliary channel.
   - Specify the `NOFILENAMECHECK` parameter in the `DUPLICATE` command, and use the `DORECOVER` option.

   For example, enter the following at the RMAN prompt:

```
# If desired, issue a LIST command to determine the SCN of the standby control file.
# The SCN to which you recover must be greater than or equal to the standby control
# file SCN.
LIST BACKUP OF CONTROLFILE;
LIST COPY OF CONTROLFILE;

RUN {
     # If desired, issue a SET command to terminate recovery at a specified point.
     # SET UNTIL SCN 143508;

     ALLOCATE AUXILIARY CHANNEL ch1 TYPE 'sbt_tape';
     DUPLICATE TARGET DATABASE FOR STANDBY
       NOFILENAMECHECK
      DORECOVER;
}
```

   RMAN uses all incremental backups, archived log backups, and archived logs to perform incomplete recovery. The standby database is left mounted.

Creating a Standby Database on a Remote Host with a Different Directory Structure

If you create the standby database on a host with a different directory structure, you need to specify new filenames for the standby database datafiles and online redo logs. You can do the following:

- Set `LOG_FILE_NAME_CONVERT` in the standby initialization parameter file to name the standby database online redo logs. If you do not set `LOG_FILE_NAME_CONVERT`, then you must use the `NOFILENAMECHECK` option of the `DUPLICATE` command.
- Set `DB_FILE_NAME_CONVERT` in the standby initialization parameter file to name the standby datafiles.
- Issue the `SET NEWNAME` command or the `SET AUXNAME` command when using the RMAN `DUPLICATE` command to name the datafiles.

When creating the standby database on a host with a different directory structure, follow one of the procedures in the following sections:

- Creating a Standby Database Using DB_FILE_NAME_CONVERT
- Creating a Standby Database Using SET NEWNAME Commands
- Creating a Standby Database Using SET AUXNAME Commands

See Also:

- "Creating a Duplicate Database with Recovery Manager" and "Recovery Manager Command Syntax" in *Oracle8i Recovery Manager User's Guide and Reference* to learn about the difference between `SET NEWNAME` and `SET AUXNAME`
- *Oracle8i Standby Database Concepts and Administration* for a full discussion of standby database preparation and creation

Creating a Standby Database Using DB_FILE_NAME_CONVERT

In this procedure, you use `DB_FILE_NAME_CONVERT` to name the standby datafiles and `LOG_FILE_NAME_CONVERT` to name the standby online redo logs.

To create a standby database using parameters to name standby files:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 7-12. Make sure to set all necessary parameters in the standby initialization parameter file.

2. Perform this step only if you are *not* using RMAN to recover the standby database (the default option). If you *are* using RMAN to recover the standby database, then skip to the next step.

   After allocating one or more auxiliary channels, issue the `DUPLICATE` command. For example, enter the following:

```
RUN {
     ALLOCATE AUXILIARY CHANNEL ch1 TYPE 'sbt_tape';
     DUPLICATE TARGET DATABASE FOR STANDBY;
}
```

   After restoring the backups, RMAN leaves the standby database mounted.

3. Perform this step only if you *are* using RMAN to recover the standby database after creating it. Otherwise, skip this step.

   Follow these steps:

   - Ensure that the end recovery time is greater than or equal to the checkpoint SCN of the standby control file and that a log containing the checkpoint SCN is available for recovery (as described in "RMAN Standby Creation with Recovery" on page 7-10).
   - If desired, issue a `SET` command to specify the end time, SCN, or log sequence number for recovery.
   - Allocate at least one auxiliary channel.
   - Issue the `DUPLICATE` command with the `DORECOVER` option.

   For example, enter the following at the RMAN prompt:

```
# If desired, issue a LIST command to determine the SCN of the standby control file.
# The SCN to which you recover must be greater than or equal to the control file SCN.
LIST BACKUP OF CONTROLFILE;
LIST COPY OF CONTROLFILE;

RUN {
     # If desired, issue a SET command to terminate recovery at a specified point.
     # SET UNTIL LOGSEQ 612;

     ALLOCATE AUXILIARY CHANNEL ch1 TYPE 'sbt_tape';
     DUPLICATE TARGET DATABASE FOR STANDBY
      DORECOVER;
}
```

   RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. The standby database is left mounted.

Creating a Standby Database Using SET NEWNAME Commands

In this procedure, you use SET NEWNAME commands to name the standby datafiles.

To create a standby database using the SET NEWNAME command:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 7-12. Make sure to set all necessary parameters in the standby initialization parameter file.

2. Perform this step only if you are *not* using RMAN to recover the standby database (the default option). If you *are* using RMAN to recover the standby database, then skip to the next step.

   Perform the following operations:

   - Allocate at least one auxiliary channel.
   - Specify new filenames for the standby database datafiles using SET NEWNAME commands.
   - Issue the DUPLICATE command.

   For example, enter the following:

```
RUN {
     # allocate at least one auxiliary channel of type DISK or 'sbt_tape'
     ALLOCATE AUXILIARY CHANNEL standby1 TYPE 'sbt_tape';
     . . .
     # set new filenames for the datafiles
     SET NEWNAME FOR DATAFILE 1 TO '$ORACLE_HOME/dbs/standby_data_01.f';
     SET NEWNAME FOR DATAFILE 2 TO '$ORACLE_HOME/dbs/standby_data_02.f';
     . . .
     # issue the DUPLICATE command
     DUPLICATE TARGET DATABASE FOR STANDBY;
}
```

   RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery.

3. Perform this step only if you *are* using RMAN to recover the standby database after creating it. Otherwise, skip this step.

   Follow these steps:

   - Ensure that the end recovery time is greater than or equal to the checkpoint SCN of the standby control file and that a log containing the checkpoint SCN is available for recovery (as described in "RMAN Standby Creation with Recovery" on page 7-10).
   - If desired, issue a SET command to specify the end time, SCN, or log sequence number for recovery.

- Allocate at least one auxiliary channel.
- Specify new filenames for the standby database datafiles.
- Issue the DUPLICATE command with the DORECOVER option.

For example, enter the following at the RMAN prompt:

```
# If desired, issue a LIST command to determine the SCN of the standby control file.
# The SCN to which you recover must be greater than or equal to the control file SCN.

LIST BACKUP OF CONTROLFILE;
LIST COPY OF CONTROLFILE;
RUN {
     # If desired, issue a SET command to terminate recovery at a specified point.
     # SET UNTIL TIME 'SYSDATE-7';

     ALLOCATE AUXILIARY CHANNEL ch1 TYPE 'sbt_tape';

     # Set new filenames for the datafiles
     SET NEWNAME FOR DATAFILE 1 TO '$ORACLE_HOME/dbs/standby_data_01.f';
     SET NEWNAME FOR DATAFILE 2 TO '$ORACLE_HOME/dbs/standby_data_02.f';
     . . .
     DUPLICATE TARGET DATABASE FOR STANDBY
      DORECOVER;
}
```

RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. The standby database is left mounted.

Creating a Standby Database Using SET AUXNAME Commands

In this procedure, you use SET AUXNAME commands to name the standby datafiles. See the chapter "Creating a Duplicate Database with Recovery Manager" in *Oracle8i Recovery Manager User's Guide and Reference*.

To create a standby database using the SET AUXNAME command:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 7-12. Make sure to set all necessary parameters in the standby initialization parameter file.
2. Set the auxiliary names for the datafiles. For example, enter the following:

```
# set auxiliary names for the datafiles
SET AUXNAME FOR DATAFILE 1 TO '/oracle/auxfiles/aux_1.f';
SET AUXNAME FOR DATAFILE 2 TO '/oracle/auxfiles/aux_2.f';
...
SET AUXNAME FOR DATAFILE n TO '/oracle/auxfiles/aux_n.f';
```

3. Perform this step only if you are *not* using RMAN to recover the standby database (the default option). If you *are* using RMAN to recover the standby database, then skip to the next step.

Allocate at least one auxiliary channel before issuing the DUPLICATE command, as in the following example:

```
RUN {
     # allocate at least one auxiliary channel of type DISK or 'sbt_tape'
     ALLOCATE AUXILIARY CHANNEL standby1 TYPE 'sbt_tape';
     . . .
     # issue the DUPLICATE command
     DUPLICATE TARGET DATABASE FOR STANDBY;
}
```

4. Perform this step only if you *are* using RMAN to recover the standby database after creating it. Otherwise, skip this step.

   Follow these steps:

   - Ensure that the end recovery time is greater than or equal to the checkpoint SCN of the standby control file and that a log containing the checkpoint SCN is available for recovery (as described in "RMAN Standby Creation with Recovery" on page 7-10).
   - If desired, issue a SET command to specify the end time, SCN, or log sequence number for recovery.
   - Allocate at least one auxiliary channel.
   - Issue the DUPLICATE TARGET DATABASE for standby command.

   For example, enter the following at the RMAN prompt:

```
# If desired, issue a LIST command to determine the SCN of the standby control file.
# The SCN to which you recover must be greater than or equal to the control file SCN.
LIST BACKUP OF CONTROLFILE;
LIST COPY OF CONTROLFILE;

RUN {
     # If desired, issue a SET command to terminate recovery at a specified point.
     # SET UNTIL SCN 1000;

     ALLOCATE AUXILIARY CHANNEL ch1 TYPE 'sbt_tape';
     DUPLICATE TARGET DATABASE FOR STANDBY
      DORECOVER;
}
```

   RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. The standby database is left mounted.

5. Unspecify the auxiliary names for the datafiles so that they are not overwritten by mistake. For example, enter the following at the RMAN prompt:

```
# un-specify auxiliary names for the datafiles
SET AUXNAME FOR DATAFILE 1 TO NULL;
SET AUXNAME FOR DATAFILE 2 TO NULL;
```

```
...
SET AUXNAME FOR DATAFILE n TO NULL;
```

Creating a Standby Database on the Local Host

When creating a standby database on the same host as the target database, follow the same procedure as for duplicating to a remote host with a different directory structure ("Creating a Standby Database on a Remote Host with a Different Directory Structure" on page 7-15).

Note that you can create the standby database in the same Oracle home as the target database, but you must convert the filenames using the same methods used for conversion on a separate host. In other words, you must treat a standby database in the same Oracle home as if it were a database on a separate host with a different directory structure. You must *not* use the same names for standby and primary database files when the two databases are on the same machine.

Caution: Do not use the `NOFILENAMECHECK` option when creating the standby database in the same Oracle home as the primary database. If you do, then you may overwrite the target database files or cause the `DUPLICATE` command to fail with an error.

Using Image Copies to Create a Standby Database

The main restriction when using RMAN image copies to create the standby datafiles is that the image copy filenames for datafiles and archived redo logs on the primary and standby hosts must be the same. For example, assume that datafile 1 is named `/oracle/dbs/df1.f` on the primary host. If you use the RMAN `COPY` command to copy this datafile to `/data/df1.f`, then this image copy must exist on the standby host with the same filename of `/data/df1.f`. Otherwise, RMAN cannot locate the metadata for the standby image copy in its repository.

You have two main ways of populating the standby host with the image copies:

- Transferring them manually using `ftp` or some other utility
- Mounting the standby directory structure on the primary host using a network file system (NFS)

When you use the NFS mount method, you can create a directory on the primary host that maps to a directory on the standby host. If you use this method, then the NFS mount point on both machines must have the same directory name. For example, you can map `/data` on the primary host to `/data` on the standby host,

but you cannot map /data on the primary host to /dir on the standby host (unless you use functionality such as symbolic links in UNIX or logical drives on Windows).

The filename of the image copy on the standby host must be the same as the filename of the image copy on the primary host. Nevertheless, you can specify a different pathname for the standby datafile by using SET NEWNAME commands or the DB_FILE_NAME_CONVERT initialization parameter.

For example, although the image copy of datafile 1 is named /data/df1.f on the standby host, you can specify the pathname /oracle/sb/df1.f in the standby control file using initialization parameters or RMAN commands. Note that you do not manually rename the physical image copy. When you issue the DUPLICATE command, RMAN restores the image copy /data/df1.f and creates the standby datafile 1 as /oracle/sb/df1.f based on the information in the initialization parameters or RMAN commands.

Table 7–3 illustrates two scenarios for creating a standby database with one datafile using NFS.

Table 7–3 Using Image Copies to Create a Standby Database: Scenario

NFS Mount Point	Primary Datafile Filename	Image Copy Filename	Standby Datafile Filename	Procedure
/data (same on both hosts)	/oracle/dbs/df1.f	/data/df1.f	/data/df1.f (same pathname as image copy)	"Creating the Standby Database When Image Copies and Standby Datafiles Share the Same Filenames" on page 7-22
/data (same on both hosts)	/oracle/dbs/df1.f	/data/df1.f	/oracle/sb/df1.f (different pathname from image copy)	"Creating the Standby Database When Image Copies and Standby Datafiles Do Not Share the Same Filenames" on page 7-23

Table 7–3 assumes that the standby directory structure is mounted on the primary host, and that the mount point is /data on both hosts. Because the primary host mounts the standby host directory structure, when you create the image copy /data/df1.f on the primary host, you are actually creating the image copy /data/df1.f on the standby host.

In the first scenario, you name the standby datafiles using the same filenames as the image copies. This case is the simplest because you do not need to use RMAN at all

to create the standby database. First, set the DB_FILE_NAME_CONVERT parameter in the standby initialization parameter file to convert the primary datafile filename /oracle/dbs/df1.f to the standby filename /data/df1.f. Then, copy the files to the standby host, and mount the standby database.

In the second scenario, you use different filenames for the standby datafiles and the image copies. To create this standby database, use the DUPLICATE command. The DUPLICATE command restores the image copy of datafile 1 and renames it according to either SET NEWNAME commands or DB_FILE_NAME_CONVERT.

Creating the Standby Database When Image Copies and Standby Datafiles Share the Same Filenames

This procedure assumes that you are using the same filenames for the standby datafiles and the image copies of the primary datafiles.

To create a standby database when the copies and standby datafiles have the same names:

1. Start RMAN and connect to the target database and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET / CATALOG rman/rman@rcat
   ```

2. Mount but do not open the target database and ensure that the database was closed cleanly prior to mounting. For example, enter:

   ```
   RMAN> STARTUP MOUNT PFILE=init.ora;
   ```

3. Make sure that you set DB_FILE_NAME_CONVERT in the standby initialization parameter file so that standby datafile filenames are translated from the primary datafile filenames. For example:

   ```
   DB_FILE_NAME_CONVERT = ('/oracle/dbs', '/dsk2/oracle')
   ```

4. Copy all of the datafiles and the standby control file. For example, enter:

   ```
   RUN {
        ALLOCATE CHANNEL ch1 TYPE DISK;
        COPY
          DATAFILE 1 TO '/dsk2/oracle/df_1.f',
          DATAFILE 2 TO '/dsk2/oracle/df_2.f',
          DATAFILE 3 TO '/dsk2/oracle/df_3.f',
          DATAFILE 4 to '/dsk2/oracle/df_4.f',
          CURRENT CONTROLFILE FOR STANDBY TO '/dsk2/oracle/cf.f';
   }
   ```

5. Start the auxiliary instance and mount the standby control file. For example, enter:

   ```
   SQL> STARTUP NOMOUNT PFILE=/dsk2/oracle/dbs/initSTANDBY1.ora
   ```

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

Creating the Standby Database When Image Copies and Standby Datafiles Do Not Share the Same Filenames

This procedure assumes that you are using different filenames for the standby datafiles and the image copies of the primary datafiles.

To create a standby database when the copies and standby datafiles have different names:

1. Start RMAN and connect to the target database, auxiliary instance, and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET / CATALOG rman/rman@rcat AUXILIARY sys/pwd@sbdb
   ```

2. Mount but do not open the target database and ensure that the database was closed cleanly prior to mounting. For example, enter:

   ```
   RMAN> STARTUP MOUNT PFILE=init.ora
   ```

3. Either set `DB_FILE_NAME_CONVERT` in the standby initialization parameter file so that standby datafile filenames are translated from the primary datafile filenames, or issue `SET NEWNAME` commands. For example, set the `DB_FILE_NAME_CONVERT` parameter as follows:

   ```
   DB_FILE_NAME_CONVERT = ('/oracle/dbs', '/dsk2/oracle')
   ```

4. Perform this step only if you are *not* using RMAN to recover the standby database (the default option). If you *are* using RMAN to recover the standby database, then skip to the next step.

   Perform these steps:

   a. Allocate at least one disk channel for the copies and one auxiliary channel for the duplication.

   b. Copy all of the datafiles and the standby control file.

   c. Issue the `DUPLICATE` command.

   For example, enter:

   ```
   RUN {
        ALLOCATE CHANNEL ch1 TYPE DISK;
        ALLOCATE AUXILIARY CHANNEL ch2 TYPE DISK;
        COPY
          DATAFILE 1 TO '/dsk2/oracle/df_1.f',
          DATAFILE 2 TO '/dsk2/oracle/df_2.f',
          DATAFILE 3 TO '/dsk2/oracle/df_3.f',
          DATAFILE 4 TO '/dsk2/oracle/df_4.f',
          CURRENT CONTROLFILE FOR STANDBY TO '/dsk2/oracle/cf.f';
   ```

```
      # To ensure that the control file checkpoint is archived, archive the current
      # redo log
      SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
    DUPLICATE TARGET DATABASE FOR STANDBY;
}
```

5. Perform this step only if you *are* using RMAN to recover the standby database after creating it. Otherwise, skip this step.

   Follow these steps:

   a. Ensure that the end recovery time is greater than or equal to the checkpoint SCN of the standby control file and that a log containing the checkpoint SCN is available for recovery (as described in "RMAN Standby Creation with Recovery" on page 7-10).

   b. If desired, issue a `SET` command to specify the end time, SCN, or log sequence number for recovery.

   c. Allocate at least one disk channel for the copies, and one auxiliary channel for the duplication.

   d. Copy every datafile and the standby control file.

   e. Archive the current redo logs.

   f. Issue the `DUPLICATE` command with the `DORECOVER` option.

   For example, enter the following:

```
RUN {
    ALLOCATE CHANNEL ch1 TYPE DISK;
    ALLOCATE AUXILIARY CHANNEL ch2 TYPE DISK;
    COPY
      DATAFILE 1 TO '/dsk2/oracle/df_1.f',
      DATAFILE 2 TO '/dsk2/oracle/df_2.f',
      DATAFILE 3 TO '/dsk2/oracle/df_3.f',
      DATAFILE 4 TO '/dsk2/oracle/df_4.f',
      CURRENT CONTROLFILE FOR STANDBY TO '/dsk2/oracle/cf.f';
      SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
    DUPLICATE TARGET DATABASE FOR STANDBY
    DORECOVER;
}
```

   RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. The standby database is left mounted.

Backing Up Files at the Standby Site Using RMAN

This section contains these topics:

- About RMAN Backups of Standby Database Datafiles and Archived Logs
- Restrictions When Making RMAN Standby Database Backups
- Interpreting the RC_ARCHIVED_LOG View
- Determining When to Back Up Standby Database Archived Redo Logs
- Backing Up a Standby Database Using RMAN

About RMAN Backups of Standby Database Datafiles and Archived Logs

RMAN can back up the standby database and its associated archived redo logs. Standby backups of datafiles and archived redo logs are fully interchangeable with primary database backups. In other words, you can issue the `RESTORE` command to restore a backup of a standby datafile to the primary database, and you can restore a backup of a primary datafile to the standby database. The standby control file and primary control file, however, are *not* interchangeable.

Backing up standby files is often better than backing up the production files, for the following reasons:

- Because the standby database is not the production database, a standby backup does not interfere with transactions or batch jobs in the production database.
- If the standby and primary databases are on separate hosts, then standby backup operations do not consume CPU cycles, allocate memory, or consume other resources on the production host.

If you activate a standby database using `ALTER DATABASE ACTIVATE STANDBY DATABASE`, then the standby database becomes the new primary database. Because a `RESETLOGS` must be performed at standby activation, RMAN creates a new incarnation record for the new primary database. Backups of this new incarnation of the primary database are not different from backups of the primary database after a `RESETLOGS` operation.

> **See Also:** "Opening the Database After Media Recovery" in "Chapter 5, Performing Media Recovery" from *Oracle8i Backup and Recovery Guide* to learn about `RESETLOGS` operations

Restrictions When Making RMAN Standby Database Backups

Note these restrictions when making backups of a standby database:

- If you want the standby database backups to be usable for restore jobs at the primary site, then you must be connected to the recovery catalog when backing up the standby database or resynchronize the standby database shortly after the backup.
- You cannot back up the standby control file being used by the standby database.
- You cannot make an image copy or non-RMAN backup of the standby control file and then use it to restore the primary database.
- You cannot use the `DUPLICATE` command on the standby database to create another standby database.

> **See Also:** *Oracle8i Standby Database Concepts and Administration* for more information about backing up standby databases, and *Oracle8i Recovery Manager User's Guide and Reference* to learn how to make RMAN backups

Interpreting the RC_ARCHIVED_LOG View

If you are making archived log backups on the standby site, you must ensure that all necessary archived logs are available on the primary site in the event of a failure. The situation can be confusing because archived logs can be:

- In the archived log destination directories on the primary site
- Backed up on disk at the primary site
- Backed up on disk at the standby site
- Backed up on tape at the primary site
- Backed up on tape at the standby site

The recovery catalog view `RC_ARCHIVED_LOG` indicates when an archived redo log is located at the primary site and when it is located at the standby site. This information is important because you need to know when you must back up a log or copy it to the primary site from the standby site.

For example, assume that you connect to the recovery catalog and issue this query:

```
SQL> SELECT SEQUENCE#, IS_STANDBY FROM RC_ARCHIVED_LOG;

   SEQUENCE# IS_
  ---------- ---
         113 YES
         114 NO
         115 NO
         116 YES
         116 NO
         117 NO
         118 NO
         119 NO
         120 NO
```

The IS_STANDBY column indicates whether the log is located at the standby site (YES) or at the primary site (NO). If the same log sequence number has IS_STANDBY set to both YES and NO, then the log is located at both the standby and primary sites. For example, sequence number 116 has both a YES and NO value for IS_STANDBY, so it is at the primary and standby sites.

To understand how an archived log can be located at both sites, assume that you have a primary database, a recovery catalog database, and a standby database in managed recovery mode. The following sequence of events occurs:

1. The primary database archives log 113 and transfers it automatically to the standby site. The standby database applies archived log 113 during managed recovery.
2. At the primary site, you issue the following RMAN command:

   ```
   RUN {
        ALLOCATE CHANNEL c1 TYPE 'sbt_tape';
        BACKUP ARCHIVELOG ALL
          DELETE INPUT;
   }
   ```

   This command backs up archived log 113 and deletes it from the primary site.
3. You query the recovery catalog as follows:

   ```
   SELECT SEQUENCE#, IS_STANDBY
   FROM RC_ARCHIVED_LOG
   WHERE SEQUENCE#=113;

    SEQUENCE# IS_
   ---------- ---
          113 YES
   ```

This query returns only one row, 113 YES, because the primary site copy of the archived log has been removed. Only the copy of log 113 at the standby site is available.

Consequently, you cannot use log 113 to perform media recovery at the primary site unless RMAN has backed it up. If you try to perform recovery of the primary database without either using the backup of the log just taken or moving the standby copy to the primary site, then RMAN stops media recovery at log 112 because it cannot locate log 113.

4. Later, the primary database generates a new archived log, sequence 120. You query RC_ARCHIVED_LOG and see the following row:

```
 SEQUENCE# IS_
---------- ---
       120 NO
```

5. After the archiver automatically transmits log 120 to the standby site, a new query of RC_ARCHIVED_LOG (after resynchronization of the recovery catalog) shows:

```
 SEQUENCE# IS_
---------- ---
       120 NO

       120 YES
```

This result indicates that archived log 120 is located at both sites.

6. At the primary site, you back up all archived logs to the tape drive attached to the primary host and remove the input archived logs:

```
RUN {
     ALLOCATE CHANNEL c1 TYPE 'sbt_tape';
     BACKUP ARCHIVELOG ALL
       DELETE INPUT;
}
```

7. You query RC_ARCHIVED_LOG again and see only one row for log sequence number 120, indicating that the log is only available at the standby site:

```
 SEQUENCE# IS_
---------- ---
       120 YES
```

8. At the standby site, you back up all archived logs to the tape drive attached to the standby host and delete the logs on disk:

```
RUN {
     ALLOCATE CHANNEL c1 TYPE 'sbt_tape';
     BACKUP ARCHIVELOG ALL
       DELETE INPUT;
}
```

When you query RC_ARCHIVED_LOG, the catalog shows no more rows for sequence number 120.

Determining When to Back Up Standby Database Archived Redo Logs

If you are making all your backups at the standby site, then you must ensure that you have backed up all the archived logs needed for recovery of the primary database. You have two methods for determining whether you need to back up a standby database archived log so that RMAN can use it for recovery.

Using the LIST Command to Determine When to Back Up Standby Logs

Use the LIST command to determine which logs RMAN has backed up.

To determine whether a log backup is needed by using the LIST command:

1. Query the recovery catalog to determine the locations of the archived redo logs. For example, issue:

```
SQL> SELECT SEQUENCE#, IS_STANDBY FROM RC_ARCHIVED_LOG;

 SEQUENCE# IS_
---------- ---
       113 YES
       114  NO
       115  NO
       116  NO
```

This output indicates that log sequence 113 is at the standby site but not at the primary site, and archived logs 114 - 116 are at the primary site but not the standby site.

2. Determine which archived logs are backed up by connecting RMAN to the recovery catalog and issuing a `LIST BACKUP` command. For example, issue the following:

```
RMAN> LIST BACKUP OF ARCHIVELOG ALL;

RMAN-03022: compiling command: list
RMAN-03025: performing implicit partial resync of recovery catalog
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete

List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ----------------------
319     4          394624547  0  394624546  5          11-APR-00

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ------------------------
    320     1   1   AVAILABLE   11-APR-00              /vobs/oracle/dbs/05boavh2_1_1

    List of Archived Logs Included
    Thrd Seq     Low SCN    Next SCN   Low Time        Next Time
    ---- ------- ---------- ---------- --------------- ---------------
    1    116     95153      95156      07-APR-00       07-APR-00
```

This output shows that RMAN has backed up archived log 116, but has not backed up archived log 113. Because log 113 exists only at the standby site, you should either back up this log or copy it to the primary site.

Querying RC_ARCHIVED_LOG to Determine When to Back Up Standby Logs

You can query the recovery catalog to determine which logs RMAN has backed up.

To determine whether a log backup is needed by querying the catalog:

1. Query the `RC_ARCHIVED_LOG` recovery catalog view to determine whether all archived logs necessary for recovery are on disk. For example, issue the following query, where *first_log_needed_for_recovery* is the sequence number of the log that begins recovery and *expected_num_of_logs* is the number of logs that should be applied during complete recovery:

```
SELECT 1 FROM RC_ARCHIVED_LOG
WHERE SEQUENCE# >= first_log_needed_for_recovery
AND IS_STANDBY='NO'
AND STATUS='A'
HAVING COUNT(*) = expected_num_of_logs;
```

If the query returns no rows, then you do not have all logs necessary for complete recovery on disk. If the query does return rows, then you do have the necessary logs for complete recovery on disk.

2. Query the RC_BACKUP_REDOLOG view to determine whether you have backups of the logs necessary for complete recovery. For example, issue the following query, where *first_log_needed_for_recovery* is the sequence number of the log that begins recovery and *expected_num_of_logs* is the number of logs that should be applied during complete recovery:

```
SELECT 1 FROM RC_BACKUP_REDOLOG
WHERE SEQUENCE# >= first_log_needed_for_recovery
AND STATUS='A'
HAVING COUNT(DISTINCT SEQUENCE#) = expected_num_of_logs;
```

If the query returns no rows, then you do not have backups of all logs necessary for complete recovery. If the query does return rows, then you do have backups of all logs necessary for complete recovery.

Backing Up a Standby Database Using RMAN

Use the RMAN BACKUP command to back up the standby database. A backup of the standby database is exactly the same as a backup of the primary database, except that the backup takes place on the standby site. The primary database has no influence on the backup of the standby database. Note that when you connect to the standby database to perform the backup, you connect using the TARGET keyword and not the AUXILIARY keyword.

As the following table illustrates, whether the standby database backup is consistent or inconsistent depends on the state of the standby database when the backup is made. Only a consistent backup can be restored without performing media recovery.

Standby Database State	Consistency of Backup
Shutdown cleanly and then mounted (but not placed in recovery mode)	Consistent
Mounted after crash or abort	Inconsistent
Manual recovery mode	Inconsistent
Managed recovery mode	Inconsistent
Read-only mode	Inconsistent

To make a whole database backup of a standby database:

1. Start RMAN and connect to the standby database using the TARGET keyword (not the AUXILIARY keyword) and the recovery catalog database. You must be connected to the recovery catalog. For example, enter:

   ```
   % rman TARGET SYS/change_on_install@sbdb1 CATALOG rman/rman@rcat
   ```

2. To make a consistent backup of the standby database, make sure that the last shutdown of the standby was clean and that it was not placed in recovery mode after that time, and then mount the control file. For example, enter:

   ```
   % SQLPLUS sys/change_on_install@sbdb1
   SQL> SHUTDOWN IMMEDIATE
   SQL> STARTUP MOUNT PFILE=initSTANDBY.ora
   ```

   You can back up the standby database when it is in any other mode, but the backups will be inconsistent.

3. Allocate one or more channels of type DISK or type *'sbt_tape'*. This example backs up all the datafiles as well as the control file:

   ```
   RUN {
        ALLOCATE CHANNEL ch1 TYPE 'sbt_tape';
        BACKUP DATABASE;
        BACKUP ARCHIVELOG ALL;
   }
   ```

   If desired, use the FORMAT parameter to specify a filename for the backup piece. For example, enter:

   ```
   RUN {
        ALLOCATE CHANNEL ch1 TYPE 'sbt_tape';
        BACKUP DATABASE
          FORMAT '/oracle/backup/standby_%U';  # %U generates a unique filename
   }
   ```

   If desired, use the TAG parameter to specify a tag for the backup. For example, enter:

   ```
   RUN {
        ALLOCATE CHANNEL ch1 TYPE 'sbt_tape';
        BACKUP DATABASE
          TAG = 'weekly_standby_backup';   # gives the standby backup a tag identifier
   }
   ```

4. If desired, issue a LIST command to see a listing of backup sets and pieces.

Performing a Test Backup Using RMAN

You can use the VALIDATE keyword of the BACKUP command to do the following:

- Check datafiles for physical and logical corruption
- Confirm that all database files exist and are in the correct locations

RMAN does not actually produce backup sets, but scans the specified files to determine whether they can be backed up. In this sense, the BACKUP VALIDATE command is similar to the RESTORE VALIDATE command, except for backups rather than restore jobs.

For example, you can validate that all database files and archived redo logs can be backed up by issuing a command as follows:

```
RUN {
     ALLOCATE CHANNEL ch1 TYPE 'sbt_tape';
     BACKUP VALIDATE
        DATABASE
        ARCHIVELOG ALL;
}
```

Note that you cannot use the MAXCORRUPT or PROXY parameters with the VALIDATE option.

See Also: "BACKUP" on page 7-42

Setting the Default Location of the RMAN Snapshot Control File

Use the SET SNAPSHOT CONTROLFILE LOCATION TO DEFAULT command to set the default filename that is used for the snapshot control file. This default value is platform-specific and depends on the Oracle home. For example, the default filename on some UNIX platforms in release 8.1.7 is $ORACLE_HOME/dbs/snapcf_@.f.

If you upgrade to the current release from a previous release, for example, to release 8.1.7 from release 8.1.6, then the snapshot control file location is not set to the default value. RMAN uses the snapshot location that is already stored in the control file. In this case, the snapshot control file location does not change if you change the Oracle home.

If you create a new database in release 8.1.7, then the snapshot control file location does have the default value. In this case, the default snapshot control file location changes if you change the Oracle home.

Crosschecking and Deleting on Multiple RMAN Channels

This section contains these topics:

- About Allocating Multiple RMAN Channels for Maintenance Commands
- How RMAN Crosschecks and Deletes on Multiple Channels
- Crosschecking Simultaneously on Disk and Tape Channels: Example
- Crosschecking on Multiple OPS Nodes: Example
- Deleting Simultaneously on Disk and Tape Channels: Example
- Releasing Multiple Channels with One Command: Example

About Allocating Multiple RMAN Channels for Maintenance Commands

You can allocate multiple maintenance channels before issuing one of these commands:

- `CROSSCHECK`
- `CHANGE ... CROSSCHECK`
- `DELETE EXPIRED`
- `CHANGE ... DELETE`

RMAN checks on all channels that have the same device type as the channel used to create the backup. Because the multi-channel feature does not parallelize maintenance operations, you should only use this feature in these scenarios:

- To allow crosschecking or deletion of all backup pieces or proxy copies, both on disk and tape, with a single command
- To make crosschecking and deleting work correctly in an Oracle Parallel Server (OPS) cluster in which each backup piece or proxy copy exists only on one node

> **Caution:** Using multiple RMAN channels in other circumstances slows down the maintenance operations and may cause them to fail

This feature uses the existing RMAN syntax. No new syntax options are required.

How RMAN Crosschecks and Deletes on Multiple Channels

When you allocate multiple maintenance channels before a CROSSCHECK, DELETE EXPIRED, or CHANGE . . . DELETE command, RMAN performs the crosscheck or delete on all channels that have the appropriate device type. For example, assume that you do not use a media manager, and have created only one backup of a database as follows:

```
RUN {
   ALLOCATE CHANNEL ch1 TYPE DISK CONNECT 'x/x@node2';
   BACKUP DATABASE;
}
```

Assume that you issue the following series of commands at the RMAN prompt:

```
ALLOCATE CHANNEL FOR MAINTENANCE TYPE DISK CONNECT 'x/x@node1';
allOCATE CHANNEL FOR MAINTENANCE TYPE DISK CONNECT 'x/x@node2';
ALLOCATE CHANNEL FOR MAINTENANCE TYPE 'sbt_tape';
CROSSCHECK BACKUP OF DATABASE;
```

RMAN checks the first two channels because they both have the device type of disk and finds the backup on the second channel. RMAN does not perform a crosscheck on the third channel because you did not make backups using a media manager.

This feature adds the following new restrictions when deleting backup pieces or proxy copies:

- The CHANGE . . . DELETE command results in an error and fails if you attempt to delete a backup using CHANGE . . . DELETE when either of these conditions is met:
  - The backup does not actually exist on the media.
  - The backup is marked EXPIRED in the repository.
- The command DELETE EXPIRED BACKUP results in an error and fails if any expired backups really do exist. In some rare cases, the repository can mark a backup as EXPIRED even though the backup exists. For example, a directory containing a backup is corrupted at the time of the crosscheck, but is later repaired.
- When you specify a media handle with CHANGE . . . CROSSCHECK, the media handle must have been created on the same device type specified on the first allocated maintenance channel. Typically, you allocate only a single maintenance channel before issuing CHANGE . . . CROSSCHECK, but you may want to crosscheck several individual backup pieces or backup sets that were created on different device types. You can still supply keys instead of media

handles. This restriction exists because a media handle is not unique among device types.

Crosschecking Simultaneously on Disk and Tape Channels: Example

RMAN can perform crosschecks on more than one media at the same time. For example, you can perform a crosscheck for backups of the database on disk and tape channels as follows:

```
RMAN> ALLOCATE CHANNEL FOR MAINTENANCE TYPE DISK;
RMAN> ALLOCATE CHANNEL FOR MAINTENANCE TYPE 'sbt_tape';
RMAN> CROSSCHECK BACKUP OF DATABASE;
```

Crosschecking on Multiple OPS Nodes: Example

This feature is useful in an OPS configuration in which tape backups exist on various nodes in the cluster. For example, you can perform a crosscheck on two nodes of an OPS cluster as follows:

```
RMAN> ALLOCATE CHANNEL node_1 TYPE DISK CONNECT 'sys/sys_pwd@node_1';
RMAN> ALLOCATE CHANNEL node_2 TYPE DISK CONNECT 'sys/sys_pwd@node_2';
RMAN> CROSSCHECK BACKUP;
```

Deleting Simultaneously on Disk and Tape Channels: Example

You can also perform deletions on all allocated channels as in this example:

```
RMAN> ALLOCATE CHANNEL FOR MAINTENANCE TYPE DISK;
RMAN> ALLOCATE CHANNEL FOR MAINTENANCE TYPE 'sbt_tape';
RMAN> CHANGE BACKUPSET 1,2,3,4,5 DELETE;
```

RMAN searches for the specified backup sets on all channels and deletes any that it finds.

Releasing Multiple Channels with One Command: Example

You can release all allocated maintenance channels currently allocated by using this command:

```
RMAN> RELEASE CHANNEL;
```

Connecting RMAN to a Target Database in an OPS Cluster

RMAN can only connect to one instance in an Oracle Parallel Server (OPS) cluster at a time. For example, assume that `inst1`, `inst2`, and `inst3` are net services names

for the three instances in an OPS cluster. In this case, you can connect to the target database using only one of these net service names. For example, you can connect as follows:

```
% rman TARGET SYS/sys_pwd@inst2 CATALOG rman/cat_pwd@cat_str
```

Each net service name must specify one and only instance. You cannot specify a net service name that is used by all three instances in the cluster.

Note that the fact that RMAN connects to only one instance does not preclude running a backup using all three instances. For example, you can run a backup job as follows:

```
RUN {
       ALLOCATE CHANNEL c1 TYPE DISK CONNECT = 'SYS/sys_pwd@inst1';
       ALLOCATE CHANNEL c2 TYPE DISK CONNECT = 'SYS/sys_pwd@inst2';
       ALLOCATE CHANNEL c3 TYPE DISK CONNECT = 'SYS/sys_pwd@inst3';
       BACKUP DATABASE;
  }
```

Selected RMAN Maintenance Commands No Longer Require a Catalog

The RMAN maintenance commands are as follows:

- `CHANGE`
- `CROSSCHECK`
- `DELETE EXPIRED`

In release 8.1.6, several uses of these commands required a recovery catalog. In release 8.1.7, fewer maintenance commands require the use of a recovery catalog.

The only options of the `CHANGE` command that now require a recovery catalog are the following:

- `CHANGE ... AVAILABLE` (however, the catalog is only required when used on objects *other than* backup sets, backup pieces, and proxy copies)
- `CHANGE ... UNAVAILABLE`

The `CROSSCHECK` and `DELETE EXPIRED` commands no longer require a recovery catalog in release 8.1.7.

Troubleshooting RMAN Character Set Errors: Scenario

In this scenario, you are connected to the target database while it is not open and attempt to perform an RMAN operation. You receive the following error message:

```
PLS-00553: character set name is not recognized
```

Diagnosis of Cause

Typically, this error message means that the character set value in the client environment, that is, the environment in which you are running the RMAN executable, is different from the character set value in the target database environment.

Solution

1. Determine the character set values in the client and server and environments. For example, on a UNIX system you could query the `NLS_LANG` variable on the client and server:

   ```
   % echo $NLS_LANG
   ```

2. Set the character set environment variable in the client to the same value as the variable in the server. For example, you can set the `NLS_LANG` environment variable on a UNIX system as follows:

   ```
   % setenv NLS_LANG WE8ISO8859P1
   ```

Setting Recovery Catalog Compatibility Level No Longer Required

In Oracle8*i* Release 2 (8.1.6), the `CONFIGURE COMPATIBLE` command set the compatibility level in the recovery catalog. This command helped to solve problems resulting from the way in which RMAN updated and deleted catalog records. In Oracle8*i* Release 3 (8.1.7), these problems are solved, so the `CONFIGURE COMPATIBLE` command is deprecated.

In release 8.1.6, if you set `CONFIGURE COMPATIBLE` to 8.1.5 or lower, then RMAN updated the associated record to status `DELETED` when you issued `CHANGE ... DELETE`. RMAN did not remove the record, but only changed its status. In release 8.1.7, RMAN always removes the recovery catalog record when you issue `CHANGE ... DELETE`.

Recovery Catalog Compatibility Changes

This section contains these topics:

- About RMAN Compatibility
- RMAN Compatibility Matrix
- RMAN Compatibility: Scenario

About RMAN Compatibility

The RMAN environment can contain the following components:

- RMAN executable
- Recovery catalog database
- Recovery catalog schema in the recovery catalog database
- Target database
- Auxiliary database (that is, a duplicate or standby database)

Each component has a release number. For example, you can use a release 8.1.5 RMAN executable with:

- A release 8.1.6 target database
- A release 8.1.6 duplicate database
- A release 8.1.5 recovery catalog database whose catalog tables were created using RMAN release 8.1.6

RMAN Compatibility Matrix

In general, the rules of RMAN compatibility are as follows:

- The RMAN catalog schema version should be greater than or equal to the catalog database version. See "Note: 8.0 Catalog Schemas and 8.1 Targets" on page 7-40.
- The RMAN catalog is backwards compatible with target databases from earlier releases. See "Note: 8.1 Catalog Schemas and 8.0 Targets" on page 7-40.
- The versions of the RMAN executable and the target database should be the same. See Table 7–4 for other legal combinations.

Table 7–4 shows version requirements for RMAN components.

Table 7–4 RMAN Compatibility Table

Target/Auxiliary Database	RMAN Executable	Catalog Database	Catalog Schema
8.0.3	8.0.3	8.x	8.0.3
8.0.4	8.0.4	8.x	>= 8.0.4, see "Note: 8.1 Catalog Schemas and 8.0 Targets"
8.0.5	8.0.5	8.x	>= 8.0.5, see "Note: 8.1 Catalog Schemas and 8.0 Targets"
8.0.6	8.0.6	8.x	8.0.6
8.0.6	8.0.6	8.1.x	8.1.x
8.1.5	8.1.5	8.1.x	>= 8.1.5
8.1.6	8.0.6.1	8.x	8.0.6
8.1.6	8.0.6.1	8.1.x	>= RMAN executable
8.1.6	>= 8.1.5	8.1.x	>= RMAN executable
8.1.7	8.0.6.1	8.x	8.0.6
8.1.7	8.0.6.1	8.1.x	8.1.x
8.1.7	>= 8.1.5	8.1.x	>= RMAN executable

Note: 8.0 Catalog Schemas and 8.1 Targets

RMAN cannot create 8.1 catalog schemas in 8.0 catalog databases.

Note: 8.1 Catalog Schemas and 8.0 Targets

Restore operations for an 8.0.4 or 8.0.5 target with an 8.1 catalog schema do not work when both these conditions are met:

- The target database is mounted or open
- You are connected to a recovery catalog

If any of these conditions is not met, then you can use an 8.1 catalog schema with an 8.0.4 or 8.0.5 target database

Note: 8.1.6 Catalog Schema with Pre-8.1.6 RMAN Executable

Using a pre-8.1.6 release of the RMAN executable with recovery catalog schema of release 8.1.6 (newly created by 8.1.6 RMAN executable using the CREATE CATALOG command) requires the following update at the catalog database:

```
SQL> UPDATE CONFIG SET VALUE='080004' WHERE NAME='COMPATIBLE';
```

RMAN Compatibility: Scenario

Assume that you maintain four production databases of the following releases:

- 8.0.5
- 8.0.6
- 8.1.6
- 8.1.7

You want to record metadata about these databases in a single recovery catalog database. According to Table 7–4, you can use a single 8.1.7 recovery catalog database, but you must use two different catalog schemas. You must use an 8.0.5 catalog schema to register the 8.0.5 target database, but you can use an 8.1.7 catalog schema to register the 8.0.6, 8.1.6, and 8.1.7 target databases. The 8.0.5 catalog schema and 8.1.7 catalog schema can co-exist in a single 8.1.7 catalog database.

Note that you cannot use a single RMAN executable to back up all the databases. You must use an 8.0.5 executable to back up the 8.0.5 target database, but you can use an 8.0.6.1 RMAN executable to back up the other databases.

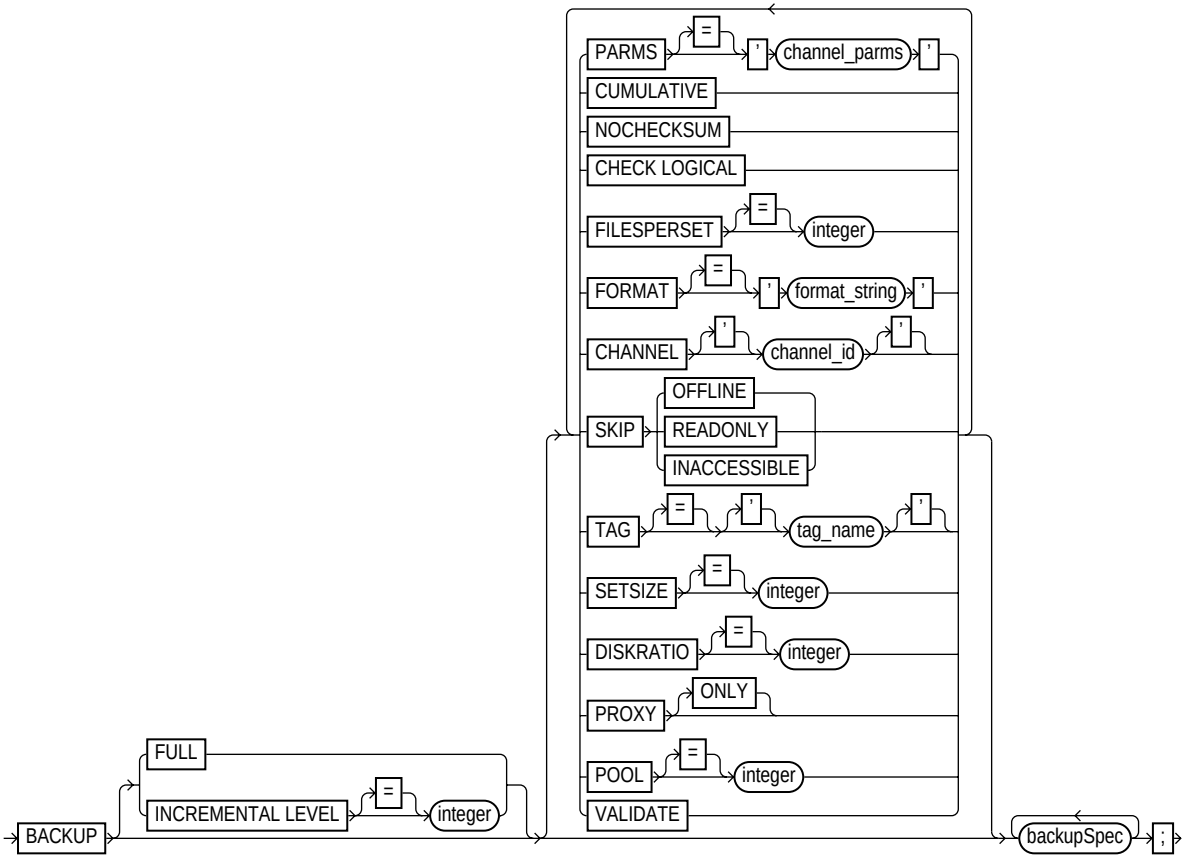
RMAN Syntax Diagram Changes

The following RMAN syntax diagrams have changed for release 8.1.7:

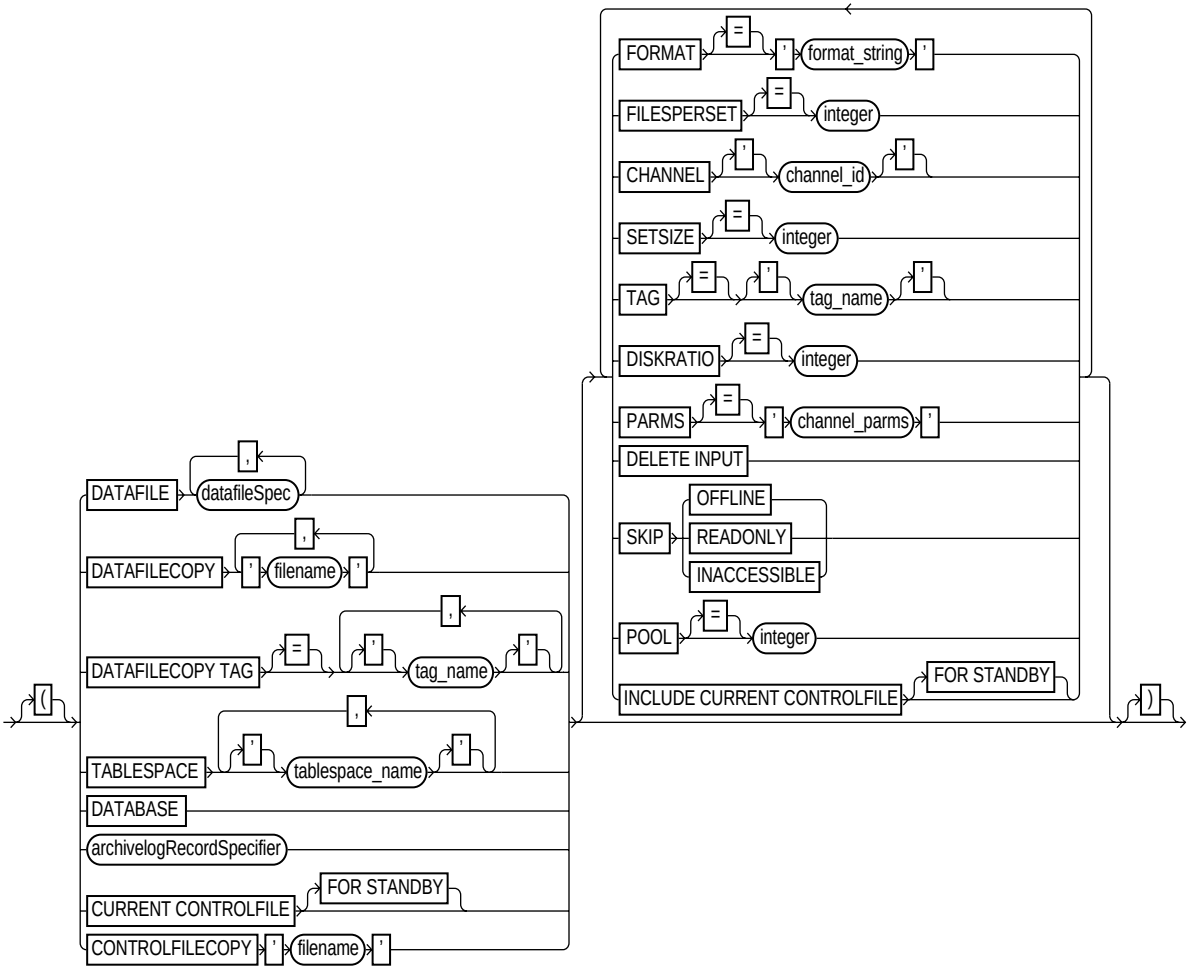
- BACKUP
- COPY
- DEBUG
- DUPLICATE
- SET

BACKUP

Syntax



backupSpec::=



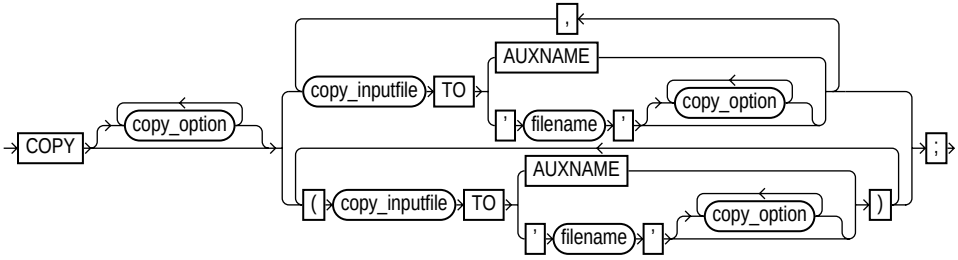
Changed Keywords and Parameters

`VALIDATE`	causes RMAN to scan the specified files and verify their contents. This operation creates no output files. Use this command periodically to check for physical and logical errors in database files.

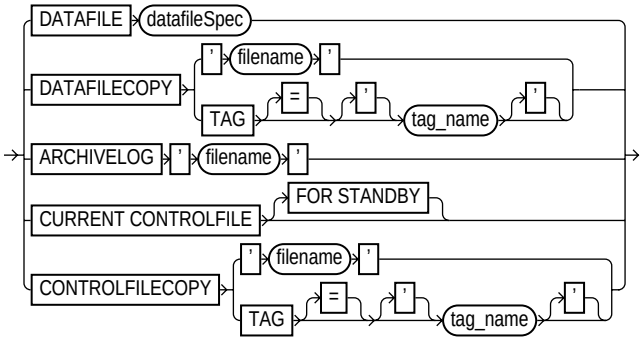
`CURRENT CONTROLFILE`	backs up the current control file.	
	`FOR STANDBY`	makes a backup of the current control file that can be used with a standby database. A standby control file can also be used as an ordinary control file backup, so you can restore it in the target database if necessary.

COPY

Syntax



copy_inputfile::=

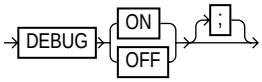


Changed Keywords and Parameters

`CURRENT CONTROLFILE`	backs up the current control file.	
	`FOR STANDBY`	makes an image copy of the current control file that can be used with a standby database. A standby control file can also be used as an ordinary control file backup, so you can restore it in the target database if necessary.

DEBUG

Syntax

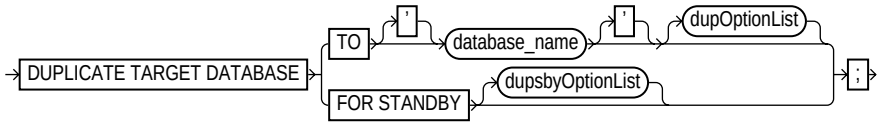


Changed Keywords and Parameters

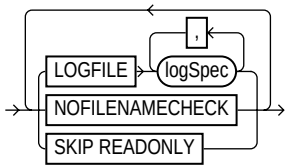
No keywords or parameters have changed, but the semi-colon is now optional.

DUPLICATE

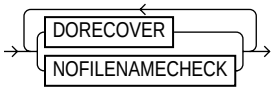
Syntax



dupOptionList::=



dupsbyOptionList::=



Changed Restrictions

Note the following restrictions involved when using the `DUPLICATE` command to create a standby database:

- The standby instance must be started but not mounted.
- RMAN must be connected to the target database and to the auxiliary instance. If desired, you can connect to the recovery catalog.
- At least one auxiliary instance channel must be allocated in the `RUN` block. You can allocate multiple auxiliary channels if needed.
- All backups and copies located on disk must be available at the standby host with the same pathnames as in the target host.

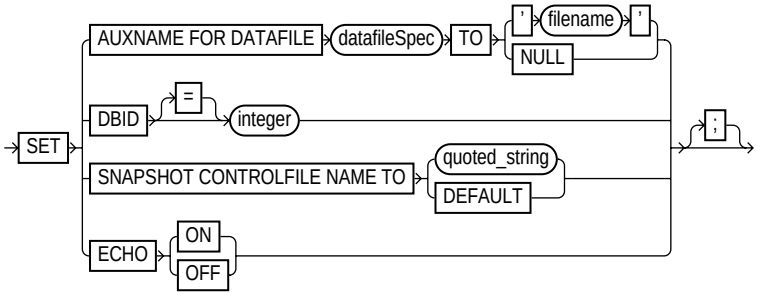
- Backups on tape must be accessible from the standby host.
- If archived logs have not been backed up, then archived logs must be available at the standby host with the same pathnames as in the target host.
- If RMAN recovers the standby database, then the checkpoint SCN of the control file must be included in an archived redo log that is either available at the standby site or included in an RMAN backup. For example, assume that you create the standby control file and then immediately afterwards archive the current log, which has a sequence of 100. You must recover the standby database up to at least log sequence 100, or Oracle signals `ORA-1152` because the standby control file backup or copy was taken after the point in time.
- You cannot specify the `SKIP READONLY` and `LOGFILE` options of the `DUPLICATE` command. These options are legal for a duplicate database but illegal for the creation of the standby database.
- You cannot use `SET NEWNAME` or `SET AUXNAME` to transform the filenames for the online redo logs on the standby database.
- You cannot use the `DUPLICATE` command to activate a standby database.

Changed Keywords and Parameters

`FOR STANDBY`	creates a standby database rather than a duplicate database. Specify this keyword only when creating a standby database. If you do not specify the `DORECOVER` keyword, then RMAN creates the standby database and then leaves it mounted.	
	`DORECOVER`	specifies that RMAN should recover the database after creating it. If you specify an *untilClause*, then RMAN recovers to the specified point and leaves the database mounted.

SET

Syntax



Changed Keywords and Parameters

`SNAPSHOT CONTROLFILENAME TO`	sets the snapshot control file filename to either '*filename*' or the `DEFAULT` value. The default value is platform-specific and dependent on the Oracle home. For example, the default on some UNIX system is `$ORACLE_HOME/dbs/snapcf_@.f`. Note that if you set the control file name using the `DEFAULT` keyword, and you change the Oracle home, then the default location of the snapshot control file changes as well. If you upgrade to the current release from a previous release (for example, release 8.1.6), then the snapshot control file is not automatically set to the default value. RMAN uses the snapshot filename that is already stored in the control file. In this case, the snapshot filename does not change if you change the Oracle home. If you create a new database in release 8.1.7, however, then the snapshot filenames does automatically have the platform-specific default value. The default snapshot control file location changes when you change the Oracle home.

Recovery Catalog View Changes

The recovery catalog views are unchanged except for new columns that are added to the following views.

View	New Column	Datatype	Null	Description
RC_ARCHIVED_LOG	IS_STANDBY	VARCHAR2(1)	NOT NULL	The location of archived log: Y (located on the standby database host) or N (located on the primary database). A standby log cannot be used for recovery of the primary unless it is first backed up by RMAN.
RC_BACKUP_CONTROLFILE	CONTROLFILE_TYPE	VARCHAR2(1)	NOT NULL	The type of control file backup: B (normal backup) or S (standby backup).
RC_BACKUP_SET	CONTROLFILE_ INCLUDED	VARCHAR2(1)	NOT NULL	Possible values are NONE (backup set does not include a control file), BACKUP (backup set includes a normal control file), and STANDBY (backup set includes a standby control file).
RC_CONTROLFILE_COPY	CONTROLFILE_TYPE	VARCHAR2(1)	NOT NULL	The type of control file copy: B (normal) or S (standby).
RC_PROXY_CONTROLFILE	CONTROLFILE_TYPE	VARCHAR2(1)	NOT NULL	The type of control file copy: B (normal) or S (standby).

Backup and Recovery Dynamic Performance View Changes

The following changes to V$ views are relevant for RMAN operations involving the standby database.

View	New Column	Datatype	Null	Description
V$ARCHIVED_LOG	CREATOR	VARCHAR2(4)	NOT NULL	The database process that generated the archived log: ARCH, FGRD (foreground process), LGWR, or RMAN.
V$ARCHIVED_LOG	REGISTAR	VARCHAR2(3)	NOT NULL	The database process that writes the archived log record: ARCH, FGRD (foreground process), LGWR, RFS, RMAN (if registered by RMAN at the primary), and SRMN (if registered by RMAN at a standby database). Only RFS and SRMN write to the standby database: all other processes write to the primary database.
V$ARCHIVED_LOG	STANDBY_DEST	VARCHAR2(3)	NOT NULL	Y if the name of the log is a standby net service name and N if the name of the log is an archived log filename.

8

Reference

This chapter describes new or changed features in the Oracle8i Reference for release 8.1.7. The following parameter and dynamic performance views are new or have been revised for this release:

- ORACLE_TRACE_ENABLE
- V$ARCHIVED_LOG
- V$BACKUP_DATAFILE
- V$BACKUP_SET
- V$DATAFILE_COPY
- V$PROXY_DATAFILE
- V$SQL_SHARED_CURSOR

ORACLE_TRACE_ENABLE

This parameter was static previous to release 8.1.7. It is now dynamic.

Parameter type	Boolean
Syntax	ORACLE_TRACE_ENABLE = {TRUE | FALSE}
Default value	FALSE
Parameter class	Dynamic
	Scope= ALTER SESSION, ALTER SYSTEM

To enable Oracle Trace collections for the server, set ORACLE_TRACE_ENABLE to TRUE. This setting alone does not start an Oracle Trace collection, but it allows Oracle Trace to be used for the server.

With ORACLE_TRACE_ENABLE set to TRUE, you can perform Oracle Trace collection of server event data in any of the following ways:

- By using Oracle Trace Manager, which is supplied with the Oracle Diagnostic Pack
- By using the TRACE statement
- By specifying a collection name in the ORACLE_TRACE_COLLECTION_NAME parameter

See Also:

- *Oracle8i Designing and Tuning for Performance* for more information on the Oracle Trace facility and on setting this parameter

V$ARCHIVED_LOG

This view displays archived log information from the control file, including archive log names. An archive log record is inserted after the online redo log is successfully archived or cleared (name column is NULL if the log was cleared). If the log is archived twice, there will be two archived log records with the same THREAD#, SEQUENCE#, and FIRST_CHANGE#, but with a different name. An archive log record is also inserted when an archive log is restored from a backup set or a copy and whenever a copy of a log is made with the RMAN **copy** command.

Column	Datatype	Description
RECID	NUMBER	Archived log record ID
STAMP	NUMBER	Archived log record stamp
NAME	VARCHAR2(513)	Archived log file name. If set to NULL, the log file was cleared before it was archived.

Column	Datatype	Description
DEST_ID	NUMBER	Identifies the original destination from which the archivelog was generated. A value of 0 indicates the destination identifier is not available.
THREAD#	NUMBER	Redo thread number
SEQUENCE#	NUMBER	Redo log sequence number
RESETLOGS_CHANGE#	NUMBER	Resetlogs change# of the database when this log was written
RESETLOGS_TIME	DATE	Resetlogs time of the database when this log was written
FIRST_CHANGE#	NUMBER	First change# in the archived log
FIRST_TIME	DATE	Timestamp of the first change
NEXT_CHANGE#	NUMBER	First change in the next log
NEXT_TIME	DATE	Timestamp of the next change
BLOCKS	NUMBER	Size of the archived log in blocks
BLOCK_SIZE	NUMBER	Redo log block size
CREATOR	VARCHAR2(4)	Identifies the creator of the archivelog (ARCH, FRDF, or RMAN)
REGISTRAR	VARCHAR2(4)	Identifies the registrar of the entry (RFS, ARCH, FRGD, RMAN, or SRMN, which is RMAN at standby)
STANDBY_DEST	VARCHAR2(3)	(YES/NO) Indicates if the entry is an archivelog destination
ARCHIVED	VARCHAR2(3)	Indicates that the online redo log was archived (YES) or that RMAN only inspected the log and created a record for future application of redo logs during recovery. **See Also:** *Oracle8i Recovery Manager User's Guide and Reference.*
DELETED	VARCHAR2(3)	Specifies (YES | NO) whether an RMAN **delete** command has physically deleted the archived log file from disk, as well as logically removing it from the control file of the target database and from the recovery catalog.
COMPLETION_TIME	DATE	Time when the archiving completed

V$BACKUP_DATAFILE

For release 8.1.7, V$BACKUP_DATAFILE has a new column: CONTROLFILE_TYPE.

This view displays backup datafile and backup control file information from the control file.

Column	Datatype	Description
RECID	NUMBER	Backup datafile record ID
STAMP	NUMBER	Backup datafile record stamp

Column	Datatype	Description
SET_STAMP	NUMBER	Backup set stamp
SET_COUNT	NUMBER	Backup set count
FILE#	NUMBER	Datafile number; set to 0 for control file
CREATION_CHANGE#	NUMBER	Creation change of the datafile
CREATION_TIME	DATE	Creation timestamp of the datafile
RESETLOGS_CHANGE#	NUMBER	Resetlogs change# of the datafile when it was backed up
RESETLOGS_TIME	DATE	Resetlogs timestamp of the datafile when it was backed up
INCREMENTAL_LEVEL	NUMBER	(0-4) incremental backup level
INCREMENTAL_CHANGE#	NUMBER	All blocks changed after incremental change# is included in this backup; set to 0 for a full backup
CHECKPOINT_CHANGE#	NUMBER	All changes up to checkpoint change# are included in this backup
CHECKPOINT_TIME	DATE	Timestamp of the checkpoint
ABSOLUTE_FUZZY_CHANGE#	NUMBER	Highest change# in this backup
MARKED_CORRUPT	NUMBER	Number of blocks marked corrupt
MEDIA_CORRUPT	NUMBER	Number of blocks media corrupt
LOGICALLY_CORRUPT	NUMBER	Number of blocks logically corrupt
DATAFILE_BLOCKS	NUMBER	Size of the datafile in blocks at backup time. This value is also the number of blocks taken by the datafile restarted from this backup.
BLOCKS	NUMBER	Size of the backup datafile in blocks. Unused blocks are not copied to the backup.
BLOCK_SIZE	NUMBER	Block size
OLDEST_OFFLINE_RANGE	NUMBER	The RECID of the oldest offline range record in this backup control file. 0 for datafile backups.
COMPLETION_TIME	DATE	The time completed
CONTROLFILE_TYPE	VARCHAR2(1)	B indicates normal copies S indicates standby copies

V$BACKUP_SET

This view displays backup set information from the control file. A backup set record is inserted after the backup set is successfully completed.

Column	Datatype	Description
RECID	NUMBER	Backup set record ID

Column	Datatype	Description
STAMP	NUMBER	Backup set record timestamp
SET_STAMP	NUMBER	Backup set stamp. The backup set stamp and count uniquely identify the backup set. Primary key for the V$BACKUP_SET table, and the foreign key for the following tables: V$BACKUP_PIECE, V$BACKUP_DATAFILE, V$BACKUP_REDOLOG, V$BACKUP_CORRUPTION
SET_COUNT	NUMBER	Backup set count. The backup set count is incremented by one every time a new backup set is started (if the backup set is never completed the number is "lost"). If the control file is recreated then the count is reset to 1. Therefore the count must be used with the stamp to uniquely identify a backup set. Primary key for the V$BACKUP_SET table, and the foreign key for the following tables: V$BACKUP_PIECE, V$BACKUP_DATAFILE, V$BACKUP_REDOLOG, V$BACKUP_CORRUPTION
BACKUP_TYPE	VARCHAR2(1)	Type of files that are in this backup. If the backup contains archived redo logs, the value is `L'. If this is a datafile full backup, the value is `D'. If this is an incremental backup, the value is `I'.
CONTROLFILE_ INCLUDED	VARCHAR2(3)	Set to YES if there is a control file included in this backup set, otherwise set to NO
INCREMENTAL_LEVEL	NUMBER	Location where this backup set fits into the database's backup strategy. Set to zero for full datafile backups, non-zero for incremental datafile backups, and NULL for archivelog backups.
PIECES	NUMBER	Number of distinct backup pieces in the backup set
START_TIME	DATE	The starting time
COMPLETION_TIME	DATE	The time that this backup set completed
ELAPSED_SECONDS	NUMBER	The number of elapsed seconds
BLOCK_SIZE	NUMBER	Block size of the backup set
INPUT_FILE_SCAN_ ONLY	VARCHAR2(3)	YES indicates no actual backup is performed, but the datafiles are read. NO indicates a normal backup is performed

V$DATAFILE_COPY

For release 8.1.7, V$DATAFILE_COPY has a new column: CONTROLFILE_TYPE.

This view displays datafile copy information from the control file.

Column	Datatype	Description
RECID	NUMBER	Datafile copy record ID
STAMP	NUMBER	Datafile copy record stamp

Column	Datatype	Description
NAME	VARCHAR2(513)	Filename of the datafile copy. The maximum length of the name is OS dependent.
TAG	VARCHAR2(32)	Datafile copy tag
FILE#	NUMBER	Absolute datafile number
RFILE#	NUMBER	Tablespace relative datafile number
CREATION_CHANGE#	NUMBER	Datafile creation change#
CREATION_TIME	DATE	Datafile creation timestamp
RESETLOGS_CHANGE#	NUMBER	Resetlogs change# of the datafile when the copy was made
RESETLOGS_TIME	DATE	Resetlogs timestamp of the datafile when the copy was made
INCREMENTAL_LEVEL	NUMBER	The incremental level
CHECKPOINT_CHANGE#	NUMBER	Checkpoint change# of the datafile when the copy was made
CHECKPOINT_TIME	DATE	Checkpoint timestamp of the datafile when the copy was made
ABSOLUTE_FUZZY_ CHANGE#	NUMBER	Highest change seen when the datafile was copied
RECOVERY_FUZZY_ CHANGE#	NUMBER	Highest change written to the file by media recovery
RECOVERY_FUZZY_TIME	DATE	Timestamp of the highest change written to the file by media recovery
ONLINE_FUZZY	VARCHAR2(3)	(YES | NO) If set to YES, this is a copy taken using an operating system utility after a crash or offline immediate (or an invalid copy taken while datafile was online and the database open). Recovery will need to apply all redo up to the next crash recovery marker to make the file consistent.
BACKUP_FUZZY	VARCHAR2(3)	(YES | NO) If set to YES, this is a copy taken using the BEGIN BACKUP/ END BACKUP technique. Recovery will need to apply all redo up to the end backup marker to make this copy consistent.
MARKED_CORRUPT	NUMBER	Number of blocks marked corrupt by this copy operation. That is, blocks that were not marked corrupted in the source datafile, but were detected and marked as corrupted during the copy operation.
MEDIA_CORRUPT	NUMBER	Total number of media corrupt blocks. For example, blocks with checksum errors are marked media corrupt.
LOGICALLY_CORRUPT	NUMBER	Total number of logically corrupt blocks. For example, applying redo for unrecoverable operations will mark affected blocks logically corrupt.
BLOCKS	NUMBER	Size of the datafile copy in blocks (also the size of the datafile when the copy was made)
BLOCK_SIZE	NUMBER	Block size of the datafile
OLDEST_OFFLINE_RANGE	NUMBER	The RECID of the oldest offline range record in this control file copy; 0 for datafile copies

Column	Datatype	Description
DELETED	VARCHAR2(3)	(YES | NO) If set to YES the datafile copy has been deleted or overwritten
COMPLETION_TIME	DATE	Time when the copy was completed
CONTROLFILE_TYPE	VARCHAR2(1)	B indicates normal copies S indicates standby copies

V$PROXY_DATAFILE

For release 8.1.7, V$PROXY_DATAFILE has a new colum: CONTROLFILE_TYPE.

This view contains descriptions of datafile and control file backups that are taken with Proxy Copy. Each row represents a backup of one database file.

Column	Datatype	Description
RECID	NUMBER	Proxy copy record ID
STAMP	NUMBER	Proxy copy record stamp
DEVICE_TYPE	VARCHAR2(17)	Type of the device on which the copy resides
HANDLE	VARCHAR2(513)	Proxy copy handle identifies the copy for restore
COMMENTS	VARCHAR2(81)	Comment returned by the operating system or storage subsystem. This value is informational only; not needed for restore.
MEDIA	VARCHAR2(65)	Name of the media on which the copy resides. This value is informational only; not needed for restore.
MEDIA_POOL	NUMBER	The media pool in which the copy resides. This is the same value that was entered in the **pool** operand of the Recovery Manager **backup** command
TAG	VARCHAR2(32)	Proxy copy tag
STATUS	VARCHAR2(1)	Indicates the status of the copy: ■ A - The object is available ■ D - The object is deleted ■ X - The object has been "cross-checked" and found not to exist. A subsequent "delete expired" command will change the status to D. If, for some reason, the object really does still exist, then a subsequent "cross-check" command will change the status back to A.
FILE#	NUMBER	Absolute datafile number, or 0 if this is a control file backup
CREATION_CHANGE#	NUMBER	Datafile creation change number
CREATION_TIME	DATE	Datafile creation Timestamp
RESETLOGS_CHANGE#	NUMBER	Resetlogs change number of the datafile when the copy was made
RESETLOGS_TIME	DATE	Resetlogs timestamp of the datafile when the copy was made

Column	Datatype	Description
CHECKPOINT_CHANGE#	NUMBER	Checkpoint change number of the datafile when the copy was made
CHECKPOINT_TIME	DATE	Checkpoint timestamp of the datafile when the copy was made
ABSOLUTE_FUZZY_ CHANGE#	NUMBER	The highest change in any block of the file, if known
RECOVERY_FUZZY_ CHANGE#	NUMBER	Highest change written to the file by media recovery
RECOVERY_FUZZY_TIME	DATE	Timestamp of the highest change written to the file by media recovery
INCREMENTAL_LEVEL	NUMBER	0 if this backup is part of an incremental backup strategy, otherwise NULL
ONLINE_FUZZY	VARCHAR2(3)	(YES|NO) If set to YES, this copy was made after a crash or offline immediate (or is a copy of a copy which was taken improperly while the database was open). Recovery will need to apply all redo up to the next crash recovery marker to make the file consistent.
BACKUP_FUZZY	VARCHAR2(3)	(YES|NO) If set to YES, this is a copy taken using the BEGIN BACKUP|END BACKUP technique. Note that the BEGIN BACKUP|END BACKUP technique is used internally when proxy copies of open files are created. Recovery will need to apply all redo up to the end backup marker to make this copy consistent.
BLOCKS	NUMBER	Size of the copy in blocks (also the size of the datafile when the copy was made)
BLOCK_SIZE	NUMBER	Block size of the datafile
OLDEST_OFFLINE_RANGE	NUMBER	If file# is 0 (ie, this is a control file backup), the RECID of the oldest offline range record in this control file copy. 0 for datafile copies.
START_TIME	DATE	The starting time
COMPLETION_TIME	DATE	The completion time
ELAPSED_SECONDS	NUMBER	The number of elapsed seconds
CONTROLFILE_TYPE	VARCHAR2(1)	B indicates normal copies S indicates standby copies

V$SQL_SHARED_CURSOR

This new dynamic performance view describes explains why a particular child cursor is not shared with existing child cursors. Each column identifies a specific reason why the cursor cannot be shared.

Column	Datatype	Description
ADDRESS	RAW(4)	Address of the child cursor
KGLHDPAR	VARCHAR2(1)	(Y or N) Address of the pattern cursor

Column	Datatype	Description
UNBOUND_CURSOR	VARCHAR2(1)	(Y or N) The existing child cursor was not fully built (in other words, it was not optimized)
SQL_TYPE_MISMATCH	VARCHAR2(1)	(Y or N) The SQL type does not match the existing child cursor
OPTIMIZER_MISMATCH	VARCHAR2(1)	(Y or N) The optimizer environment does not match the existing child cursor
OUTLINE_MISMATCH	VARCHAR2(1)	(Y or N) The outlines do not match the existing child cursor
STATS_ROW_MISMATCH	VARCHAR2(1)	(Y or N) The existing statistics do not match the existing child cursor
LITERAL_MISMATCH	VARCHAR2(1)	(Y or N) Non-data literal values do not match the existing child cursor
SEC_DEPTH_MISMATCH	VARCHAR2(1)	(Y or N) Security level does not match the existing child cursor
EXPLAIN_PLAN_CURSOR	VARCHAR2(1)	(Y or N) The child cursor is an explain plan cursor and should not be shared
BUFFERED_DML_ MISMATCH	VARCHAR2(1)	(Y or N) Buffered DML does not match the existing child cursor
PDML_ENV_MISMATCH	VARCHAR2(1)	(Y or N) PDML environment does not match the existing child cursor
INST_DRTLD_MISMATCH	VARCHAR2(1)	(Y or N) Insert direct load does not match the existing child cursor
SLAVE_QC_MISMATCH	VARCHAR2(1)	(Y or N) The existing child cursor is a slave cursor and the new one was issued by the coordinator (or, the existing child cursor was issued by the coordinator and the new one is a slave cursor)
TYPECHECK_MISMATCH	VARCHAR2(1)	(Y or N) The existing child cursor is not fully optimized
AUTH_CHECK_MISMATCH	VARCHAR2(1)	(Y or N) Authorization/translation check failed for the existing child cursor
BIND_MISMATCH	VARCHAR2(1)	(Y or N) The bind metadata does not match the existing child cursor
DESCRIBE_MISMATCH	VARCHAR2(1)	(Y or N) The typecheck heap is not present during the describe for the child cursor
LANGUAGE_MISMATCH	VARCHAR2(1)	(Y or N) The language handle does not match the existing child cursor
TRANSLATION_ MISMATCH	VARCHAR2(1)	(Y or N) The base objects of the existing child cursor do not match
ROW_LEVEL_SEC_ MISMATCH	VARCHAR2(1)	(Y or N) The row level security policies do not match
INSUFF_PRIVS	VARCHAR2(1)	(Y or N) Insufficient privileges on objects referenced by the existing child cursor
INSUFF_PRIVS_REM	VARCHAR2(1)	(Y or N) Insufficient privileges on remote objects referenced by the existing child cursor
REMOTE_TRANS_ MISMATCH	VARCHAR2(1)	(Y or N) The remote base objects of the existing child cursor do not match

9

Replication

This chapter is an addendum to the release 8.1.6 *Oracle8i Replication* book and describes new features in Oracle replication in release 8.1.7. This chapter also includes documentation additions and documentation updates.

This chapter contains these topics:

- New Replication Features
- Documentation Additions
- Documentation Updates
- The Replication Management Tool in DBA Studio

New Replication Features

This section describes new replication features in release 8.1.7.

Reduced Quiesce for Single Master Replication Environments

At times, you must stop all replication activity for a master group so that you can perform certain administrative tasks on the master group. For example, you must stop all replication activity for a master group to issue data definition language (DDL) statements on any table in the group. Stopping all replication activity for a master group is called quiescing the group. When a master group is quiesced, users cannot perform data manipulation language (DML) statements on any of the objects in the master group. Also, all deferred transactions must be propagated before you can quiesce a master group.

Because transactions cannot be run on tables in a master group when the master group is quiesced, database administrators typically try to avoid quiescing a master group. Release 8.1.7 reduces the number of operations that require you to quiesce a master group.

The following sections describe operations that no longer require quiesce in single master environments. In these sections, the replication management API procedure that corresponds to each operation is listed.

Note: The reduced quiesce feature only applies to single master replication environments, not to multimaster environments.

Reduced Quiesce in Single Master Environments

Table 9–1 lists the operations that no longer require quiesce in single master replication environments. This table also lists the procedure in the DBMS_REPCAT package that corresponds to each operation. In addition, these operations no longer require quiesce when you are using the Replication Management tool in DBA Studio.

Table 9–1 Single Master Operations That No Longer Require Quiesce **(Page 1 of 2)**

Operation	DBMS_REPCAT Procedure
Designating a method for resolving a delete conflict	ADD_DELETE_RESOLUTION
Adding a member to a priority group	ADD_PRIORITY_*datatype*
Adding a new site to a site priority group	ADD_SITE_PRIORITY_SITE
Designating a method for resolving a uniqueness conflict	ADD_UNIQUE_RESOLUTION
Designating a method for resolving an update conflict	ADD_UPDATE_RESOLUTION
Altering the priority level associated with a specified priority group member	ALTER_PRIORITY
Altering the value of a member in a priority group	ALTER_PRIORITY_*datatype*
Altering the priority level associated with a specified site	ALTER_SITE_PRIORITY
Altering the site associated with a specified priority level	ALTER_SITE_PRIORITY_SITE
Specifying whether to compare old column values at each master site for each non-key column of a replicated table when executing updates and deletes	COMPARE_OLD_VALUES
Specifying that an object is a replicated object	CREATE_MASTER_REPOBJECT
Creating a priority group for conflict resolution	DEFINE_PRIORITY_GROUP
Creating a new site priority group for a replicated master group for conflict resolution	DEFINE_SITE_PRIORITY
Dropping a delete conflict resolution method	DROP_DELETE_RESOLUTION
Dropping a member of a priority group by priority level	DROP_PRIORITY
Dropping a member of a priority group by value	DROP_PRIORITY_*datatype*
Dropping a priority group	DROP_PRIORITY_GROUP
Dropping a site priority group	DROP_SITE_PRIORITY
Dropping a specified site, by name, from a site priority group	DROP_SITE_PRIORITY_SITE

Table 9–1 Single Master Operations That No Longer Require Quiesce **(Page 2 of 2)**

Operation	DBMS_REPCAT Procedure
Dropping a uniqueness conflict resolution method	DROP_UNIQUE_RESOLUTION
Dropping an update conflict resolution method	DROP_UPDATE_RESOLUTION
Generating the triggers, packages, and procedures needed to support replication for a specified object	GENERATE_REPLICATION_SUPPORT
Specifying whether to send old column values for each non-key column of a replicated table when executing updates and deletes	SEND_OLD_VALUES

Reduced Quiesce in Single Master Environments with No Updatable Snapshots

If a single master site does not have any updatable snapshots, quiesce is no longer required for the operations listed in Table 9–2. This table also lists the procedure in the DBMS_REPCAT package that corresponds to each operation. If a single master site has updatable snapshots, quiesce is still required for these operations.

Note: These operations still require quiesce if you use the Replication Management tool.

Table 9–2 SIngle Master Operations That No Longer Require Quiesce (No Updatable Snapshots)

Operation	DBMS_REPCAT Procedure
Adding members to an existing column group	ADD_GROUPED_COLUMN
Creating an empty column group	DEFINE_COLUMN_GROUP
Dropping a column group	DROP_COLUMN_GROUP
Removing members from a column group	DROP_GROUPED_COLUMN
Creating a new column group with one or more members	MAKE_COLUMN_GROUP
Specifying use of an alternate column or group of columns, instead of the primary key, to determine which columns of a table to compare when using row-level replication	SET_COLUMNS

Documentation Additions

This section provides additional documentation about Oracle replication that was not included in the release 8.1.6 *Oracle8i Replication* book.

UTL_FILE_DIR Initialization Parameter and Generated Replication Files

The `UTL_FILE_DIR` initialization parameter specifies the default location for generated files. These files may be generated by the Replication Management tool or by the replication management API. For example, an offline instantiation file for a deployment template is placed, by default, in the directory specified by this initialization parameter.

Procedural Replication and Snapshot Sites

The following bullet is added to the list of bullets in the "Restrictions on Procedural Replication" section starting on page 6-2 in the release 8.1.6 *Oracle8i Replication* book:

- When using procedural replication, a procedure call is only propagated to master replication sites. The procedure call is not propagated to snapshot sites. However, procedural replication can be initiated at a snapshot site. In this case, the procedure call is propagated to all of the master sites in the replicated environment, but the procedure call is not propagated to any other snapshot sites. Other snapshot sites must pull changes made at the master site by performing a snapshot refresh.

  For example, suppose a replicated environment includes two master sites named `msite1` and `msite2` and two snapshot sites named `snap1` and `snap2`. If procedural replication is initiated at `snap1`, then the procedure is run at `snap1` and the procedure call is propagated to the two master sites, `msite1` and `msite2`, where the procedure is also run. However, the procedure call is not propagated to `snap2`. Therefore, during the next refresh, `snap2` pulls down all of the changes made by the procedure at its master site.

Avoiding Problems When Adding a New Snapshot Site

After you have created a snapshot environment with one or more snapshot sites, you may need to add new snapshot sites. You may encounter problems when you try to fast refresh the snapshots you create at a new snapshot site if both of the following conditions are true:

- Snapshots at the new snapshot site and existing snapshots at other snapshot sites are based on the same master table.
- Existing snapshots can be refreshed while you create the new snapshots at the new snapshot site.

The problem arises when the snapshot logs for the master tables are purged before a new snapshot can perform its first fast refresh. If this happens and you try to fast refresh the snapshots at the new snapshot site, then you may encounter the following errors:

```
ORA-12004 REFRESH FAST cannot be used for snapshot snapshot_name
ORA-12034 snapshot log on snapshot_name younger than last refresh
```

If you receive these errors, then the only solution is to perform a complete refresh of the new snapshot.

To avoid this problem, you can:

- Use deployment templates to create the snapshot environment at snapshot sites. You will not encounter this problem if you use deployment templates.

  See Also: Chapter 4, "Deployment Templates Concepts & Architecture," in the release 8.1.6 *Oracle8i Replication* book for information about deployment templates

- Create a dummy snapshot at the new snapshot site before you create your production snapshots. The dummy snapshot ensures that the snapshot log will not be purged while your production snapshots are being created.

If you choose to create a dummy snapshot at the snapshot site, complete the following steps:

1. Create a dummy snapshot called `dummy_snap` based on the master table. For example, to create a dummy snapshot based on a master table named `sales`, issue the following statement at the new snapshot site:

   ```
   CREATE SNAPSHOT dummy_snap REFRESH FAST AS
     SELECT * FROM sales@acme.com WHERE 1=0;
   ```

2. Create your production snapshots at the new snapshot site.
3. Perform fast refresh of your production snapshots at the new snapshot site.
4. Drop the dummy snapshot.

Documentation Updates

The following sections are updates to sections in the release 8.1.6 *Oracle8i Replication* book.

Replication of Sequences Is Not Supported

Sequences are included in the list of supported replicated objects in the "Replication Objects" section on page 1-4 in the release 8.1.6 *Oracle8i Replication* book. However, replication of sequences is not supported.

Updated "Base Table" Section

The following section replaces the "Base Table" section on page 3-21 in the release 8.1.6 *Oracle8i Replication* book. The new section has detailed information about base tables and the compatibility level of the database. In addition, the new section corrects an error in the release 8.1.6 book about the truncation of long snapshot names.

Base Table

The way snapshot base tables function depends on the compatibility level of your snapshot database. The compatibility setting is defined by the `COMPATIBLE` initialization parameter in the initialization parameter file.

Base Table with Compatibility at 8.1.0 or Higher If the compatibility setting is 8.1.0 or higher, the following applies:

- The base table is the actual snapshot (an additional view is not required).
- The size limit for a snapshot name is 30 bytes. If you try to create a snapshot with a name larger than 30 bytes, Oracle returns an error.
- The snapshot has the name that you specified during snapshot creation.

Base Table with Compatibility Lower Than 8.1.0 If the compatibility setting is lower than 8.1.0, the following applies:

- The base table is the underlying support object for a view, and the view is the snapshot.
- The size limit for a snapshot name is 30 bytes. If you try to create a snapshot with a name larger than 30 bytes, Oracle returns an error.
- The snapshot has the name that you specified during snapshot creation.
- Any indexes generated when you create the snapshot are created on the base table.
- When the snapshot name is less than or equal to 19 bytes, the base table is named `SNAP$_`*snapshot_name*.
- When the snapshot name is between 20 and 30 bytes, the base table name is truncated to 20 bytes, prefixed by `SNAP$_`, and possibly postfixed by a number. If no other base table at the snapshot site has the same name as the truncated name, then nothing is added to the truncated name. If another base table at the snapshot site has the same name as the truncated name, then the first number postfixed is 1, the second postfixed is 2, and so on up to 9999. Therefore, the name of the base table is `SNAP$_`, followed by the first 20 bytes of the snapshot name, followed by nothing or a one to four digit number.

  For example, a snapshot named `abcdefghijklmnopqrstuvwxyz` has a base table named `SNAP$_abcdefghijklmnopqrst`, assuming no other base table has the same name. A subsequently created snapshot at the same snapshot site named `abcdefghijklmnopqrstuvwxy` has a base table named `SNAP$_abcdefghijklmnopqrst1`.

Note: The compatibility setting for Oracle release 8.0 databases must be lower than 8.1.0.

See Also: "Initialization Parameters" on page 7-3 in the release 8.1.6 *Oracle8i Replication* book and *Oracle8i Migration* for more information about the `COMPATIBLE` parameter, and see "View" on page 3-22 in the release 8.1.6 *Oracle8i Replication* book for more information about the view that is created in support of snapshots with a compatibility level lower than 8.1.0.

Guidelines for Scheduled Links

This section replaces the "Guidelines for Scheduled Links" section that starts on page 7-10 in the release 8.1.6 *Oracle8i Replication* book.

A scheduled link determines how a master site propagates its deferred transaction queue to another master site, or how a snapshot site propagates its deferred transaction queue to its master site. When you create a scheduled link, Oracle creates a job in the local job queue to push the deferred transaction queue to another site in the system. When Oracle propagates deferred transactions to a remote master site, it does so within the security context of the replication propagator.

You can configure a scheduled link to push information using serial or parallel propagation. In general, you should use parallel propagation, even if you set parallel propagation to 1.

Before creating the scheduled links for a replication environment, carefully consider how you want replication to occur globally throughout the system. For example, you may choose to propagate deferred transactions at intervals, with time in between these intervals when the deferred transactions are not propagated. In this case, you must decide how often and when to schedule pushes. Alternatively, if you want to simulate real-time (or synchronous) replication, then you may want to have each scheduled link continuously push a master site's deferred transaction queue to its destination.

Also, you may want to schedule pushes at a time of the day when connectivity is guaranteed or when communications costs are lowest, such as during evening hours. Furthermore, you may want to stagger the scheduling for links among all master sites to distribute the load that replication places on network resources.

> **See Also:** "Serial and Parallel Propagation" on page 9-14 for more information about issues related to serial and parallel propagation

Scheduling Periodic Pushes

You can schedule periodic intervals between pushes of a site's deferred transaction queue to a remote destination. Examples of periodic intervals are once an hour or once a day. To do so, you can use the `DBMS_DEFER_SYS.SCHEDULE_PUSH` procedure and specify the settings shown in Table 9–3.

Table 9–3 Settings to Schedule Periodic Pushes

SCHEDULE_PUSH Procedure Parameter	Value
`delay_seconds`	0
`interval`	An appropriate date expression; for example, to specify an interval of one hour, use `'sysdate + 1/24'`

You can also use the Replication Management tool to schedule periodic pushes. To do so, set Delay Seconds to the default value of 0 when configuring a scheduled link in any of the following places:

- The Replication Management tool's Setup Wizard
- The Edit Push Schedule dialog box

Then configure the interval (the "then push every" control) to push the deferred transaction queue periodically.

> **See Also:**
>
> - "Delay Seconds" on page 2-39 in the release 8.1.6 *Oracle8i Replication* book for more information about setting delay seconds
> - *Oracle8i Replication Management API Reference* for information about the `DBMS_DEFER_SYS.SCHEDULE_PUSH` procedure
> - The Replication Management tool online help for information about using this tool

Scheduling Continuous Pushes

Even when using Oracle's asynchronous replication mechanisms, you can configure a scheduled link to simulate continuous, real-time replication. To do so, use the `DBMS_DEFER_SYS.SCHEDULE_PUSH` procedure and specify the settings shown in Table 9–4.

Table 9–4 Settings to Simulate Continuous Push

SCHEDULE_PUSH Procedure Parameter	Value
`delay_seconds`	1200
`interval`	Lower than the `delay_seconds` setting
`parallelism`	1 or higher
`execution_seconds`	Higher than the `delay_seconds` setting

With this configuration, Oracle continues to push transactions that enter the deferred transaction queue for the duration of the entire interval. If the deferred transaction queue has no transactions to propagate for the amount of time specified by the `delay_seconds` parameter, then Oracle releases the resources used by the job and starts fresh when the next SNP process becomes available.

If you are using serial propagation by setting the `parallelism` parameter to 0 (zero), you can simulate continuous push by reducing the settings of the `delay_seconds` and `interval` parameters to an appropriate value for your environment. However, if you are using serial propagation, simulating continuous push is costly when the push job must initiate often.

See Also:

- "Delay Seconds" on page 2-39 in the release 8.1.6 *Oracle8i Replication* book for more information about setting delay seconds
- "Serial and Parallel Propagation" on page 9-14 for more information about issues related to serial and parallel propagation
- *Oracle8i Replication Management API Reference* for information about the `DBMS_DEFER_SYS.SCHEDULE_PUSH` procedure

Guidelines for Scheduled Purges of a Deferred Transaction Queue

This section replaces the "Guidelines for Scheduled Purges of a Deferred Transaction Queue" section that starts on page 7-11 in the release 8.1.6 *Oracle8i Replication* book.

A scheduled purge determines how a master or snapshot site purges applied transactions from its deferred transaction queue. When you use the Replication Management tool's Setup Wizard to create a master or snapshot site, Oracle creates a job in each site's local job queue to purge the local deferred transaction queue on a regular basis. Carefully consider how you want purging to occur before configuring the sites in a replication environment. For example, consider the following options:

- You can synchronize the pushing and purging of a site's deferred transaction queue. For example, you can configure continuous pushing and purging of the transaction queue. This type of configuration can offer performance advantages because it is likely that information about recently pushed transactions is already in the server's buffer cache for the corresponding purge operation.
- When a server is not CPU bound, you can schedule continuous purging of the deferred transaction queue to keep the size of the queue as small as possible.
- For servers that experience a high-volume of transaction throughput during normal business hours, you can schedule purges to occur during off-peak hours if you can store an entire day's deferred transactions.

Scheduling Periodic Purges

You can schedule periodic purges of a site's deferred transaction queue. Examples of periodic purges are purges that occur once a day or once a week. To do so, you can use the `DBMS_DEFER_SYS.SCHEDULE_PURGE` procedure and specify the settings shown in Table 9–5.

Table 9–5 Settings to Schedule Periodic Purges

SCHEDULE_PURGE Procedure Parameter	Value
`delay_seconds`	0
`interval`	An appropriate date expression; for example, to specify an interval of one day, use `'sysdate + 1'`

You can also use the Replication Management tool's Setup Wizard, or the Purge sub tab of the Schedule tab on the Administration property sheet to schedule periodic purges. To do so, set Delay Seconds to the default value of 0 (zero). Then configure the interval (the "then purge every" control) to purge the deferred transaction queue.

See Also:

- "Delay Seconds" on page 2-39 in the release 8.1.6 *Oracle8i Replication* book for more information about setting delay seconds
- *Oracle8i Replication Management API Reference* for information about the `DBMS_DEFER_SYS.SCHEDULE_PURGE` procedure
- The Replication Management tool online help for information about using this tool

Scheduling Continuous Purges

To configure continuous purging of a site's deferred transaction queue, you can use the `DBMS_DEFER_SYS.SCHEDULE_PURGE` procedure and specify the settings shown in Table 9–6.

Table 9–6 Settings to Schedule Periodic Purges

SCHEDULE_PURGE Procedure Parameter	Value
`delay_seconds`	500000
`interval`	Lower than the `delay_seconds` setting

You can also use the Replication Management tool to configure continuous purge. To do so, on the Purge sub tab of the Schedule tab on the Administration property sheet, set Delay Seconds to 500,000 and set interval (the "then purge every" control) to a value less than the Delay Seconds setting.

See Also:

- "Delay Seconds" on page 2-39 in the release 8.1.6 *Oracle8i Replication* book for more information about setting delay seconds
- *Oracle8i Replication Management API Reference* for information about the `DBMS_DEFER_SYS.SCHEDULE_PURGE` procedure
- The Replication Management tool online help for information about using this tool

Serial and Parallel Propagation

This section replaces the "Serial and Parallel Propagation" section that starts on page 7-12 in the release 8.1.6 *Oracle8i Replication* book.

When you create the scheduled links for a replication environment, each link can asynchronously propagate changes to a destination using either serial or parallel propagation. Before you configure your replication environment, decide whether you want to use serial propagation or parallel propagation.

- With serial propagation, Oracle propagates replicated transactions one at a time in the same order that they are committed on the source system. To configure a scheduled link with serial propagation, set the `parallelism` parameter to 0 (zero) in the `DBMS_DEFER_SYS.SCHEDULE_PUSH` procedure. Or, using the Replication Management tool, set the Parallel Propagation Processes control to 0 in the Edit Push Schedule dialog box. Typically, you should use serial propagation only when the destination is an Oracle7 site.
- With parallel propagation, Oracle propagates replicated transactions using multiple parallel streams for higher throughput. When necessary, Oracle orders the execution of dependent transactions to preserve data integrity. To configure a scheduled link with parallel propagation, set the `parallelism` parameter to 1 or higher in the `DBMS_DEFER_SYS.SCHEDULE_PUSH` procedure. Or, using the Replication Management tool, set the Parallel Propagation Processes control to 1 or higher in the Edit Push Schedule dialog box.

See Also:

- "Parallel Propagation" on page 2-36 in the release 8.1.6 *Oracle8i Replication* book
- *Oracle8i Replication Management API Reference* for information about the `DBMS_DEFER_SYS` package
- The Replication Management tool online help for information about using this tool

The Replication Management Tool in DBA Studio

In release 8.1.6, Replication Manager was a stand-alone tool. In release 8.1.7, this tool is part of DBA Studio and is called the Replication Management tool. As part of DBA Studio, the Replication Management tool can be accessed through a standard web browser. In addition, Oracle made many improvements to the interface of this tool in release 8.1.7.

The Replication Management tool provides a graphical user interface (GUI) for setting up, managing, and monitoring a replication environment. The Replication Management tool includes wizards that guide you through many important operations. You can use the Replication Management tool to manage both multimaster and snapshot replication environments.

The primary documentation for using this tool is the Replication Management tool online help. The following sections introduce you to the new Replication Management tool. These sections completely replace Chapter 8, "Introduction to Replication Manager," in the release 8.1.6 *Oracle8i Replication* book.

- Usage Scenarios for the Replication Management Tool
- Logging in to the Replication Management Tool
- The Replication Management Tool Interface
- The Replication Management Tool Wizards
- Flowchart for Creating a Replicated Environment

Note: Other chapters in the release 8.1.6 *Oracle8i Replication* book also refer to specific controls in the Replication Management tool. Because the interface for the Replication Management tool has changed in release 8.1.7, these references may no longer be valid.

See Also:

- The Replication Management tool online help for detailed instructions about using the Replication Management tool
- The DBA Studio online help for information about using DBA Studio
- Oracle Enterprise Manager documentation set and online help for information about using Oracle Enterprise Manager

Usage Scenarios for the Replication Management Tool

Using the Replication Management tool, you can:

- Set up master sites and snapshot sites
- Add master sites to and remove master sites from a replication environment
- Create and manage master groups
- Monitor master sites and snapshot sites with a topology view
- Create and manage snapshot logs
- Create and manage snapshot groups
- Create and manage individual snapshots
- Create and manage refresh groups for snapshots
- Create and manage deployment templates
- Package deployment templates for offline instantiation. Note that you must use the replication management API to package deployment templates for online instantiation.
- Configure conflict resolution methods
- Create, monitor, and manage scheduled links
- Monitor and manage administration requests
- Monitor and manage deferred transactions
- Monitor and manage error transactions
- Create, monitor, and manage local jobs

Logging in to the Replication Management Tool

Assuming DBA Studio is configured to show the databases in your replication environment, complete the following steps to log in to the Replication Management tool:

1. Open DBA Studio.

   See Also: The Oracle Enterprise Manager documentation set for information about configuring and opening DBA Studio

2. Expand a database node. The Database Connect Information dialog box appears.
3. Enter the username and password for the database in the Database Connect Information dialog box.

   If you have an established replication environment, log in as the replication administrator. If you have not set up a replication environment, then log in as `SYSTEM` or `SYS` user. Next, use the Setup Wizard to set up your master definition site and, if you want a multimaster replication environment, your other master sites.

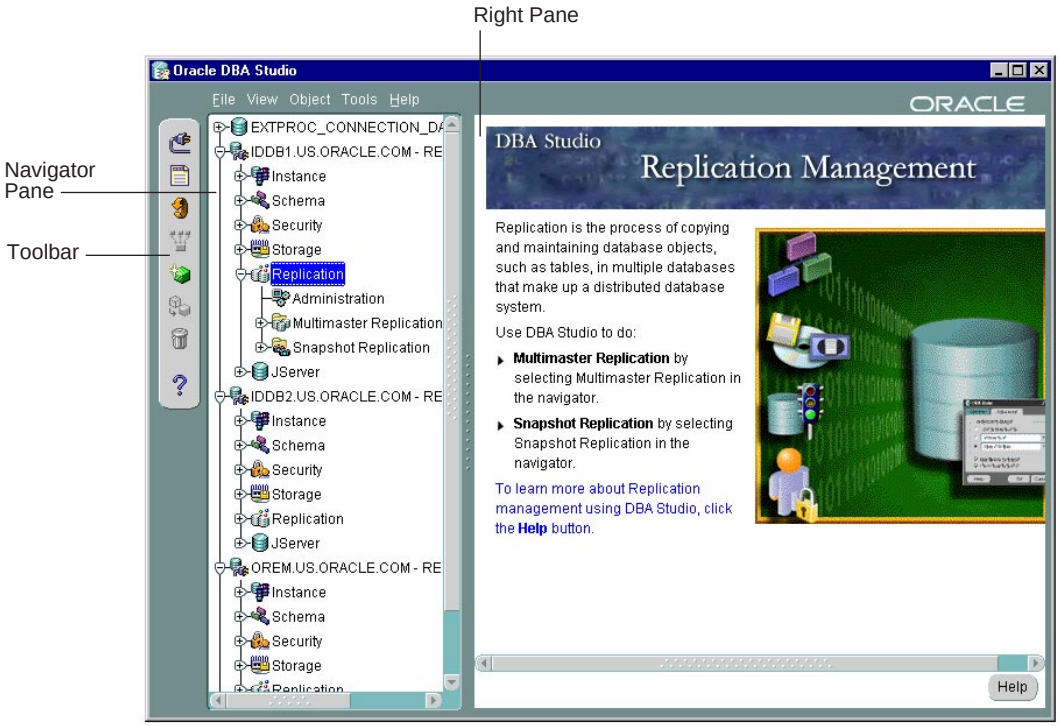
   When setup is complete, log in to the Replication Management tool as the replication administrator you specified in the Setup Wizard. If you follow convention, then the username of the replication administrator is `repadmin`. You should only log in to the Replication Management tool as the replication administrator, not as any other user, after setup is complete.

4. Select Replication in the navigator pane.

The Replication Management Tool Interface

The Replication Management tool interface in DBA Studio includes a toolbar and two panes: the navigator pane and the right pane.

Figure 9–1 The Replication Management Tool Interface



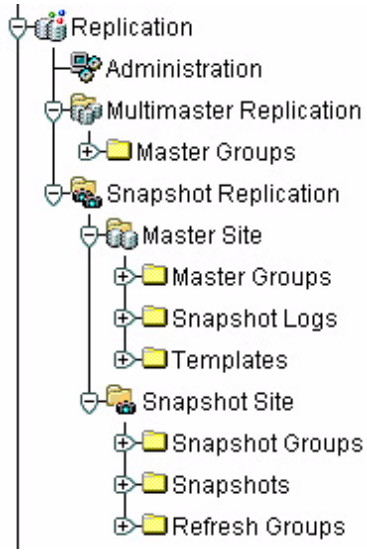
Navigator Pane

The navigator pane in the DBA Studio functions the same way as it does in other Oracle Enterprise Manager applications. That is, the navigator pane lets you:

- Access all of the nodes in your replication environment
- Expand and collapse objects and folders so that you can navigate to the object you want to monitor or manage. Examples of objects are master groups, snapshot groups, snapshots, snapshot logs, deployment templates, and so on.
- Right-click on a folder or object to create a new object or perform operations on an existing object

Oracle completely redesigned the navigator tree structure in the Replication Management tool for release 8.1.7. You begin many replication administration operations by selecting the correct object or folder in the navigator tree of the Replication Management tool.

Figure 9–2 Replication Management Tool Navigator Tree



The following sections describe the contents of each object and folder in the tree structure.

Administration Object The Administration object enables you to manage the entire replication site. Select the Administration object to display the following tabs in the right pane:

- **Topology:** Displays a graphical view of the replication environment for the selected site.
- **Errors:** Displays and lets you manage local errors at the selected site.
- **Transactions:** Displays and lets you manage the deferred transactions at the selected site.
- **Schedule:** Displays and lets you manage the push schedules for the database links at the selected site. Also displays and lets you manage the purge schedule for successfully propagated deferred transactions at the selected site.
- **Configuration:** Displays and lets you manage configuration information at the selected site, including initialization parameters important for replication and administration requests.
- **DBMS Jobs:** Displays and lets you manage the jobs created with the `DBMS_JOBS` package. Many of these jobs perform important replication functions, such as pushing deferred transactions and purging the deferred transactions queue.

Multimaster Replication Object If your replication environment is configured for multimaster replication, you use the Multimaster Replication object to set up your master sites and manage your master groups. To set up master sites with the Setup Wizard, right-click the Multimaster Replication object and select Setup Master Sites.

Note: Oracle8*i* Enterprise Edition is required for multimaster replication. If your installation is Oracle8*i* server (Standard Edition), you can have only one master site for each master group.

The Master Groups folder under the Multimaster Replication Object contains the master groups at the selected site. When you select a master group, you can:

- Start or stop (quiesce) the master group
- View and apply administration requests for the master group
- Purge administration requests for the master group
- Perform DDL operations on objects in the master group
- Manage the replicated objects in a master group
- Manage the replication sites participating in the master group

To create a new master group, right-click the Master Groups folder and select Create.

Snapshot Replication Object The Snapshot Replication object lets you administer the replication site as it relates to snapshot replication. You can administer sites that are master sites of snapshot sites, and you can administer snapshot sites themselves. To set up master sites or snapshot sites with the Setup Wizard, right-click the Snapshot Replication object and select Setup Sites.

The Snapshot Replication object contains the following objects:

- Master Site object
- Snapshot Site object

Note: A replication site can be:

- A master site only, if it contains master groups but no snapshots. In this case, use the Master Site object to manage the site.
- A snapshot site only, if it contains snapshots but no master groups. In this case, use the Snapshot Site object to manage the site.
- Both a master site and a snapshot site, if it contains both master groups and snapshots. In this case, use both the Master Site object and the Snapshot Site object to manage the site.

The Master Site object under the Snapshot Replication object lets you manage the master groups at the replication site, and lets you manage sites that are master sites of snapshot sites. Specifically, this object lets you create, manage, and package deployment templates, and lets you create and manage snapshot logs. To set up master sites with the Setup Wizard, right-click the Master Site object and select Setup Master Sites.

The Master Site object has the following folders:

- **Master Groups Folder:** This folder provides the same functions as the Master Groups folder under the Multimaster Replication object. See the description of the Master Groups folder in "Multimaster Replication Object" on page 9-20 for information.
- **Snapshot Logs Folder:** This folder lets you create and manage snapshot logs at the master site. To create a new snapshot log, right-click the Snapshot Logs folder and select Create.
- **Templates Folder:** This folder lets you create, manage, and package deployment templates at the master site. Right-click the Templates folder and select:
  - Create Using Wizard if you want to create a new deployment template with the Deployment Template Wizard
  - Copy if you want to create a local or remote copy of a deployment template with the Copy Template Wizard
  - Compare if you want to compare two deployment templates
  - Offline Instantiation File Generation if you want to package a deployment template for offline instantiation with the Offline Instantiation Wizard

The Snapshot Site object under the Snapshot Replication object lets you manage snapshot sites. Specifically, this object lets you create and manage snapshot groups, snapshots, and refresh groups. To set up snapshot sites with the Setup Wizard, right-click the Snapshot Site object and select Setup Snapshot Sites.

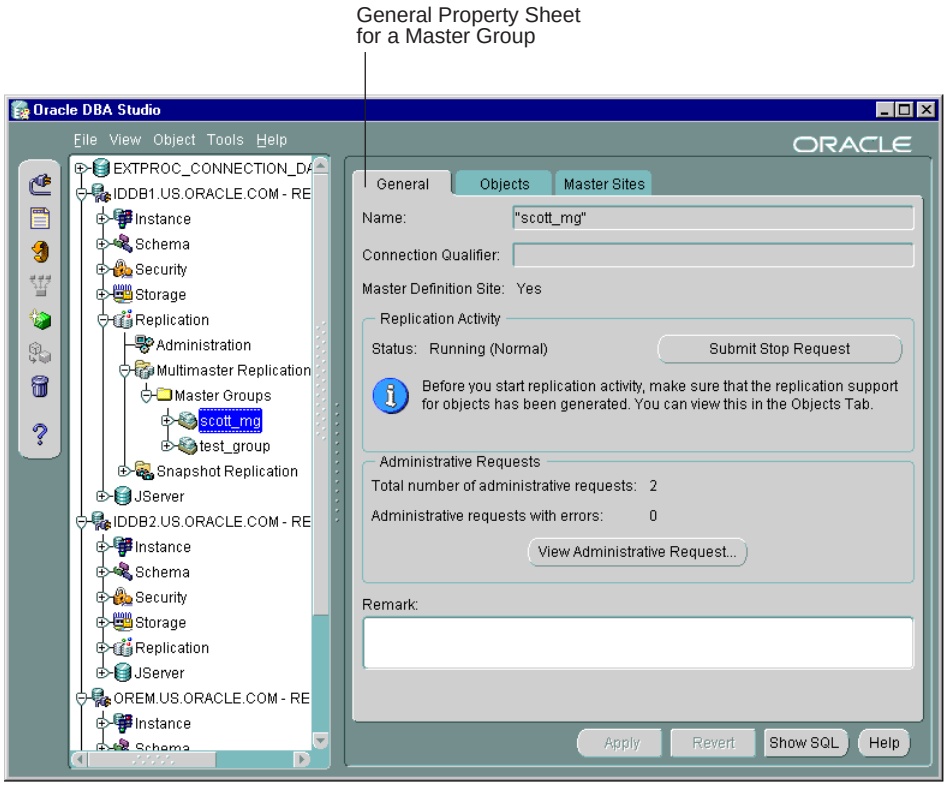
The Snapshot Site object has the following folders:

- **Snapshot Groups Folder:** This folder lets you create and manage snapshot groups at the snapshot site. To create a new snapshot group with the Snapshot Group Wizard, right-click the Snapshot Groups folder and select Create Using Wizard.
- **Snapshots Folder:** This folder lets you create and manage individual snapshots at the snapshot site. To create a new snapshot, right-click the Snapshots folder and select Create.
- **Refresh Groups Folder:** This folder lets you create and manage refresh groups at the snapshot site. To create a new refresh group, right-click the Refresh Groups folder and select Create.

Right Pane

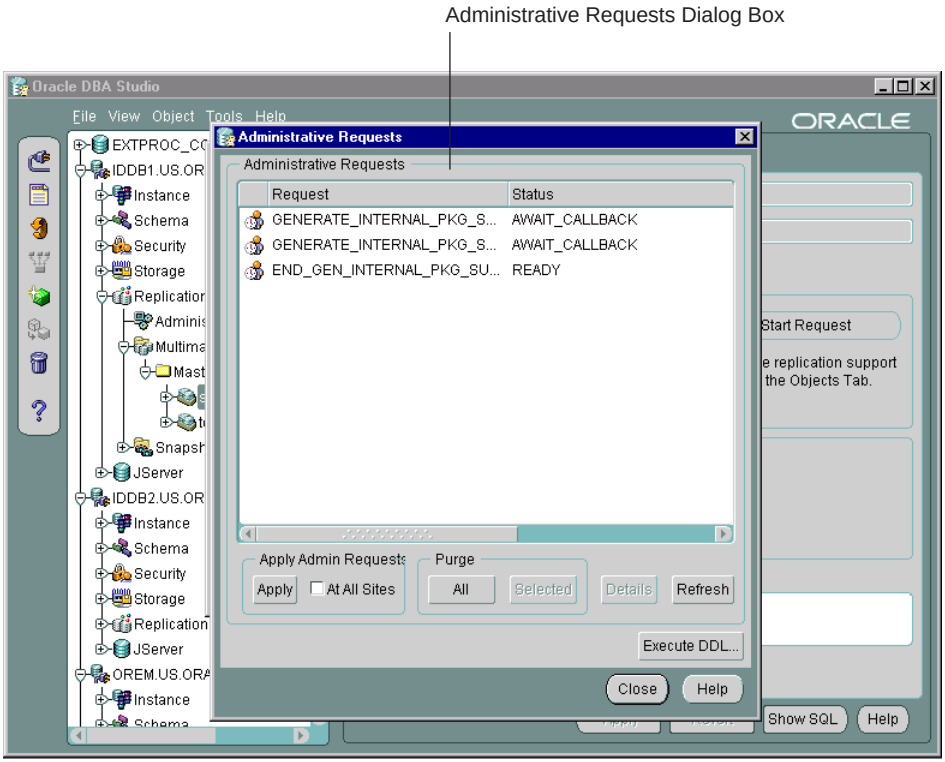
The right pane of the Replication Management tool enables you to monitor and manage your replication environment. Figure 9–3 shows an example of a property sheet in the right pane.

Figure 9–3 Example Property Sheet



When you are working with a property sheet, you may click on a button that opens a new dialog box. For example, if you click the View Administrative Request button on the General property sheet for a master group, then the Administrative Requests dialog box appears, as shown in Figure 9–4.

Figure 9–4 Example Dialog Box



Topology Tab You can also use the right pane of the Replication Management tool to monitor your replication environment. The Topology tab of the Administration property sheet is an example of a monitoring tool that is available to you. This tab displays a graphical representation of the nodes in your replication environment and the links between them.

Specifically, the Topology tab provides the following information:

- All of the master sites participating in replication activity for the master groups of the selected site
- The number of deferred transactions at each master site
- The destination site for deferred transactions
- The number of administration requests at each master site
- Whether local errors exist at a master site and the number of errors
- The total number of the snapshots or snapshot groups connected to the selected master site
- Whether the master site is also functioning as a snapshot site

The Topology tab only displays values if they are greater than zero. For example, if there are zero administration requests at a master site, the number of administration requests is not displayed.

Note: If you have not created at least one master group, snapshot group, or snapshot at the selected site, then the Topology tab displays a message stating that replication must be configured.

The Topology tab displays the following icons and images:

Indicates a master site. The current number of administration requests being processed at the master is displayed next to the icon.

Indicates a master site with errors. The number of errors is displayed next to the icon.

Indicates a site that is functioning as both a master site and a snapshot site (a dual site).

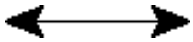
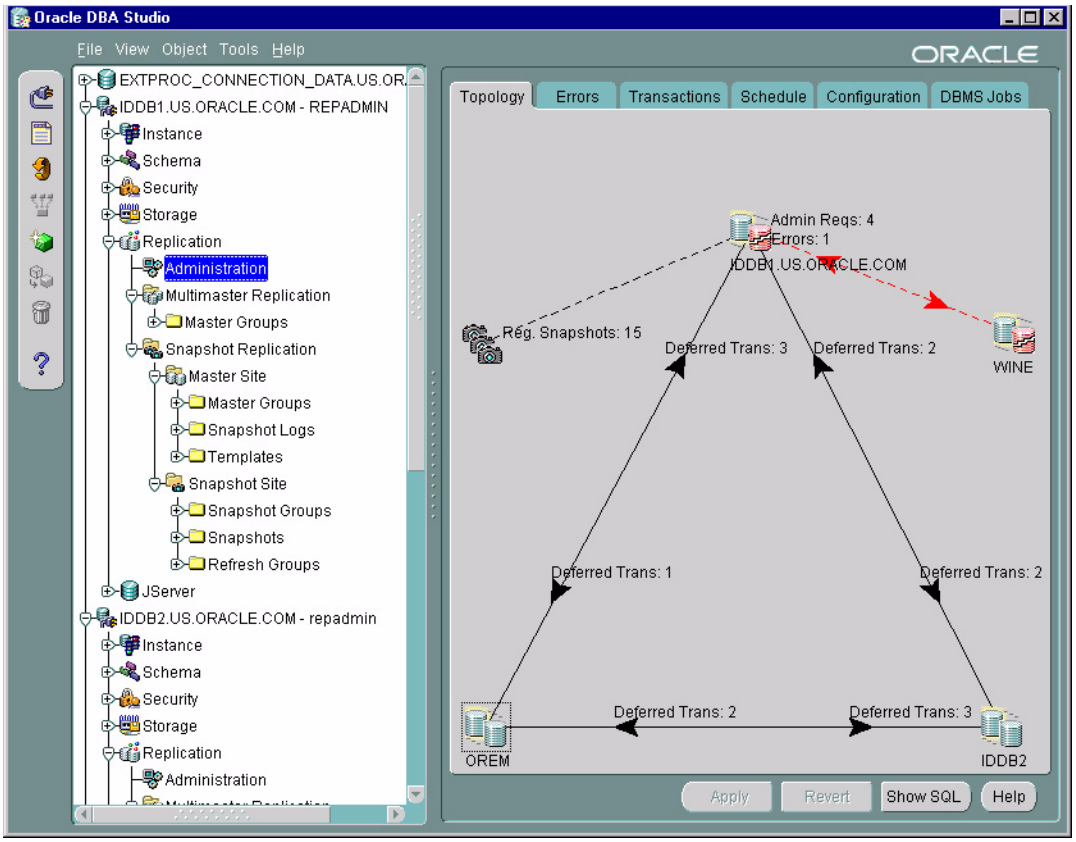
	Indicates a dual site with errors. The number of errors is displayed next to the icon.
	When you are connected to a master site, indicates that the master site has snapshot sites. Snapshots and/or snapshot groups may be registered at the master site.
	When you are connected to a snapshot site, indicates the snapshot site.
	Indicates a snapshot site with errors. The number of errors is displayed next to the icon.
 (Solid black arrow)	Indicates a working database link between two master sites. The current number of deferred transactions is displayed next to the arrow.
(Dashed red arrow)	Indicates a broken database link between two master sites. The current number of deferred transactions is displayed next to the arrow. If you are not logged in as the replication administrator (typically `repadmin` user), then a database link may appear broken even though the link is working normally. The link appears broken because the current user does not have a private database link between the sites, while the replication administrator does have a private database link. In this case, log in to the site as the replication administrator to see if the link is broken.
(Dashed black line)	Indicates a database link between a master site and a snapshot site. There may or may not be a network connection currently between the two sites.

Figure 9–5 Example Topology Tab in the Replication Management Tool



When connected to the master database `iddb1`, this example Topology tab provides the following information:

- The master database `iddb1` has four administration requests to process.
- The master database `iddb1` has one local error. A red icon indicates an error.
- The master database `iddb2` has no administration requests to process and no local errors.
- The master database `orem` has no administration requests to process and no local errors.

- A database link is broken between the `iddb1` master site and the `wine` master site. The dashed red arrow indicates a broken link.
- The master database `iddb1` has one deferred transaction in its deferred transaction queue that will be applied at `orem`.
- The master database `iddb1` has two deferred transactions in its deferred transaction queue that will be applied at `iddb2`.
- The master database `orem` has three deferred transactions in its deferred transaction queue that will be applied at `iddb1`.
- The master database `orem` has three deferred transactions in its deferred transaction queue that will be applied at `iddb2`.
- The master database `iddb2` has two deferred transactions in its deferred transaction queue that will be applied at `iddb1`.
- The master database `iddb2` has two deferred transactions in its deferred transaction queue that will be applied at `orem`.
- The master database `iddb1` has 15 registered snapshots. These registered snapshots may be spread over several snapshot sites.

The other master databases in this replication environment may also have snapshot sites, but they are only visible on the Topology tab when you are connected to those other databases. For example, to see if the master database `orem` has any snapshot sites, connect to the `orem` database in DBA Studio and open the Topology tab.

The Replication Management Tool Wizards

The Replication Management tool wizards provide step-by-step guidance for tasks that require many steps. The wizards simplify complex tasks by guiding you through a task in manageable steps. The following sections describe the Replication Management tool wizards:

- Setup Wizard
- Snapshot Group Wizard
- Deployment Template Wizard
- Offline Instantiation Wizard
- Copy Template Wizard

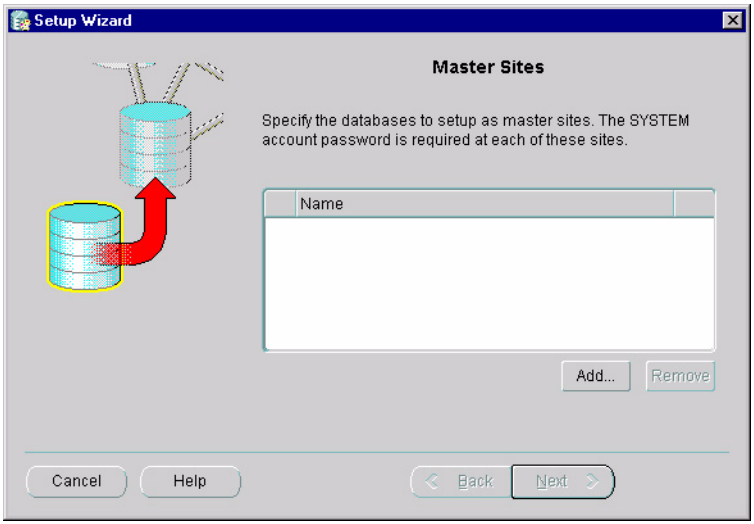
Setup Wizard

The Setup Wizard guides you through setting up master sites and snapshot sites for replication. Preparing sites for replication is a simple process using the Setup Wizard. At each site that you specify, the Setup Wizard performs the following steps:

- Creates a database account to serve as a replication administrator. By default, the Setup Wizard creates this account to serve also as the replication propagator and receiver. The default username for the replication administrator at a master site is `repadmin`, and the default username at a snapshot site is `snapadmin`. However, you can change these usernames if you wish.
- Grants the necessary privileges to the replication administrator account.
- Creates database links to correspond to new replication administrator accounts at each site.
- For master sites, schedules a job to push changes from the master site to each other master site.
- Schedules a job to purge the deferred transaction queue of completed transactions for all sites in the system.

The Setup Wizard is slightly different for master sites than for snapshot sites. To open the Setup Wizard to set up master sites, right-click the Multimaster Replication object in the navigator pane and select Setup Master Sites.

Figure 9–6 Opening Screen of Setup Wizard for Master Sites

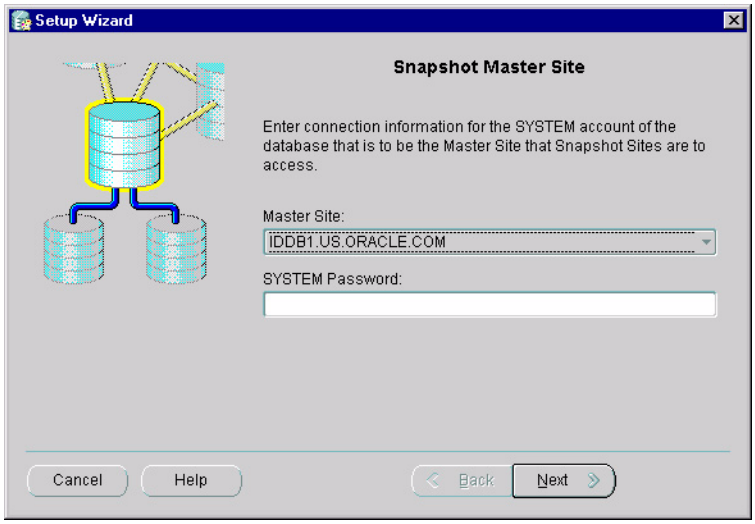


See Also: The "Setup Master Site: Overview" topic in the Replication Management tool online help for detailed information about using the Setup Wizard to set up a master site. To access this topic in the online help, open Replication > Set Up Replication Sites > Master Site in the Help Contents.

To open the Setup Wizard to set up snapshot sites, right-click the Snapshot Site object in the navigator pane and select Setup Snapshot Sites.

Note: The Setup Wizard sets up snapshot sites with untrusted security. See Appendix A, "Security Options," in the release 8.1.6 *Oracle8i Replication Management API Reference* for more information about untrusted security.

Figure 9–7 Opening Screen of Setup Wizard for Snapshot Sites



See Also: The "Set Up Snapshot Site: Overview" topic in the Replication Management tool online help for detailed information about using the Setup Wizard to set up a snapshot site. To access this topic in the online help, open Replication > Set Up Replication Sites > Snapshot Site in the Help Contents.

See Also:

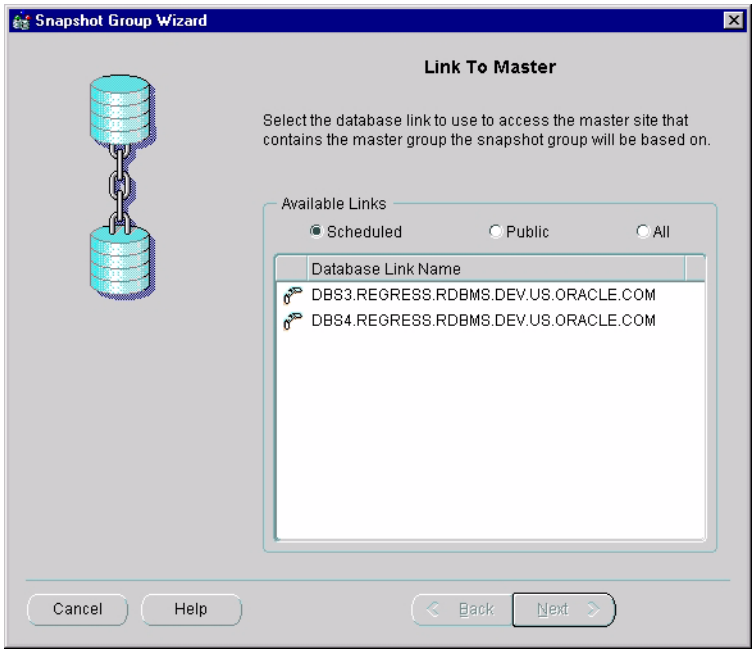
- Chapter 2, "Master Replication Concepts & Architecture," in the release 8.1.6 *Oracle8i Replication* book
- Chapter 3, "Snapshot Concepts & Architecture," in the release 8.1.6 *Oracle8i Replication* book

Snapshot Group Wizard

The Snapshot Group Wizard guides you through creating a group of snapshots based on a master group. Each snapshot can be a partial or complete copy of the master table from its source master group. Snapshot groups are located at remote snapshot sites and are based on a single, target master group at a master site.

Run the Snapshot Group Wizard at the snapshot site where you want to create the snapshot group. To open the Snapshot Group Wizard, right-click the Snapshot Groups folder in the navigator pane and select Create Using Wizard.

Figure 9–8 Opening Screen of the Snapshot Group Wizard



See Also: The "Create Snapshot Group: Overview" topic in the Replication Management tool online help for detailed information about using the Snapshot Group Wizard to create a snapshot group. To access this topic in the online help, open Replication > Snapshot Replication > Create in the Help Contents.

See Also: Chapter 3, "Snapshot Concepts & Architecture," in the release 8.1.6 *Oracle8i Replication* book

Deployment Template Wizard

Deployment templates simplify the task of deploying and maintaining many remote snapshot sites. Using deployment templates, you can define a collection of snapshot definitions at a master site and use parameters in these definitions to customize the snapshots for individual users or types of users.

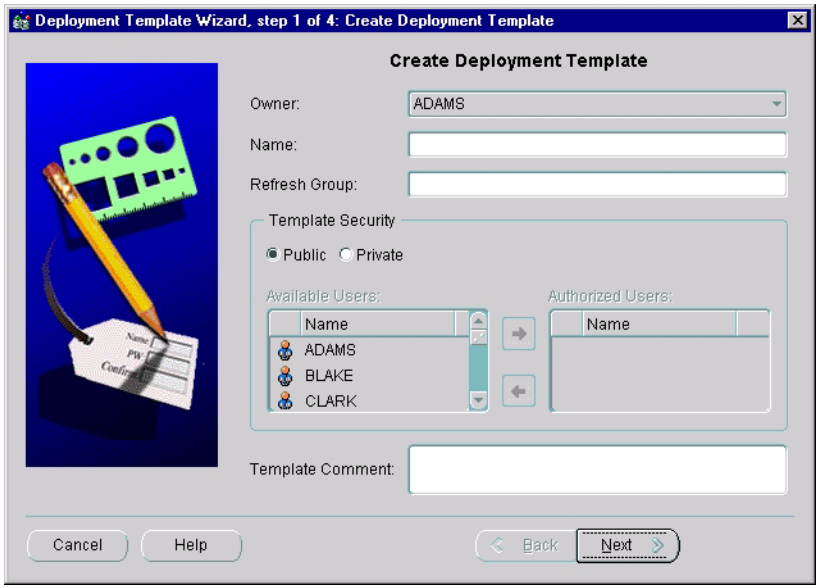
For example, you might create one template for the sales force and another template for field service representatives. In this case, a parameter value might be the sales territory or the customer support level. When a user instantiates a template by running a SQL script, the appropriate snapshots are created and populated at the remote site.

The Deployment Template Wizard guides you through creating a deployment template. Individual screens in the Deployment Template Wizard let you:

- Name the deployment template and specify whether it is public or private. If it is private, then you can specify authorized users.
- Add objects to the deployment template
- Specify parameters for the deployment template

Run the Deployment Template Wizard from the master site where you want to create the deployment template. To open the deployment template wizard, right-click the Templates folder in the navigator pane and select Create Using Wizard.

Figure 9–9 Opening Screen of the Deployment Template Wizard



See Also: The "Overview of Creating a Deployment Template" topic in the Replication Management tool online help for detailed information about using the Deployment Template Wizard to create a deployment template. To access this topic in the online help, open Replication > Deployment Templates > Create in the Help Contents.

See Also: Chapter 4, "Deployment Templates Concepts & Architecture," in the release 8.1.6 *Oracle8i Replication* book

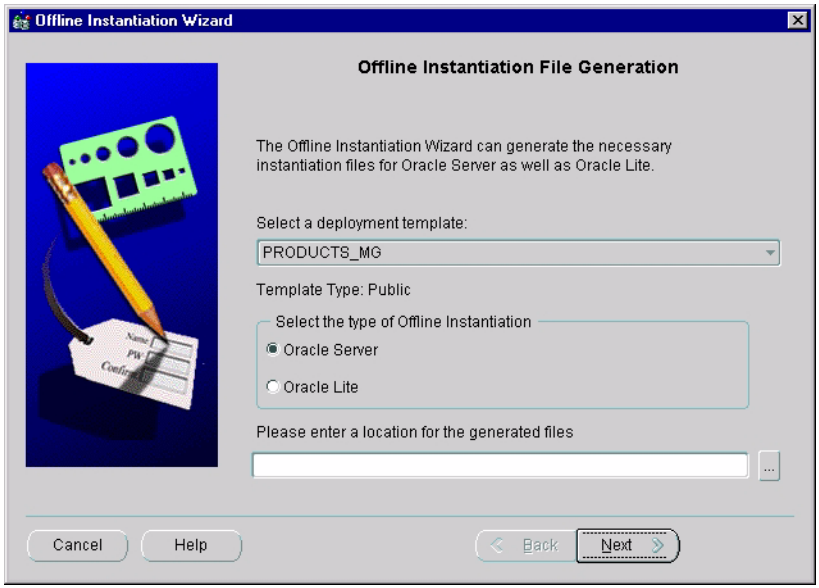
Offline Instantiation Wizard

Offline instantiation allows an end user to use a generated file to instantiate a template without being connected to the master site through a network. The Offline Instantiation Wizard enables you to write all of the necessary DDL and data to a file that you then transfer and run at your snapshot site. This solution is best suited for laptop users with low-speed WAN connections, or in other cases where connectivity is unstable or slow.

The Offline Instantiation Wizard guides you through packaging a deployment template for offline instantiation. The Offline Instantiation Wizard generates offline instantiation files that you use to build snapshots and other objects at your snapshot sites.

Run the Offline Instantiation Wizard at the master site that contains the template for which you want to generate the offline instantiation files. To run the Offline Instantiation Wizard, right-click the Templates folder in the navigator pane and select Offline Instantiation File Generation.

Figure 9–10 Opening Screen of the Offline Instantiation Wizard



Note: To generate online instantiation files, you must use the replication management API. See the *Oracle8i Replication Management API Reference* for information about generating online instantiation files.

See Also: The "Package for Offline Instantiation: Overview" topic in the Replication Management tool online help for detailed information about using the Offline Instantiation Wizard to package a deployment template for offline instantiation. To access this topic in the online help, open Replication > Deployment Templates > Packaging and Instantiation in the Help Contents.

> **See Also:** Chapter 4, "Deployment Templates Concepts & Architecture," in the release 8.1.6 *Oracle8i Replication* book for more information on packaging and instantiating deployment templates

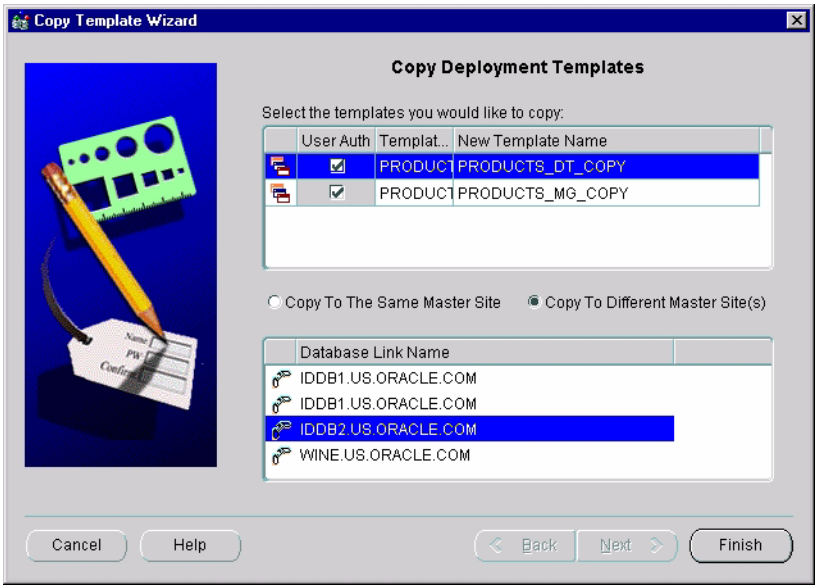
Copy Template Wizard

The Copy Template Wizard guides you through copying a deployment template. You may need to copy them to multiple master sites for various reasons. For example, you may want to:

- **Distribute network load:** Before allowing users to instantiate the template, you need to locate the template at the master site of the target snapshot sites. If you have a large network, then you may want to copy the template definition to multiple master sites, thereby distributing the network load of multiple snapshot sites.
- **Make changes to a template:** After building a template, you may need to create another template that has many of the same characteristics of the first template. Instead of building an entirely new template, copy the template and modify it as necessary.

Run the Copy Template Wizard from the master site that contains the deployment template. To open the Copy Template Wizard, right-click the Templates folder in the navigator pane and select Copy.

Figure 9–11 Copy Template Wizard

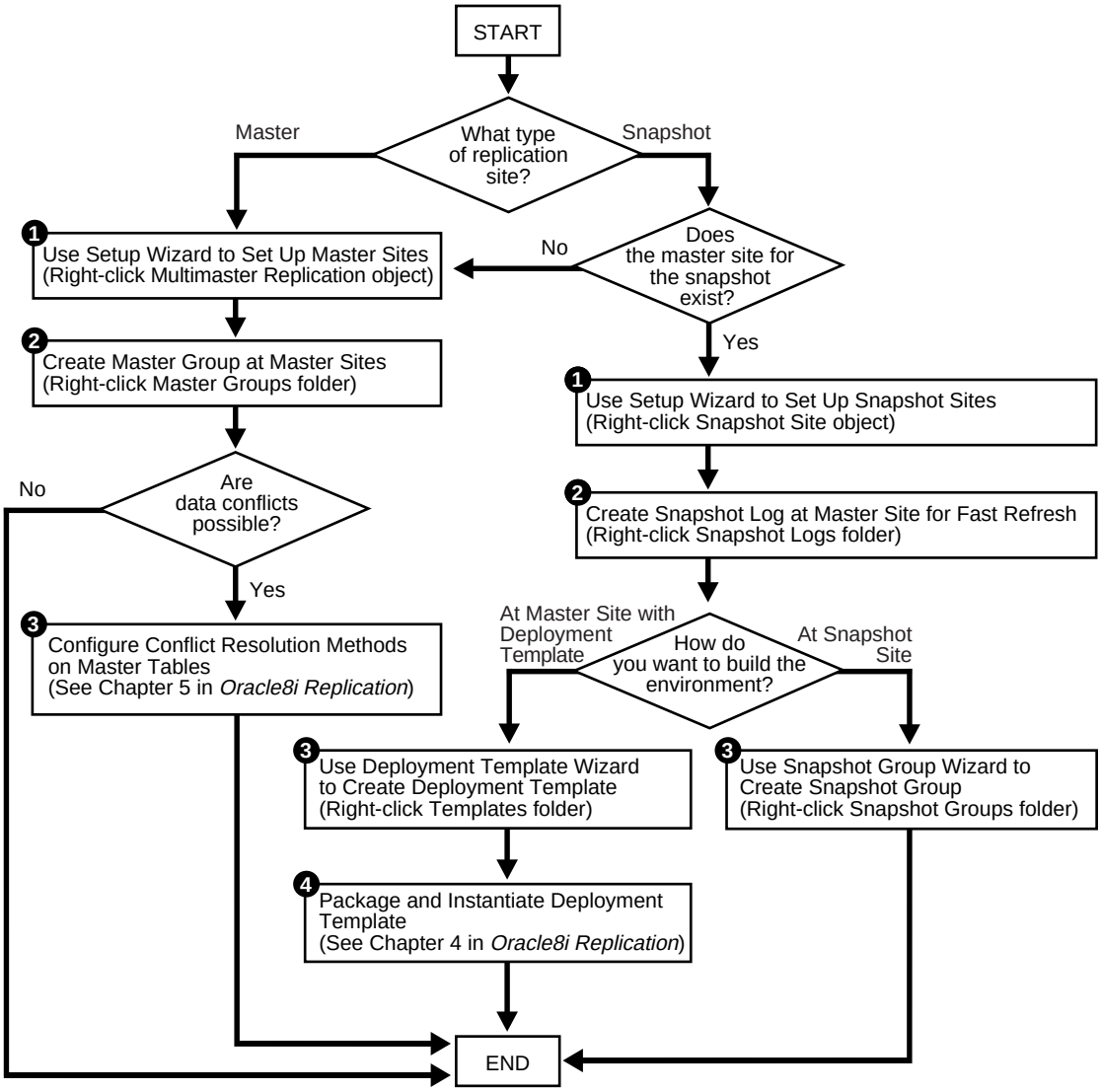


See Also: The "Copying a Template" topic in the Replication Management tool online help for detailed information about using the Copy Template Wizard to copy a deployment template. To access this topic in the online help, open Replication > Deployment Templates > Manage in the Help Contents.

Flowchart for Creating a Replicated Environment

The flowchart in Figure 9–12 displays the major decisions and tasks that are involved when you create a replication environment using the Replication Management tool. The flowchart shows the major decisions and steps for building both multimaster and snapshot environments. Each task in the flowchart includes a cross reference to a section in this chapter or in the release 8.1.6 *Oracle8i Replication* book that provides more information about the task. Detailed instructions about completing these tasks are in the Replication Management tool online help.

Figure 9–12 Create Replicated Environment Using the Replication Management Tool



10

Replication Management API Reference

This chapter is an addendum to the release 8.1.6 *Oracle8i Replication Management API Reference* and describes changes in the replication management API in release 8.1.7. This chapter also includes documentation additions.

This chapter contains these topics:

- Changes to the DBMS_REPCAT_INSTANTIATE Package
- Changes to the DBMS_REPCAT_RGT Package
- Documentation Additions

Changes to the DBMS_REPCAT_INSTANTIATE Package

The following section describes changes to the `DBMS_REPCAT_INSTANTIATE` package in release 8.1.7.

The offline_dirpath Parameter Added to INSTANTIATE_OFFLINE_REPAPI

The `DBMS_REPCAT_INSTANTIATE.INSTANTIATE_OFFLINE_REPAPI` function includes the new required parameter `offline_dirpath`. The following section includes documentation for the new parameter and replaces the description of this function in the release 8.1.6 *Oracle8i Replication Management API Reference*.

Updated Description of the INSTANTIATE_OFFLINE_REPAPI Function

The `DBMS_REPCAT_INSTANTIATE.INSTANTIATE_OFFLINE_REPAPI` function generates a file at the master site that is used to create the snapshot environment at a remote RepAPI snapshot site while offline. This offline instantiation file should be used at remote RepAPI sites that are not able to remain connected to the master site for an extended amount of time.

Offline instantiation is an ideal solution when the remote snapshot site has limited network connectivity. The generated file can be posted on an FTP site or loaded to a CD-ROM, floppy disk, and so on.

The file generated by this function is stored at the master site in the directory specified by the parameter `offline_dirpath`. The file is named based on the username of the connected user and the `refresh_template_name` and `site_id` parameters. The file is identified with the file type extension `.oli`. For example, an offline instantiation for the user `scott` of the template named `mytemplate` at site `1234` is named the following:

```
scott_mytemplate_1234.oli
```

This is a public function to generate an offline instantiation file for the connected user.

Note: This function is used in performing an offline instantiation of a deployment template.

This function should not be confused with the procedures in the `DBMS_OFFLINE_OG` package (used for performing an offline instantiation of a master table) nor with the procedures in the `DBMS_OFFLINE_SNAPSHOT` package (used for performing an offline instantiation of a snapshot). See these respective packages for more information on their use.

Syntax

```
DBMS_REPCAT_INSTANTIATE.INSTANTIATE_OFFLINE_REPAPI(
  refresh_template_name   IN   VARCHAR2,
  site_id                 IN   VARCHAR2   := NULL,
  master                  IN   VARCHAR2   := NULL,
  url                     IN   VARCHAR2   := NULL,
  ssl                     IN   NUMBER     := 0,
  offline_dirpath         IN   VARCHAR2   := NULL,
  trace_vector            IN   NUMBER     := DBMS_REPCAT_RGT.NO_TRACE_DUMP,
  resultset_threshold     IN   NUMBER     := DBMS_REPCAT_INSTANTIATE.
                                             RESULTSET_THRESHOLD,
  lob_threshold           IN   NUMBER     := DBMS_REPCAT_INSTANTIATE.
                                             LOB_THRESHOLD);
```

Table 10–1 INSTANTIATE_OFFLINE_REPAPI Function Parameters **(Page 1 of 2)**

Parameter	Description
`refresh_template_name`	The name of the deployment template to be instantiated.
`site_id`	A temporary name assigned to this offline instantiation. This temporary name is part of the offline instantiation file name. If the default `NULL` value is used, Oracle generates a number for this temporary name. It may be useful to specify a name if you want the offline instantiation file name to indicate its intended snapshot site. The `site_id` must be unique. That is, no two offline instantiation files can have the same `site_id`. This temporary name is always overwritten by the RepAPI client when the deployment template is instantiated. Oracle Corporation recommends that you use the default `NULL` value for this parameter.
`master`	An optional alias used for the server by the RepAPI client. If specified, then the RepAPI client must always refer to the server by this alias.

Table 10–1 INSTANTIATE_OFFLINE_REPAPI Function Parameters **(Page 2 of 2)**

Parameter	Description
`url`	The published URL at the master site for access to the database. If specified, then the RepAPI client must always refer to the server by this URL. The following is an example of a URL: `sess_iiop://myserver:2481:RNDM/etc/repapi` This URL has the following parts: ■ `myserver` is the host name. ■ `2481` is the Internet Inter-ORB Protocol (IIOP) listener port number. ■ `RNDM` is the SID for the database. ■ `/etc/repapi` is the published name for the object.
`ssl`	1 indicates that the snapshots use secure sockets layer (SSL) to communicate with the master site. 0 indicates that SSL is not used. **Note:** Use of SSL is not supported in this release of Oracle.
`offline_dirpath`	The directory where the offline instantiation file is saved. If `offline_dirpath` is omitted, then the file is saved to the path specified by the `UTL_FILE_DIR` initialization parameter. If neither `offline_dirpath` nor `UTL_FILE_DIR` are specified, then an error is raised.
`trace_vector`	The trace level for debugging.
`resultset_threshold`	The maximum size of non-LOB row data sent during the snapshot refresh process.
`lob_threshold`	The maximum size of LOB row data sent during the snapshot refresh process.

Table 10–2 INSTANTIATE_OFFLINE_REPAPI Function Exceptions

Exception	Description
`miss_refresh_template`	The template does not exist.
`miss_user`	The username does not exist in the database.
`miss_template_site`	The template has not been instantiated for the user and site.

Returns

Table 10–3 INSTANTIATE_OFFLINE_REPAPI Function Returns

Return Value	Description
`0`	An error was encountered.
`1`	No errors were encountered.

Changes to the DBMS_REPCAT_RGT Package

The following section describes changes to the `DBMS_REPCAT_RGT` package in release 8.1.7.

The offline_dirpath Parameter Required in INSTANTIATE_OFFLINE_REPAPI

The `offline_dirpath` parameter is required in the `DBMS_REPCAT_RGT.INSTANTIATE_OFFLINE_REPAPI` function in release 8.1.7. The following section includes documentation for this parameter and replaces the description of this function in the release 8.1.6 *Oracle8i Replication Management API Reference*.

Note: Due to a documentation error, the `offline_dirpath` parameter is not included in the description for this function in the release 8.1.6 *Oracle8i Replication Management API Reference*. However, the parameter was included in this function in release 8.1.6. The parameter was optional in release 8.1.6 but is required in release 8.1.7.

Updated Description of the INSTANTIATE_OFFLINE_REPAPI Function

The `DBMS_REPCAT_RGT.INSTANTIATE_OFFLINE_REPAPI` function generates a file at the master site that is used to create the snapshot environment at a remote RepAPI snapshot site while offline. This offline instantiation file should be used at remote RepAPI sites that are not able to remain connected to the master site for an extended amount of time.

Offline instantiation is an ideal solution when the remote snapshot site has limited network connectivity. The generated file can be posted on an FTP site or loaded to a CD-ROM, floppy disk, and so on.

The file generated by this function is stored at the master site in the directory specified by the parameter `offline_dirpath`. The file is named based on the `user_name`, `refresh_template_name`, and `site_id` parameters. The file is identified with the file type extension `.oli`. For example, an offline instantiation for the user `scott` of the template named `mytemplate` at site `1234` is named the following:

```
scott_mytemplate_1234.oli
```

Note: This function is used in performing an offline instantiation of a deployment template. Additionally, this function is for replication administrators that are instantiating for another user. Users wanting to perform their own instantiation should use the public version of this function: DBMS_REPCAT_INSTANTIATE.INSTANTIATE_OFFLINE_REPAPI. See the "Updated Description of the INSTANTIATE_OFFLINE_REPAPI Function" on page 10-2 for information.

This function should not be confused with the procedures in the DBMS_OFFLINE_OG package (used for performing an offline instantiation of a master table) nor with the procedures in the DBMS_OFFLINE_SNAPSHOT package (used for performing an offline instantiation of a snapshot). See these respective packages for more information on their use.

Syntax

```
DBMS_REPCAT_RGT.INSTANTIATE_OFFLINE_REPAPI(
  refresh_template_name   IN   VARCHAR2,
  site_id                 IN   VARCHAR2,
  user_name               IN   VARCHAR2   := USER,
  master                  IN   VARCHAR2   := NULL,
  url                     IN   VARCHAR2   := NULL,
  ssl                     IN   NUMBER     := 0,
  offline_dirpath         IN   VARCHAR2   := NULL,
  trace_vector            IN   NUMBER     := DBMS_REPCAT_RGT.NO_TRACE_DUMP,
  resultset_threshold     IN   NUMBER     := DBMS_REPCAT_INSTANTIATE.
                                             RESULTSET_THRESHOLD,
  lob_threshold           IN   NUMBER     := DBMS_REPCAT_INSTANTIATE.
                                             LOB_THRESHOLD);
```

Table 10–4 INSTANTIATE_OFFLINE_REPAPI Function Parameters **(Page 1 of 2)**

Parameter	Description
`refresh_template_name`	The name of the deployment template to be instantiated.
`site_id`	A temporary name assigned to this offline instantiation. This temporary name is part of the offline instantiation file name. If the default `NULL` value is used, Oracle generates a number for this temporary name. It may be useful to specify a name if you want the offline instantiation file name to indicate its intended snapshot site. The `site_id` must be unique. That is, no two offline instantiation files can have the same `site_id`. This temporary name is always overwritten by the RepAPI client when the deployment template is instantiated. Oracle Corporation recommends that you use the default `NULL` value for this parameter.
`user_name`	The name of the user for whom the instantiation file is being generated.
`master`	An optional alias used for the server by the RepAPI client. If specified, then the RepAPI client must always refer to the server by this alias.
`url`	The published URL at the master site for access to the database. If specified, then the RepAPI client must always refer to the server by this URL. The following is an example of a URL: `sess_iiop://myserver:2481:RNDM/etc/repapi` This URL has the following parts: ■ `myserver` is the host name. ■ `2481` is the Internet Inter-ORB Protocol (IIOP) listener port number. ■ `RNDM` is the SID for the database. ■ `/etc/repapi` is the published name for the object.
`ssl`	1 indicates that the snapshots use secure sockets layer (SSL) to communicate with the master site. 0 indicates that SSL is not used. **Note:** Use of SSL is not supported in this release of Oracle.

Table 10–4 INSTANTIATE_OFFLINE_REPAPI Function Parameters **(Page 2 of 2)**

Parameter	Description
offline_dirpath	The directory where the offline instantiation file is saved. If offline_dirpath is omitted, then the file is saved to the path specified by the UTL_FILE_DIR initialization parameter. If neither offline_dirpath nor UTL_FILE_DIR are specified, then an error is raised.
trace_vector	The trace level for debugging.
resultset_threshold	The maximum size of non-LOB row data sent during the snapshot refresh process.
lob_threshold	The maximum size of LOB row data sent during the snapshot refresh process.

Table 10–5 INSTANTIATE_OFFLINE_REPAPI Function Exceptions

Exception	Description
miss_refresh_template	The template does not exist.
miss_user	The username does not exist in the database.
miss_template_site	The template has not been instantiated for the user and site.

Returns

Table 10–6 INSTANTIATE_OFFLINE_REPAPI Function Returns

Return Value	Description
0	An error was encountered.
1	No errors were encountered.

Documentation Additions

This section provides additional documentation about the replication management API that was not included in the release 8.1.6 *Oracle8i Replication Management API Reference*.

Switching Master Sites that are Running Different Releases of Oracle

The following describes the required actions if you switch the master site for a snapshot site to a different master site that is running a different release of Oracle. This additional documentation is added to the description for the `DBMS_REPCAT.SWITCH_SNAPSHOT_MASTER` procedure.

If `min_communication` is `true` for the snapshot and the new master is an Oracle7 master, regenerate replication support for the snapshot with `min_communication` set to `false`.

If `generate_80_compatible` is `false` for the snapshot and the new master is a release lower than Oracle8*i* (Oracle7 or Oracle8), regenerate replication support for the snapshot with `generate_80_compatible` set to `true`.

You can set both parameters for a snapshot in one call to `DBMS_REPCAT.GENERATE_SNAPSHOT_SUPPORT`.

11

Oracle Label Security Error Messages

ORA-12400 invalid argument to facility error handling

Cause: An argument to a facility error handling function exceeded a maximum limit or referred to an invalid product/facility.

Action: Specify a valid facility error handling parameter value.

ORA-12401 invalid label string: *string*

Cause: The policy could not convert the label string to a valid internal label.

Action: Correct the syntax of the label string.

ORA-12402 invalid format string: *string*

Cause: The format string is not supported by the policy.

Action: Correct the syntax of the format string.

ORA-12403 invalid internal label

Cause: An internal label could not be converted to a valid label for the policy.

Action: Analyze any additional messages on the error stack and consult the policy documentation.

ORA-12404 invalid privilege string: *string*

Cause: The policy could not interpret the privilege string.

Action: Specify a privilege string that is supported by the policy.

ORA-12405 invalid label list

Cause: The policy determined that the label list was invalid for its intended use.

Action: Check the policy constraints on the specific list of labels.

ORA-12406 unauthorized SQL statement for policy *string*

Cause: The policy did not authorize the database session to perform the requested SQL statement.

Action: Grant the user or program unit the necessary policy privilege or additional authorizations.

ORA-12407 unauthorized operation for policy *string*

Cause: The policy did not authorize the database session to perform the requested operation.

Action: Grant the user or program unit the necessary policy privilege or additional authorizations.

ORA-12408 unsupported operation: *string*

Cause: The specified policy does not support the requested operation.

Action: Consult the policy documentation to determine the supported access mediation operations.

ORA-12409 policy startup failure for *string* policy

Cause: The policy encountered an error during startup processing; access to the data protected by the policy is prohibited.

Action: Check the alert log for additional information, correct the policy error, and restart the instance.

ORA-12410 internal policy error for policy: *string* Error: *string*

Cause: The policy enforcement encountered an internal error.

Action: Consult the policy documentation for details.

ORA-12411 invalid label value

Cause: The specified label value does not exist.

Action: Check the data dictionary views for the policy to identify valid labels.

ORA-12412 policy package *string* is not installed

Cause: The policy package does not exist in the database.

Action: Check that the policy package name is correct or install the required policy package.

ORA-12413 labels do not belong to the same policy

Cause: The labels being compared belong to different policies.

Action: Only compare labels that belong to the same policy.

ORA-12414 internal LBAC error: *string* Error: *string*

Cause: An internal label policy framework error occurred.

Action: Contact Oracle Support Services.

ORA-12415 A column of another datatype exists on the specified table

Cause: The datatype of the column present in the table is different from the datatype set for the policy column.

Action: Drop the column on the table or change the datatype for policy column.

ORA-12416 policy *string* not found

Cause: The specified policy does not exist in the database.

Action: Enter the correct policy name or create the policy.

ORA-12417 database object "*string*" not found

Cause: The specified object was not in the database.

Action: Enter the correct name for the database object.

ORA-12418 user *string* not found

Cause: The specified user does not exist in the database.

Action: Correct the user name or create the user.

ORA-12419 null binary label value

Cause: A null value was provided for a binary label operation.

Action: Provide a valid binary label for the operation.

ORA-12420 required procedures and functions not in policy package "*string*"

Cause: The policy package did not contain all of the procedures and functions necessary to enforce the policy.

Action: Consult the label framework documentation for a list of required procedures and functions for a policy package.

ORA-12421 different size binary labels

Cause: The label sizes for the binary label operation were not equal.

Action: Provide binary labels with the same lengths for the operation.

ORA-12422 max policies exceeded

Cause: You tried to create a new policy, but the maximum number of policies for the instance had already been created.

Action: Increase the size of the MAX_LABEL_POLICIES initialization parameter and restart the server.

ORA-12423 invalid position specified

Cause: The position specified for a binary label operation was invalid.

Action: Provide a position that is within the label size limits.

ORA-12424 length exceeds binary label size

Cause: The length specified for a binary label operation exceeded the size of the binary label.

Action: Provide a bit or byte length that is within the label size limits.

ORA-12425 cannot apply policies or set authorizations for system schemas

Cause: You tried to either apply a policy to the SYS, SYSTEM, or LBACSYS schema or to set user labels/privileges for the SYS, SYSTEM, or LBACSYS user.

Action: Apply policies and set authorizations only for non-system users.

ORA-12426 invalid audit option

Cause: The option specified was not a valid audit option for the specified policy.

Action: Enter a correct audit option.

ORA-12427 invalid input value for *string* parameter

Cause: An input parameter was specified incorrectly.

Action: Correct the parameter value.

ORA-12429 label list range exceeded

Cause: The specified index value was not between 1 and 6.

Action: Correct the index value for the label list operation.

ORA-12430 invalid privilege number

Cause: The specified privilege number was not between 1 and 32.

Action: Correct the privilege number.

ORA-12431 invalid audit action

Cause: The specified audit action was not a valid audit action.

Action: Correct the audit action number.

ORA-12432 LBAC error: *string*

Cause: LBAC enforcement resulted in an error.

Action: Correct the problem identified in the error message.

ORA-12433 create trigger failed, policy not applied

Cause: The policy could not be applied due to errors during the creation of a DML trigger.

Action: Correct the SQL syntax of the label function specification.

ORA-12434 invalid audit type: *string*

Cause: The audit type must be BY ACCESS or BY SESSION.

Action: Correct the audit type value.

ORA-12435 invalid audit success: *string*

Cause: The audit success parameter must be SUCCESSFUL or NOT SUCCESSFUL.

Action: Correct the audit success value.

ORA-12436 no policy options specified

Cause: A NULL option string was specified, but no default schema or policy option string was found.

Action: Enter a valid option string, or alter the schema or policy to have a valid default option string.

ORA-12437 invalid policy option: *string*

Cause: A value that was not a valid policy option was entered.

Action: Correct the policy option value.

ORA-12438 repeated policy option: *string*

Cause: A policy option was entered more than once in the option string.

Action: Remove the duplicate policy option value.

ORA-12439 invalid combination of policy options

Cause: A set of contradictory policy options was entered.

Action: Provide a set of compatible policy options.

ORA-12440 insufficient authorization for the SYSDBA package

Cause: The use of the SYSDBA package requires the LBAC_DBA role.

Action: Grant the LBAC_DBA role to the database user.

ORA-12441 policy *string* already exists

Cause: You tried to create a policy with the same name as an existing one.

Action: Use a different name or drop the existing policy.

ORA-12442 policy column "*string*" already used by an existing policy

Cause: You tried to create a policy with the same policy column name as an existing policy.

Action: Use a different name for the policy column or drop the existing policy.

ORA-12443 policy not applied to some tables in schema

Cause: You applied a policy to a schema, and some of the tables in the schema already had the policy applied.

Action: No action necessary; the policy was applied to the remaining tables.

ORA-12444 policy already applied to table

Cause: You tried to apply a policy to a table that was already protected by the policy.

Action: To change the policy options, predicate, or label function, remove the policy from the table and re-apply it.

ORA-12445 cannot change HIDDEN property of column

Cause: You tried to specify a different HIDE option for a table with an existing policy column.

Action: Drop the column from the table and reapply the policy with the new HIDE option.

ORA-12446 Insufficient authorization for administration of policy *string*

Cause: You tried to perform an administrative function for a policy, but you have not been granted the *policy*_DBA role.

Action: Grant the user the *policy*_DBA role for the specified policy.

ORA-12447 policy role already exists for policy *string*

Cause: The role named *policy*_DBA already exists.

Action: Correct the policy name or delete the existing policy.

ORA-12448 policy *string* not applied to schema *string*

Cause: You tried to alter a schema policy that was not applied.

Action: Correct the policy name or schema name.

ORA-12449 Labels specified for user must be of type USER

Cause: You tried to set labels for a user, but the labels in the list were not all designated as USER labels.

Action: Alter the labels to be USER labels.

ORA-12450 LOB datatype disabled in LBAC initialization file

Cause: You tried to specify a LOB datatype for a column or attribute, but the use of the LOB datatype has been disabled.

Action: Change the LBAC initialization file to allow the creation of LOB columns and attributes.

ORA-12451 label not designated as USER or DATA

Cause: A label is either a DATA label, a USER label, or both DATA and USER.

Action: Enter TRUE for at least DATA or USER.

ORA-12452 label tag *string* already exists

Cause: The label tag value you entered is already in use for another label.

Action: Enter a different value for the label tag.

ORA-12453 label *string* already exists

Cause: The label value you entered already exists.

Action: No action necessary; alter the label to change its tag or type.

ORA-12454 label *string* does not exist for policy *string*

Cause: The label tag or value you entered did not identify a label for the policy.

Action: Enter a label value or tag that is in use by the policy.

ORA-12461 undefined level *string* for policy *string*

Cause: The specified level is not defined for the policy.

Action: Correct the level identifier value.

ORA-12462 undefined compartment *string* for policy *string*

Cause: The specified compartment is not defined for the policy.

Action: Correct the compartment identifier value.

ORA-12463 undefined group *string* for policy *string*

Cause: The specified group is not defined for the policy.

Action: Correct the group identifier value.

ORA-12464 invalid characters in label component *string*

Cause: Label components can contain only alphanumeric characters, blanks, and underscores.

Action: Correct syntax of the label component.

ORA-12465 Not authorized for write on specified groups or compartments

Cause: You included groups or compartments that are not in the user's list of groups and compartments authorized for write access.

Action: Enter only groups and compartments that are authorized for write.

ORA-12466 default level is greater than the user's maximum

Cause: The default level cannot be greater than the user's maximum.

Action: Enter an authorized level.

ORA-12467 minimum label can contain a level only

Cause: You included compartments or groups in the minimum label.

Action: Enter only an authorized minimum level as the label.

ORA-12468 max write level does not equal max read level

Cause: The level in the max write label must equal the level in the max read label.

Action: Enter max read and max write labels with the same level component.

ORA-12469 no user levels found for user *string* and policy *string*

Cause: No levels have been specified for the user.

Action: Enter the maximum and minimum labels for the user.

ORA-12470 NULL or invalid user label: *string*

Cause: The label entered is NULL or not within the user's authorizations.

Action: Enter the authorized labels for the user.

12

Standby Database

This chapter describes changes to the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual. This chapter contains the following sections:

- Revised Script to Identify the Logs in a Gap Sequence
- Additions to Compatibility and Operational Requirements
- Changes to Enabling Online Changes to the Initialization Parameter Settings
- Changes to Receiving Archived Redo Logs While in Read-Only Mode
- Changes to Creating the Standby Database Files
- Revised Scenario 1: Creating a Standby Database on the Same Host
- Revised Scenario 2: Creating a Standby Database on a Remote Host

Revised Script to Identify the Logs in a Gap Sequence

The 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual documents a script for identifying the logs in a gap sequence. This script, which is referenced throughout the manual, has been updated for release 8.1.7. The updated script, shown in Figure 12–1, supersedes the previous version of the script. The updated script handles the case where one or more of the datafiles are offline. The previous version of the script gave incorrect results when one or more of the datafiles were offline.

Example 12–1 Updated Script to Identify the Logs in a Gap Sequence

```
SELECT   high.thread#, "LowGap#", "HighGap#"
FROM
 (SELECT   thread#, MIN(sequence#)-1 "HighGap#"
  FROM
        (SELECT  a.thread#, a.sequence#
         FROM
                (SELECT * FROM V$ARCHIVED_LOG) a,
                (SELECT thread#, MAX(next_change#) gap1
                 FROM  V$LOG_HISTORY
                 GROUP BY thread# )  b
         WHERE
                a.thread# = b.thread#
                AND
                a.next_change# > gap1
        ) GROUP BY thread#
 ) high,
 (SELECT thread#, MIN(gap2) "LowGap#"
  FROM
        (SELECT thread#, sequence#+1 gap2
         FROM   V$LOG_HISTORY, V$DATAFILE
         WHERE
                checkpoint_change# <= next_change#
                  AND
                checkpoint_change# >= first_change#
                  AND
                enabled = 'READ WRITE'
        ) GROUP BY thread#
  ) low
WHERE
   low.thread# = high.thread#
     AND
   "LowGap#" < "HighGap#";
```

Additions to Compatibility and Operational Requirements

Chapter 1 in the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual has a section titled "Compatibility and Operational Requirements." The following section supersedes the section in the 8.1.6 documentation.

Compatibility and Operational Requirements

Note the following requirements for maintaining a standby database:

- The primary database must run in ARCHIVELOG mode.
- A standby database in manual recovery mode operates only on Oracle release 7.3 or higher.
- A standby database in managed recovery mode operates only on Oracle release 8.1 or higher.
- A standby database in read-only mode operates only on Oracle release 8.1.5 or higher.
- The redo logs that you apply to the standby database must be either archived or noncurrent online redo logs. Note that you can salvage the transactions in the current redo log by archiving it manually.
- You must use the same version and release of the operating system on the primary and standby hosts. The standby host can, however, use a different directory structure.
- The hardware architecture on the primary and standby hosts must be the same.
- You should use the same version, release, and patch of the Oracle RDBMS for the primary and standby databases so that failover operations are not compromised.
- The primary database and standby database cannot share the same control file.
- If you place your primary and standby databases on the same host, some operating systems will not allow you to mount two instances with the same database name on the same machine simultaneously. Workarounds for this situation exist for every platform.
- You cannot activate a standby database and then return it to managed recovery mode; an activated standby database becomes a normal primary database.
- You can use standby database datafiles to recover a primary database only on Oracle release 8.0.4 or higher.

Changes to Enabling Online Changes to the Initialization Parameter Settings

Chapter 2 in the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual has a section on page 2-32 titled, "Enabling Changes to the Initialization Parameter Settings." The following section supersedes the section in the 8.1.6 documentation.

Enabling Online Changes to the Initialization Parameter Settings

If you configured the primary initialization parameter file to archive to the standby site, you should enable these new parameter settings *after* starting the standby instance and the listener on the standby site.

You can make changes to the LOG_ARCHIVE_DEST parameters in the primary database initialization parameter file while the database is open, but the changes only take effect when the instance is restarted. If the database is open and you want to avoid restarting it, enable the parameter changes dynamically using ALTER SYSTEM statements.

For example, assume that you made the following changes to the initialization parameter file while the database was open:

```
LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest/ MANDATORY REOPEN=2";
LOG_ARCHIVE_DEST_2="SERVICE=stby1 MANDATORY REOPEN=2";
LOG_ARCHIVE_DEST_STATE_1=ENABLE;
LOG_ARCHIVE_DEST_STATE_2=ENABLE;
LOG_ARCHIVE_MIN_SUCCEED_DEST=2;
```

You can then connect to the primary database using SQL*Plus and issue ALTER SYSTEM statements as follows to enable these settings:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest/ MANDATORY REOPEN=2";
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2="SERVICE=stby1 MANDATORY REOPEN=2";
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE;
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST=2;
```

Once you have enabled these changes, the primary database can start attempting to archive redo logs to a standby site. Once an archiving attempt has succeeded, archived redo logs continue to transfer from the primary site to the standby site unless the standby instance is shut down, the data communication link goes down, or a destination is manually disabled.

Changes to Receiving Archived Redo Logs While in Read-Only Mode

Chapter 3 in the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual has a section on page 3-18 titled, "Receiving Archived Redo Logs While in Read-Only Mode." The following section supersedes the section in the 8.1.6 documentation.

Receiving Archived Redo Logs While in Read-Only Mode

While the standby database is in read-only mode, the site can still receive archived redo logs from the primary site. Nevertheless, Oracle does not apply these logs automatically, as in managed recovery. Consequently, a read-only standby database is not synchronized with the primary database at the archive level. You should not activate the standby database in a failover situation unless all archived redo logs have been applied.

> **See Also:** Enabling Changes to the Initialization Parameter Settings on page 2-33 of the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual for examples of initialization parameter settings you need to define to automatically archive from the primary site to the standby site

Changes to Creating the Standby Database Files

Chapter 2 in the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual has a section on page 2-7 titled, "Creating the Standby Database Files." The following section supersedes the section in the 8.1.6 documentation.

Creating the Standby Database Files

You can create a standby database on the same host as your primary database or on a separate host. If you create your standby database on the same host, follow the creation procedure carefully when creating the standby database files so that you do not overwrite files on the primary database.

The creation of the standby database files occurs in three stages:

1. Creating the Standby Datafiles
2. Creating the Standby Control File
3. Transferring Files to the Standby Site

Creating the Standby Datafiles

First, make backups of your primary database datafiles. You create the standby datafiles from these backups.

You can use any backup of the primary database so long as you have archived redo logs to completely recover the database. The backup can be old or new, consistent or inconsistent. Hot backups have the advantage of allowing you to keep the database open while performing the backup. Nevertheless, you may prefer to make a new closed, consistent backup to prevent the application of a large number of archived redo logs.

To make a consistent, whole database backup to serve as the basis for the standby database:

1. Start a SQL*Plus session on your primary database and query the V$DATAFILE view to obtain a list of the primary datafiles. For example, enter:

```
SQL> SELECT name FROM v$datafile;
NAME
--------------------------------------------------------------------------------
/oracle/dbs/tbs_01.f
/oracle/dbs/tbs_02.f
/oracle/dbs/tbs_03.f
/oracle/dbs2/tbs_11.f
/oracle/dbs2/tbs_12.f
/oracle/dbs3/tbs_21.f
/oracle/dbs3/tbs_22.f
7 rows selected.
```

2. Shut down the primary database cleanly:

```
SQL> SHUTDOWN IMMEDIATE;
```

3. Make a consistent backup of the datafiles from your primary database using an operating system utility. For example, to copy all of the datafiles into the `/backup` temporary directory, enter:

```
% cp /oracle/dbs/*.f /backup
% cp /oracle/dbs2/*.f /backup
% cp /oracle/dbs3/*.f /backup
```

4. Start and mount the primary database without opening it. For example, enter:

```
SQL> STARTUP MOUNT pfile=initPROD1.ora;
```

See Also: The *Oracle8i Backup and Recovery Guide*, to learn how to make operating system backups

Creating the Standby Control File

After you have created the backups that will be used as the standby datafiles, you can create the standby database control file. The control file must have been created at a time later than the latest timestamp for the backup datafiles.

Note: You cannot use a single control file for both the primary and standby databases. The standby instance is independent from the primary instance and so requires exclusive possession of its database files.

To create the standby database control file:

1. Connect to the primary database and create the control file for your standby database. For example, if you want to create the standby control file as `/oracle/dbs/stbycf.f` on the primary site, enter the following:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/oracle/dbs/stbycf.f';
   ```

   Note that the filename for the created standby control file *must* be different from the filename of the current control file of the primary database.

   See Also: For ALTER DATABASE syntax, see the *Oracle8i SQL Reference*.

2. Ensure that the primary database is in ARCHIVELOG mode and that archiving is enabled. Either issue the ARCHIVE LOG LIST statement or query the V$DATABASE view:

   ```
   SQL> CONNECT sys/change_on_install@prod1 AS SYSDBA
   SQL> ARCHIVE LOG LIST
   Database log mode              Archive Mode
   Automatic archival             Enabled
   Archive destination            /vobs/oracle/work/arc_dest/arc
   Oldest online log sequence     821
   Next log sequence to archive   822
   Current log sequence           822
   ```

   If the output from the ARCHIVE LOG LIST statement displays "No Archive Mode," set the log archive mode as follows:

```
SQL> ALTER DATABASE ARCHIVELOG;
```

3. Open the primary database as follows:

```
SQL> ALTER DATABASE OPEN;
```

Transferring Files to the Standby Site

After you have successfully created the standby datafiles and control file, transfer the files to the standby site using an operating system utility. For example, if the standby site and primary site are on the same host, you can use the UNIX **cp** command to transfer files; if they are on separate hosts, you can use **ftp.**

If the standby database is on	Then you
A separate host with the same directory structure as the primary database	Can use the same path names for the standby files as the primary files. In this way, you do not have to rename the primary datafiles in the standby control file.
The same host as the primary database, or the standby database is on a separate host with a different directory structure	Must rename the primary datafiles in the standby control file after copying them to the standby site. You can: ■ Set the filename conversion initialization parameters (see Renaming Primary Filenames in the Standby Control File on page 2-22 of the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual). ■ Rename the files manually using ALTER DATABASE statements (see Manually Renaming Standby Files Not Captured by Conversion Parameters on page 2-32 of the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual). ■ Use a combination of conversion parameters and manual renames.

To transfer datafiles and the control file to the standby site:

Transfer the created control file and datafile backups to the standby site using operating system commands or utilities. Use an appropriate method for transferring binary files.

1. Transfer the control file first, because this transfer takes the least time. For example, enter the following:

```
% cp /backup/db.cf /standby/oracle/dbs/db.cf
```

2. Transfer the backup datafiles. For example, enter:

```
% cp /backup/*.df /standby/oracle/dbs
```

3. Transfer all available archived redo logs to the standby site. For example, enter:

```
% cp /arc_dest/*.arc /standby/arc_dest
```

Revised Scenario 1: Creating a Standby Database on the Same Host

Chapter 5 in the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual has a section on page 5-2 titled, "Scenario 1: Creating a Standby Database on the Same Host." The following section supersedes the section in the 8.1.6 documentation.

Scenario 1: Creating a Standby Database on the Same Host

This scenario describes the creation of a standby database STANDBY1 on the same host as the primary database PROD1. The host is a UNIX machine with three file systems, each mounted on a separate disk configuration on a different controller. By placing the standby database on a different file system from the primary database, you protect the primary database from a hard disk failure. By running the same-host standby database in managed recovery mode, you can keep it continuously up-to-date.

After you set up the standby database on the local host, you plan to create a standby database on a remote host for total disaster protection. In this way, even if all disks of the primary database crash or are destroyed in a disaster, you can fail over to the remote standby database and keep the database open.

Step 1: Plan the Standby Database.

Because the host uses three file systems, each on its own set of disks with its own controller, you decide to maintain the primary database files on the first file system, the standby database files on the second file system, and the ORACLE_HOME binaries on the third file system. If the primary database disks fail, you can switch to the standby database; if the ORACLE_HOME disks fail, you can switch to the remote standby database.

To host the standby database on the same machine as the primary database, you must set the following parameters in the standby database initialization parameter file:

- CONTROL_FILES
- LOCK_NAME_SPACE

- DB_FILE_NAME_CONVERT
- LOG_FILE_NAME_CONVERT

Fortunately, most (but not all) of the primary database datafiles and redo log files are in the same directory and are named consistently. You will have to rename some of the files manually using ALTER DATABASE statements.

Because the primary database is shut down every Sunday for an hour for maintenance, you decide to use that time to make a cold, consistent backup. You can then restart the database while you make the necessary configurations for the standby database.

Step 2: Create the Standby Database.

The next step in the procedure is to create the backup that will form the basis for the standby database. You know that you can use either an inconsistent or consistent backup, but because the database must go down every Sunday for maintenance, you decide to make a consistent backup then and use it for the standby database.

1. Determine the database files.

   On Sunday, before shutting down the primary database, you query the database to determine which datafiles it contains:

```
SQL> SELECT name FROM v$datafile;
NAME
--------------------------------------------------------------------------------
/fs1/dbs/tbs_01.f
/fs1/dbs/tbs_02.f
/fs1/dbs/tbs_11.f
/fs1/dbs/tbs_12.f
/fs1/dbs/tbs_21.f
/fs1/dbs/tbs_22.f
/fs1/dbs/tbs_13.f
/fs1/dbs/tbs_23.f
/fs1/dbs/tbs_24.f
/fs1/dbs/tbs_31.f
/fs1/dbs/tbs_32.f
/fs1/dbs/tbs_41.f
/fs1/dbs2/tbs_42.f
/fs1/dbs2/tbs_51.f
/fs1/dbs2/tbs_52.f
/fs1/dbs2/tbs_03.f
/fs1/dbs3/tbs_14.f
/fs1/dbs3/tbs_25.f
/fs1/dbs3/tbs_33.f
/fs1/dbs3/tbs_43.f
/fs1/dbs3/tbs_53.f
```

```
21 rows selected.
```

2. Back up the datafiles.

   After determining which datafiles are in the database, you shut down the database with the IMMEDIATE option:

   ```
   SQL> SHUTDOWN IMMEDIATE;
   ```

   At this point, you decide to back up all of the primary datafiles to a temporary directory as follows:

   ```
   % cp /fs1/dbs/* /fs1/temp
   % cp /fs1/dbs2/* /fs1/temp
   % cp /fs1/dbs3/* /fs1/temp
   ```

   You perform some other routine maintenance operations and then start and mount the database without opening it as follows:

   ```
   SQL> STARTUP MOUNT PFILE=initPROD1.ora;
   ```

3. Create the standby database control file.

   After a few minutes, you create the standby database control file in the same directory in which you stored the consistent backup:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/fs1/temp/stbycf.f';
   ```

4. Ensure that the primary database is in ARCHIVELOG mode and that automatic archival is enabled. Issue the ARCHIVE LOG LIST statement:

   ```
   SQL> ARCHIVE LOG LIST
   ```

   If the output from the ARCHIVE LOG LIST statement displays "No Archive Mode," set the log archive mode as follows:

   ```
   SQL> ALTER DATABASE ARCHIVELOG;
   ```

5. Open the primary database as follows:

   ```
   SQL> ALTER DATABASE OPEN;
   ```

6. Transfer files to the standby file system.

   After you have successfully created the standby database control file, you can copy the datafiles and the standby database control file from the primary file system to the standby file system.

Because the transferring of datafiles can take a long time, you first copy the control file, begin copying the datafiles, and then proceed to other tasks (such as network configuration). For example, enter the following at the UNIX command shell:

```
% cp /fs1/temp/stbycf.f /fs2/dbs/cf1.f
% cp /fs1/temp/tbs* /fs2/dbs
```

Step 3: Configure the Network Files.

In order to run a standby database in a managed standby environment, you must configure a Net8 connection between the primary and standby databases so that you can archive the redo logs to the standby service.

You use the IPC protocol to connect the primary database to the standby database because both databases are on the same host. Because you do not maintain an Oracle Names server, you must create both a `tnsnames.ora` entry for the primary database and a `listener.ora` entry for the standby database.

1. Configure the `tnsnames.ora` file.

   Your next step is to open the `tnsnames.ora` file in a text editor:

   ```
   % vi /fs3/oracle/network/admin/tnsnames.ora
   ```

   Currently, only one service name entry exists in the file, a TCP/IP connection to the PROD1 database:

   ```
   prod1 = (DESCRIPTION=
           (ADDRESS=(PROTOCOL=tcp)(PORT=1521)(HOST=dlsun183))
           (CONNECT_DATA=(SID=prod1))
   )
   ```

   To define an IPC connection between the primary and the standby database, you add an entry with the following format:

   ```
   standby_service_name = (DESCRIPTION=
                              (ADDRESS=(PROTOCOL=ipc) (KEY=keyhandle))
                              (CONNECT_DATA=(SID=standby_sid)))
   ```

   Substitute appropriate values for standby_service_name, keyhandle, and standby_sid, as the following example shows:

   ```
   standby1 = (DESCRIPTION=
                 (ADDRESS=(PROTOCOL=ipc) (KEY=kstdby1))
                 (CONNECT_DATA=(SID=stdby1)))
   ```

2. Configure the `listener.ora` file.

Your next step is to open the `listener.ora` file, which is located on file system /fs3:

```
% vi /fs3/oracle/network/admin/listener.ora
```

You discover the following list of addresses (where on the host the listener is listening) and SIDs (which connections the listener is listening for):

```
LISTENER = (ADDRESS_LIST=
 (ADDRESS=(PROTOCOL=tcp)(PORT=1521)(HOST=dlsun183))
)
SID_LIST_LISTENER = (SID_LIST=
 (SID_DESC=(SID_NAME=PROD1)(ORACLE_HOME=/fs3/oracle))
)
```

Currently, the listener is listening on port 1521 of the host dlsun183 for database PROD1.

You need to edit the `listener.ora` file and add two entries with the following format:

```
STANDBY_LISTENER = (ADDRESS_LIST=(ADDRESS=(PROTOCOL=ipc)
                        (KEY=keyhandle)))

SID_LIST_STANDBY_LISTENER = (SID_LIST=
                  (SID_DESC=(SID_NAME=standby_sid)(ORACLE_HOME=/oracle_home)))
```

The `listener.ora` file is typically located in the $ORACLE_HOME/network/admin directory on the standby site. Substitute appropriate values for keyhandle, standby_sid, and oracle_home as the following example shows:

```
STBY1_LISTENER = (ADDRESS_LIST=(ADDRESS=(PROTOCOL=ipc)   # same node as primary
                   (KEY=kstdby1)))   # ORACLE_SID standby instance is started with

SID_LIST_STBY1_LISTENER = (SID_LIST=
                  (SID_DESC=(SID_NAME=stdby1)(ORACLE_HOME=/vobs/fs3/oracle)))
```

Now that you have edited the `listener.ora` file, you must start the listener:

```
% lsnrctl
LSNRCTL for Solaris: Version 8.1.5.0.0 - Production on 05-APR-99 11:39:41

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Welcome to LSNRCTL, type "help" for information.

LSNRCTL> start stby1_listener
```

As an alternative to the steps outlined in this section, you can use the Net8 Assistant graphical user interface to configure the network files. For additional information, see the *Net8 Administrator's Guide.*

Step 4: Configure the Primary Database Parameter File.

Now that you have configured the network files, you can edit the primary database initialization parameter file. The primary database is now up and running, so these changes will only be enabled if you restart the instance or issue ALTER SESSION or ALTER SYSTEM statements.

The only changes you need to make to the file involve archiving to the standby service. Currently, the primary database parameter file appears as follows:

```
db_name=prod1
control_files=(/fs1/dbs/cf1.f,/fs1/dbs/cf2.f)
compatible=8.1.6
log_archive_start = TRUE
log_archive_dest_1 = 'LOCATION=/fs1/arc_dest/ MANDATORY REOPEN=60'
log_archive_dest_state_1 = ENABLE
log_archive_format = log_%t_%s.arc
audit_trail=FALSE
o7_dictionary_accessibility=FALSE
global_names=FALSE
db_domain=regress.rdbms.dev.us.oracle.com
remote_login_passwordfile = exclusive

# default parameters for instance 1
processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
distributed_transactions=10
db_block_buffers=1000
db_files=200
shared_pool_size=10000000
```

1. Specify standby archive destinations.

   Currently, you archive to only one location: a local directory. Because you want to maintain the local standby database in managed recovery mode, you must specify a new archiving location using a service name.

   Open the primary database initialization parameter file with a text editor and examine the current archiving location and format:

   ```
   log_archive_dest_1 = 'LOCATION=/fs1/arc_dest/ MANDATORY REOPEN=60'
   ```

```
log_archive_dest_state_1 = ENABLE
log_archive_format = log_%t_%s.arc
```

Parameter/Option	Meaning
LOG_ARCHIVE_DEST_1	Indicates an archiving destination.
LOCATION	Indicates a local directory.
LOG_ARCHIVE_DEST_STATE_1	Indicates the state of the LOG_ARCHIVE_DEST_1 archiving destination.
ENABLE	Indicates that Oracle can archive to the destination.
LOG_ARCHIVE_FORMAT	Indicates the format for filenames of log files.

2. Because you want to archive to the standby database with service STANDBY1, you edit the file, adding the following entries:

```
log_archive_dest_2 = 'SERVICE=standby1 OPTIONAL REOPEN=180'
log_archive_dest_state_2 = ENABLE
```

Parameter/Option	Meaning
LOG_ARCHIVE_DEST_2	Indicates a new archiving destination. LOG_ARCHIVE_DEST_1 is already reserved for local archiving to `/fs1/arc_dest/`.
SERVICE	Indicates the service name of the standby database.
OPTIONAL	Indicates that Oracle can reuse online redo logs even if this destination fails.
REOPEN	Indicates how many seconds the archiving process waits before reattempting to archive to a previously failed destination.
LOG_ARCHIVE_DEST_STATE_2	Indicates the state of the LOG_ARCHIVE_DEST_2 archiving destination.
ENABLE	Indicates that Oracle can archive to the destination.

After editing the primary database initialization parameter file, create a copy for use by the standby database:

```
% cp /fs1/temp/initPROD1.ora /fs3/oracle/dbs/initSTANDBY1.ora
```

If the primary database initialization parameter file contains the IFILE parameter, you also need to copy the file referred to by the IFILE parameter to the standby site and, if necessary, make appropriate changes to it.

Step 5: Configure the Standby Database Parameter File.

You know that the initialization parameters shown in Table 11–1 play a key role in the standby database recovery process, and decide to edit them.

Table 12–1 Configuring Standby Database Initialization Parameters

Parameter	Setting
COMPATIBLE	This parameter must be the same at the primary and standby databases. Because it is already set to 8.1.6 in the primary database parameter file, you can leave the standby setting as it is.
CONTROL_FILES	This parameter must be different between the primary and standby databases. You decide to locate the control files in the `/fs2/dbs` directory.
DB_FILE_NAME_CONVERT	Set when you want to make your standby datafile filenames distinguishable from your primary database filenames. Most (but not all) of the datafiles are in the `/fs1/dbs` directory. You set this parameter to `/fs2/dbs` to convert the files in the `/dbs` subdirectory automatically; the others you will convert using ALTER DATABASE RENAME FILE statements.
DB_FILES	DB_FILES must be the same at both databases so that you allow the same number of files at the standby database as you allow at the primary database. Consequently, you leave this parameter alone. An instance cannot mount a database unless DB_FILES is equal to or greater than MAXDATAFILES.
DB_NAME	This value should be the same as the DB_NAME value in the production database parameter file. Consequently, you leave this parameter alone.
LOCK_NAME_SPACE	Specifies the name space that the distributed lock manager (DLM) uses to generate lock names. Set this value if the standby and primary databases share the same host. You decide to set the name to STANDBY1.
LOG_ARCHIVE_DEST_1	This parameter specifies the location of the archived redo logs. You must use this directory when performing manual recovery. You decide to set the value to `/fs2/arc_dest/`.
LOG_FILE_NAME_CONVERT	Set when you want to make your standby redo log filenames distinguishable from your primary database redo log filenames. Because your primary redo logs are located in `/fs1/dbs`, you decide to locate the standby logs in `/fs2/dbs`.
STANDBY_ARCHIVE_DEST	Oracle uses this value to create the name of the logs received from the primary site. You decide to set it to `/fs2/arc_dest/`.

Edit the standby database parameter file as follows (with edited values in bold):

```
db_name = prod1                 #The same as PRMYinit.ora
control_files = (/fs2/dbs/cf1.f)
compatible = 8.1.6
log_archive_start = TRUE
log_archive_dest_1='LOCATION=/fs2/arc_dest/'
log_archive_dest_state_1 = ENABLE
log_archive_format = log_%t_%s.arc
standby_archive_dest = /fs2/arc_dest/
db_file_name_convert = ('/fs1/dbs','/fs2/dbs')
log_file_name_convert = ('/fs1/dbs','/fs2/dbs')
lock_name_space = standby1
audit_trail=FALSE
o7_dictionary_accessibility=FALSE
global_names=FALSE
db_domain=regress.rdbms.dev.us.oracle.com
remote_login_passwordfile = exclusive

# default parameters for instance 1
processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
distributed_transactions=10
db_block_buffers=1000
db_files=200
shared_pool_size=10000000
```

Step 6: Start the Standby Database in Preparation for Managed Recovery.

Now that you have configured all network and parameter files, you can enable archiving to the standby database.

1. Set the ORACLE_SID environment variable to the same value as the SID parameter in the `tnsnames.ora` file on the primary site and the `listener.ora` file on the standby site as follows:

   ```
   % setenv ORACLE_SID stdby1
   ```

2. Start the instance.

   First, you start the standby database instance without mounting the standby database control file, as the following example shows:

   ```
   SQL> CONNECT sys/change_on_install@standby1 AS SYSDBA
   SQL> STARTUP NOMOUNT PFILE=/fs3/oracle/dbs/initSTANDBY1.ora;
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. Manually rename datafiles.

   Next, write a SQL script to rename the datafiles not captured by the DB_FILE_NAME_CONVERT parameter:

```
ALTER DATABASE RENAME FILE /fs1/dbs2/tbs_42.f,
                           /fs1/dbs2/tbs_51.f,
                           /fs1/dbs2/tbs_52.f,
                           /fs1/dbs2/tbs_03.f,
                           /fs1/dbs3/tbs_14.f,
                           /fs1/dbs3/tbs_25.f,
                           /fs1/dbs3/tbs_33.f,
                           /fs1/dbs3/tbs_43.f,
                           /fs1/dbs3/tbs_53.f,
TO
                           /fs2/dbs/tbs_42.f,
                           /fs2/dbs/tbs_51.f,
                           /fs2/dbs/tbs_52.f,
                           /fs2/dbs/tbs_03.f,
                           /fs2/dbs/tbs_14.f,
                           /fs2/dbs/tbs_25.f,
                           /fs2/dbs/tbs_33.f,
                           /fs2/dbs/tbs_43.f,
                           /fs2/dbs/tbs_53.f
/
```

4. Enable changes to the primary database parameter file.

   Finally, you enable the changes you made to the primary database parameter file so that the standby database can begin receiving archived redo logs:

```
SQL> CONNECT sys/change_on_install@prod1 as sysdba
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 OPTIONAL REOPEN=180';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = ENABLE;
```

Step 7: Identify the Logs in the Gap Sequence.

Because you have enabled the changes to the primary database parameter file, the primary database is now able to archive to the standby service name. Before you can perform managed recovery, however, you must synchronize the standby database by applying those logs containing changes made after the primary database backup, but before the first log received by the standby database.

Write the following SQL script and run it on the standby database:

```
SELECT   high.thread#, "LowGap#", "HighGap#"
FROM
 (SELECT   thread#, MIN(sequence#)-1 "HighGap#"
  FROM
```

```
        (SELECT  a.thread#, a.sequence#
         FROM
                (SELECT * FROM V$ARCHIVED_LOG) a,
                (SELECT thread#, MAX(next_change#) gap1
                 FROM  V$LOG_HISTORY
                 GROUP BY thread#       )  b
         WHERE
                a.thread# = b.thread#
                AND
                a.next_change# > gap1
        ) GROUP BY thread#
  ) high,
  (SELECT thread#, MIN(gap2) "LowGap#"
   FROM
        (SELECT thread#, sequence#+1 gap2
         FROM   V$LOG_HISTORY, V$DATAFILE
         WHERE
                checkpoint_change# <= next_change#
                  AND
                checkpoint_change# >= first_change#
                  AND
                enabled = 'READ WRITE'
        ) GROUP BY thread#
   ) low
 WHERE
    low.thread# = high.thread#
      AND
    "LowGap#" < "HighGap#";
```

The output of the query is as follows:

```
SQL> @gap
THREAD#    LowGap#    HighGap#
---------- ---------- ----------
       1   250        252
```

Hence, you must apply log sequences 250, 251, and 252 to synchronize the standby database before initiating managed recovery.

Step 8: Copy the Logs in the Gap Sequence to the Standby File System.

The archived log filenames generated by gap sequence queries on the standby database are generated by the LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT parameters in the initialization parameter file. Before transmitting the logs from the primary site to the standby site, you must determine the correct filenames.

First, you determine the filenames of the logs in the gap that were archived by the primary database. After connecting to the primary database using SQL*Plus, issue the following SQL query to obtain the names:

```
SQL> CONNECT sys/change_on_install@prod1 AS SYSDBA
SQL> SELECT name FROM v$archived_log WHERE sequence# IN (250,251,252);

NAME
--------------------------------------------------------------------------------
/fs1/arc_dest/log_1_250.arc
/fs1/arc_dest/log_1_251.arc
/fs1/arc_dest/log_1_252.arc
```

The gap sequence in this case consists of log_1_250.arc, log_1_251.arc, and log_1_252.arc. The settings for LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT in the standby database parameter file are as follows:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION=/fs2/arc_dest/'
LOG_ARCHIVE_FORMAT = log_%t_%s.arc
```

You move the gap sequence logs from the primary file system to the standby file system, renaming them according to values for the LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT initialization parameters at the standby site:

```
% cp /fs1/arc_dest/log_1_250.arc /fs2/arc_dest/log_1_250.arc
% cp /fs1/arc_dest/log_1_251.arc /fs2/arc_dest/log_1_251.arc
% cp /fs1/arc_dest/log_1_252.arc /fs2/arc_dest/log_1_252.arc
```

Step 9: Apply the Logs in the Gap Sequence to the Standby Database.

Now you can apply the gap sequence logs using the RECOVER AUTOMATIC STANDBY DATABASE statement. This statement uses the values of the LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT parameters to construct the target filename. Once the gap sequence logs have been applied, the standby database is synchronized with the primary database and can be placed in managed recovery mode.

While connected to the standby database in SQL*Plus, recover the database using the AUTOMATIC option:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
ORA-00279: change 35083 generated at 08/16/1999 14:08:37 needed for thread 1
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
CANCEL
Media recovery cancelled.
SQL>
```

After recovering the gap sequence logs, Oracle prompts you for the name of a log that does not exist. The reason is that the recovery process does not know about the logs archived to the STANDBY service by the primary database. Cancel recovery at this point by entering CANCEL at the prompt.

Step 10: Place the Standby Database in Managed Recovery Mode.

You can now enable managed recovery using the RECOVER MANAGED STANDBY DATABASE statement. You decide to use the TIMEOUT option of the RECOVER statement to specify a time interval of 20 minutes so that Oracle waits the specified number of minutes to write the requested archived log entry to the directory of the standby database control file. If the requested archived log entry is not written to the standby database control file directory within the specified time interval, the recovery operation is canceled.

While connected to the standby database using SQL*Plus, place the standby database in managed recovery mode:

```
SQL> RECOVER MANAGED STANDBY DATABASE TIMEOUT 20;
```

Oracle now begins managed recovery. As the primary database archives redo logs to the standby site, the standby database automatically applies them.

Revised Scenario 2: Creating a Standby Database on a Remote Host

Chapter 5 in the 8.1.6 *Oracle8i Standby Database Concepts and Administration* manual has a section on page 5-14 titled, "Scenario 2: Creating a Standby Database on a Remote Host." The following section supersedes the section in the 8.1.6 documentation.

Scenario 2: Creating a Standby Database on a Remote Host

This scenario describes the creation of a standby database STANDBY1 on a remote host. The following assumptions are being made:

- You can perform a consistent backup.
- TCP/IP is used to connect to primary and standby databases.
- The standby database is part of a managed standby environment.
- The remote host name is STBYHOST.
- `PRMYinit.ora` is the initialization parameter file for the primary database.
- `STBYinit.ora` is the initialization parameter file for the standby database.

Step 1: Back Up the Primary Database Datafiles.

Create the backup that will form the basis for the standby database.

1. Query the primary database to determine the datafiles. Invoke SQL*Plus and query the V$DATAFILE view to obtain a list of the primary database datafiles, as the following example shows:

```
SQL> SELECT name FROM v$datafile;
NAME
----------------------------------------------------------------------
/vobs/oracle/dbs/dbf_1.f
1 row selected.
```

2. Shut down the primary database to make a consistent backup of the datafiles:

```
SQL> SHUTDOWN IMMEDIATE;
```

3. Copy the primary database datafiles to a temporary location (/backup), as the following example shows:

```
% cp /vobs/oracle/dbs/dbf_1.f /backup
```

4. Start and mount the primary database without opening it:

```
SQL> STARTUP MOUNT PFILE=PRMYinit.ora;
```

Step 2: Create the Standby Database Control File.

1. Create the standby database control file by issuing the following statement:

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/backup/stbycf.f'
```

   The standby database control file and the primary database datafiles are in the same temporary location at the primary site to make copying to the standby site easier.

2. Ensure that the primary database is in ARCHIVELOG mode and that automatic archival is enabled. Issue the ARCHIVE LOG LIST statement:

```
SQL> ARCHIVE LOG LIST
```

   If the output from the ARCHIVE LOG LIST statement displays "No Archive Mode," set the log archive mode as follows:

```
SQL> ALTER DATABASE ARCHIVELOG;
```

3. Open the primary database:

```
SQL> ALTER DATABASE OPEN;
```

Step 3: Transfer the Datafiles and Control File to the Standby Site.

Copy the primary database datafiles and the standby control file from the temporary location at the primary site to the standby site, as the following example shows:

```
% rcp /backup/* STBYHOST:/fs2/oracle/stdby
```

Step 4: Configure the Network Files.

This scenario assumes that the TCP/IP network protocol is used to connect to the primary and the standby databases. This step involves editing the following parameter files:

- `tnsnames.ora` file on the primary site
- `listener.ora` file on the standby site

1. Configure the `tnsnames.ora` file.

   You need to edit the `tnsnames.ora` file and add an entry with the following format:

   ```
   standby_service_name = (DESCRIPTION=
      (ADDRESS=(PROTOCOL=tcp) (PORT=port_number)(HOST=host_name))
      (CONNECT_DATA=(SID=standby_sid)))
   ```

   The `tnsnames.ora` file is typically located in the $ORACLE_HOME/network/admin directory on the primary site. Substitute appropriate values for standby_service_name, port_number, host_name, and standby_sid, as the following example shows:

   ```
   standby1 = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
            (PORT=1521)(HOST=STBYHOST))
            (CONNECT_DATA=(SID=stdby1)))
   ```

2. Configure the `listener.ora` file.

   You need to edit the `listener.ora` file and add two entries with the following format:

   ```
   STANDBY_LISTENER = (ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)
                                 (PORT=port_number)(HOST=host_name)))

   SID_LIST_STANDBY_LISTENER = (SID_LIST=
                     (SID_DESC=(SID_NAME=standby_sid)(ORACLE_HOME=/oracle_home)))
   ```

   The `listener.ora` file is typically located in the $ORACLE_HOME/network/admin directory on the standby site. Substitute appropriate

values for port_number, host_name, standby_sid, and oracle_home, as the following example shows:

```
STDBY1_LISTENER = (ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)
                        (PORT=1521)(HOST=STBYHOST)))

SID_LIST_STDBY1_LISTENER = (SID_LIST=
          (SID_DESC=(SID_NAME=stdby1)(ORACLE_HOME=/oracle)))
```

Make sure the SID_NAME in the `listener.ora` file matches the SID in the `tnsnames.ora` file. Also, make sure the PORT and HOST values are the same in the two files. Note that you have the option of creating a new listener or adding a new address to an existing listener.

Step 5: Start the Listener on the Standby Site.

The two entries that you added in step 4 define a listener to listen for connections to standby database stdby1 on port 1521. In this step, you must start the `stdby1_listener` listener. For example:

```
% lsnrctl start stdby1_listener;
```

Step 6: Configure the Standby Initialization Parameter File.

1. Copy the primary database initialization parameter file from the primary site to the standby site. From the primary site, issue a command similar to the following:

   ```
   % rcp /vobs/oracle/dbs/PRMYinit.ora STBYHOST:/fs2/oracle/stdby/STBYinit.ora
   ```

2. Edit the standby initialization parameter file (`STBYinit.ora`). Edit the following parameters:

Parameter	Value
CONTROL_FILES	stbycf.f
STANDBY_ARCHIVE_DEST	/fs2/oracle/stdby/
LOG_ARCHIVE_DEST_1	/fs2/oracle/stdby/
LOG_ARCHIVE_FORMAT	stdby_%t_%s
DB_FILE_NAME_CONVERT	('/vobs/oracle/dbs','/fs2/oracle/stdby')

Parameter	Value
LOG_FILE_NAME_ CONVERT	('/vobs/oracle/dbs','/fs2/oracle/stdby')
LOG_ARCHIVE_START	FALSE

The `STBYinit.ora` file appears as follows:

```
#
#parameter file STBYinit.ora
#

db_name=prod1                    # The same as PRMYinit.ora

# The following parameter has changed from PRMYinit.ora
control_files=/fs2/oracle/stdby/stbycf.f

# The following parameters are the same as PRMYinit.ora
audit_trail=FALSE
o7_dictionary_accessibility=FALSE
global_names=FALSE
db_domain=regress.rdbms.dev.us.oracle.com
commit_point_strength=1

processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
distributed_transactions=10
db_block_buffers=100
shared_pool_size=4000000
ifile=/vobs/oracle/work/tkinit.ora # Verify that file exists on the standby site
                                   # and that the file specification is valid

# specific parameters for standby database
log_archive_format = stdby_%t_%s.arc
standby_archive_dest=/fs2/oracle/stdby/
log_archive_dest_1='LOCATION=/fs2/oracle/stdby/'
db_file_name_convert=('/vobs/oracle/dbs','/fs2/oracle/stdby')
log_file_name_convert=('/vobs/oracle/dbs','/fs2/oracle/stdby')
log_archive_start=FALSE
log_archive_trace=127
```

Step 7: Copy the Standby Initialization Parameter File.

1. Make a copy of the `STBYinit.ora` file by issuing the following command:

   ```
   % cp STBYinit.ora Failover.ora
   ```

   Edit `Failover.ora` so if you fail over to the stdby1 standby database, you can use the `Failover.ora` file as the initialization parameter file for the new primary database. Make sure you use appropriate values for the LOG_ARCHIVE_DEST_*n* parameters.

2. Edit the `tnsnames.ora` file on the standby site in case failover to the standby database occurs. See step 4 for information on how to configure the `tnsnames.ora` file.

Step 8: Start the Standby Database.

Start the standby database to enable archiving.

1. Set the ORACLE_SID environment variable to the same value as the SID parameter in the `tnsnames.ora` file on the primary site and the `listener.ora` file on the standby site as follows:

   ```
   % setenv ORACLE_SID stdby1
   ```

2. Start SQL*Plus:

   ```
   SQL> CONNECT sys/sys_password as sysdba
   ```

3. Start the standby database instance without mounting the database:

   ```
   SQL> STARTUP NOMOUNT PFILE=STBYinit.ora;
   ```

4. Mount the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

Step 9: Configure the Primary Initialization Parameter File.

1. Specify the archive destination by adding the following entry to the `PRMYinit.ora` file:

   ```
   LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY REOPEN=60'
   ```

2. Enable the archive destination state by adding the following entry to the `PRMYinit.ora` file:

   ```
   LOG_ARCHIVE_DEST_STATE_2 = ENABLE
   ```

3. Issue the following statements to ensure that the initialization parameters you have set in this step take effect:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=standby1 MANDATORY REOPEN=60';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Step 10: Apply the Logs in the Gap Sequence.

1. On the primary database, archive the current redo log as follows:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

2. On the standby database, run the following SQL script to identify the archived redo logs in the gap sequence:

```
SELECT   high.thread#, "LowGap#", "HighGap#"
FROM
 (SELECT   thread#, MIN(sequence#)-1 "HighGap#"
  FROM
        (SELECT  a.thread#, a.sequence#
         FROM
                (SELECT * FROM V$ARCHIVED_LOG) a,
                (SELECT thread#, MAX(next_change#) gap1
                 FROM  V$LOG_HISTORY
                 GROUP BY thread#       )  b
         WHERE
                a.thread# = b.thread#
                AND
                a.next_change# > gap1
        ) GROUP BY thread#
 ) high,
 (SELECT thread#, MIN(gap2) "LowGap#"
  FROM
        (SELECT thread#, sequence#+1 gap2
         FROM   V$LOG_HISTORY, V$DATAFILE
         WHERE
                checkpoint_change# <= next_change#
                  AND
                checkpoint_change# >= first_change#
                  AND
                enabled = 'READ WRITE'
        ) GROUP BY thread#
  ) low
WHERE
   low.thread# = high.thread#
     AND
```

```
"LowGap#" < "HighGap#";

 THREAD#    LowGap#    HighGap#
---------- ---------- ----------
         1         90         92
```

3. On the primary database, obtain the filenames of the logs in the gap sequence by performing a query on the V$ARCHIVED_LOG view as follows:

```
SELECT name FROM v$archived_log
WHERE thread#=1 AND sequence#<=92 AND sequence#>=90;

NAME
----------------------------------------
/vobs/oracle/dbs/r_1_90.arc
/vobs/oracle/dbs/r_1_91.arc
/vobs/oracle/dbs/r_1_92.arc
```

4. Transfer the logs in the gap sequence from the primary database to the standby database as follows:

```
% rcp /vobs/oracle/dbs/r_1_90.arc STBYHOST:/fs2/oracle/stdby/stdby_1_90.arc
% rcp /vobs/oracle/dbs/r_1_91.arc STBYHOST:/fs2/oracle/stdby/stdby_1_91.arc
% rcp /vobs/oracle/dbs/r_1_92.arc STBYHOST:/fs2/oracle/stdby/stdby_1_92.arc
```

5. On the standby database, issue the following SQL statement to manually apply the logs in the gap sequence:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
```

Step 11: Place the Standby Database in Managed Recovery Mode.

On the standby database, enable managed recovery by issuing the following SQL statement:

```
SQL> RECOVER MANAGED STANDBY DATABASE;
```

13

Using Oracle Trace

This chapter describes new and changed features for release 8.1.7. The topics in this chapter include:

- Overview of Oracle Trace
- Using Oracle Trace Manager
- Using Oracle Trace Data Viewer
- Manually Collecting Oracle Trace Data

Overview of Oracle Trace

Oracle Trace is a general-purpose data collection product and is part of the Oracle Enterprise Manager systems management product family. The Oracle server uses Oracle Trace to collect performance and resource utilization data, such as SQL parse, execute, and fetch statistics, and wait statistics.

> **See Also:** *Oracle Enterprise Manager Oracle Trace User's Guide* and *Oracle Enterprise Manager Oracle Trace Developer's Guide*
>
> These books contain a complete list of events and data that you can collect for the Oracle server, as well as information on how to implement tracing in your own products and applications.

Using Oracle Trace Data

Among the many advantages of using Oracle Trace is the integration of Oracle Trace with other applications. You can use Oracle Trace data collected for the Oracle server in the following applications, as shown in Figure 13–1:

- Oracle Expert

  You can use information collected with Oracle Trace as an optional source of SQL workload data in Oracle Expert. This SQL data is used when recommending the addition or removal of indexes.

  > **See Also:** *Database Tuning with the Oracle Tuning Pack*

- Oracle Trace Data Viewer

  Oracle Trace Data Viewer is a simple viewer for inspecting Oracle Trace collections containing SQL and wait statistics. You can export Oracle Trace data to the following products for further analysis:

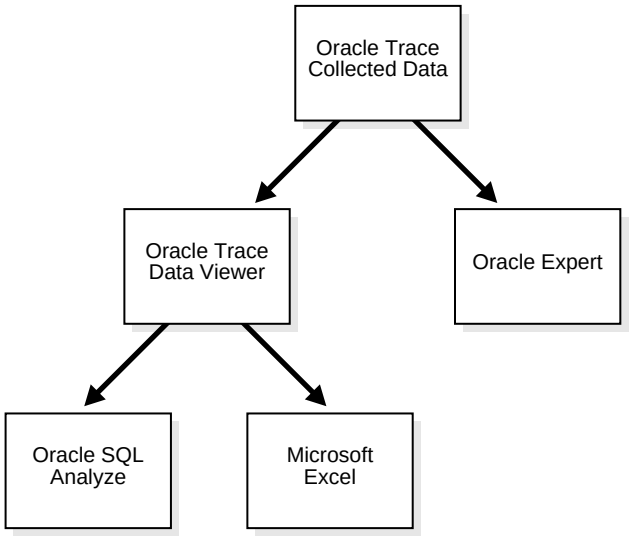
  – SQL Analyze

    Select one or more rows in Data Viewer, and save the SQL statement text to a file that you can import into SQL Analyze. Then, use SQL Analyze to tune these individual statements.

  – Third-Party Tools, such as Microsoft Excel

    SQL in Data Viewer can be saved to a CSV (Comma Separated Value) file for viewing in third-party tools, such as Microsoft Excel.

Figure 13–1 Integration of Oracle Trace with Other Applications



Using Oracle Trace Data in Oracle Expert

Use Oracle Trace to collect workload data for use in the Oracle Expert application. Oracle Trace collects resource utilization statistics for SQL statements run against a database in real time. Oracle Trace lets you collect data about all the SQL statements run against a database *during* periods of poor performance.

You control the scheduling and duration of an Oracle Trace collection period. To get SQL workload data for a 15-minute period of poor performance, stop collection immediately after the poor performance interval ends.

Importing Data Viewer SQL Into Oracle SQL Analyze

While using Data Viewer, you can select one or more rows in the top portion of the Data View window to save to a file. When you choose SQL (SQL Analyze Format) from File/Save, a file containing query text is saved. You can then import this *.sql file into Oracle SQL Analyze for tuning the selected statements.

Oracle SQL Analyze can show the execution plan for individual queries and let you experiment with various optimizer modes and hints.

Importing Data Viewer Information into Third-Party Tools

While using Data Viewer, you can select one or more rows in the top portion of the Data View window to save to a file. When you choose the CSV file format, Oracle Trace creates a *.csv file that you can load into a third-party tool, such as Microsoft Excel.

Using Oracle Trace Manager

Oracle Trace provides a graphical Oracle Trace Manager application to create, schedule, and administer Oracle Trace collections for products containing Oracle Trace calls.

The Oracle server has been coded with Oracle Trace API calls to collect both SQL and wait statistics with a minimum of resource overhead. Using the Oracle Trace Manager graphical user interface, you can do the following:

- Schedule trace collections
- Filter trace collections by user
- Filter trace collections by type of wait event
- Format collected data to database tables to preserve historical data
- View SQL and wait statistics using Oracle Trace Data Viewer

Managing Collections

Use and control of Oracle Trace revolves around the concept of a *collection*. A collection is data collected for events that occurred while a product with Oracle Trace code was running.

With the Oracle Trace Manager, you can schedule and manage collections. When creating a collection, you define the attributes of the collection, such as the collection name, the data to be included in the collection, and the start and end times. The Oracle Trace Manager includes a Collection Wizard that facilitates the creation and execution of collections.

After you create a collection, you can run it immediately or schedule it to run at a specific time or at specified intervals. When a collection runs, it produces a file containing the data for the products participating in the collection. You can also use a collection as a template for creating other similar collections.

Collecting Event Data

An event is the occurrence of some activity within a product. Oracle Trace collects data for predefined events occurring within a software product created with the Oracle Trace API. That is, the product is embedded with Oracle Trace API calls. An example of an event is a parse or fetch.

There are two types of events:

- Point events

  Point events represent an instantaneous occurrence of something in the product. An example of a point event is an error occurrence.

- Duration events

  Duration events have a beginning and an ending. An example of a duration event is a transaction. Duration events can have other events occur within them; for example, an error can occur *within* a transaction.

The Oracle server has 17 events. The following are three of these events:

- Database Connection: A point event that records data, such as the server login user name.
- SQL Parse: One of the series of SQL processing duration events. This event records a large set of data, such as sorts, resource use, and cursor numbers.
- RowSource: Data about the execution plan, such as SQL operation, position, object identification, and number of rows processed by a single row source within an execution plan.

Accessing Collected Data

During a collection, Oracle Trace stores event data in memory and periodically writes it to a collection binary file. This method ensures low resource overhead associated with the collection process. You can access event data collected in the binary file by formatting the data to predefined tables, which makes the data available for fast, flexible access. These predefined tables are called Oracle Trace formatter tables.

Oracle Trace Manager provides a mechanism for formatting collection data immediately after a collection or at a later time.

When formatting a collection, identify the database where Oracle Trace Manager creates the formatted collection as follows:

1. Using Oracle Trace Manager, select a collection to format.
2. Choose the Format statement.
3. Specify a target database where the data is to reside.

The collection you select determines which collection definition file and data collection file is used. The formatted target database determines where the formatted collection data is stored.

After the data is formatted, you can access the data using the Oracle Trace Data Viewer or by using SQL reporting tools and scripts.

Also, you can access event data by running the Detail report from the Oracle Trace reporting utility. This report provides a basic mechanism for viewing a collection's results. You have limited control over what data is reported and how it is presented.

> **See Also:** *Oracle Enterprise Manager Oracle Trace Developer's Guide* for more information about predefined SQL scripts and the Detail reports

Using Oracle Trace Data Viewer

After using Oracle Trace to collect data, run the Data Viewer by selecting "View Formatted Data..." from the Oracle Trace Collection menu, or select it directly from the Oracle Diagnostics Pack toolbar. Data Viewer can compute SQL and wait statistics and resource utilization metrics from the raw data that is collected. After Data Viewer computes statistics, targeting resource-intensive SQL becomes a much simpler task.

Data Viewer computes SQL statistics from data collected by Oracle Trace Manager for all executions of a query during the collection period. Resource utilization during a single execution of a SQL statement can be misleading due to other concurrent activities on the database or node. Combining statistics for all executions can show a clearer picture about the typical resource utilization occurring when a given query is run.

Note: You can filter out SQL statements run by SYS.

Using Oracle Trace Predefined Data Views

SQL and wait statistics are presented in a comprehensive set of Oracle Trace predefined data views. Within wait statistics, a data view is the definition of a query into the data collected by Oracle Trace. A data view consists of items or statistics to be returned and, optionally, a sort order and limit of rows to be returned.

With the data views provided by Data Viewer, you can do the following:

- Examine important statistical data; for example, elapsed times or disk-to-logical-read hit rates.
- Drill down as needed to get additional details about the statement's execution.

In addition to the predefined data views, you can define your own data views using the Create Data View Wizard.

After Data Viewer has computed SQL and wait statistics, a dialog box showing the available data views appears. SQL Statistic data views are grouped by I/O, Parse, Elapsed Time, CPU, Row, Sort, and Wait statistics, as shown in Figure 13–2. A description of the selected data view is shown on the right-hand side of the screen.

Figure 13–2 Oracle Trace Data Viewer - Collection Screen

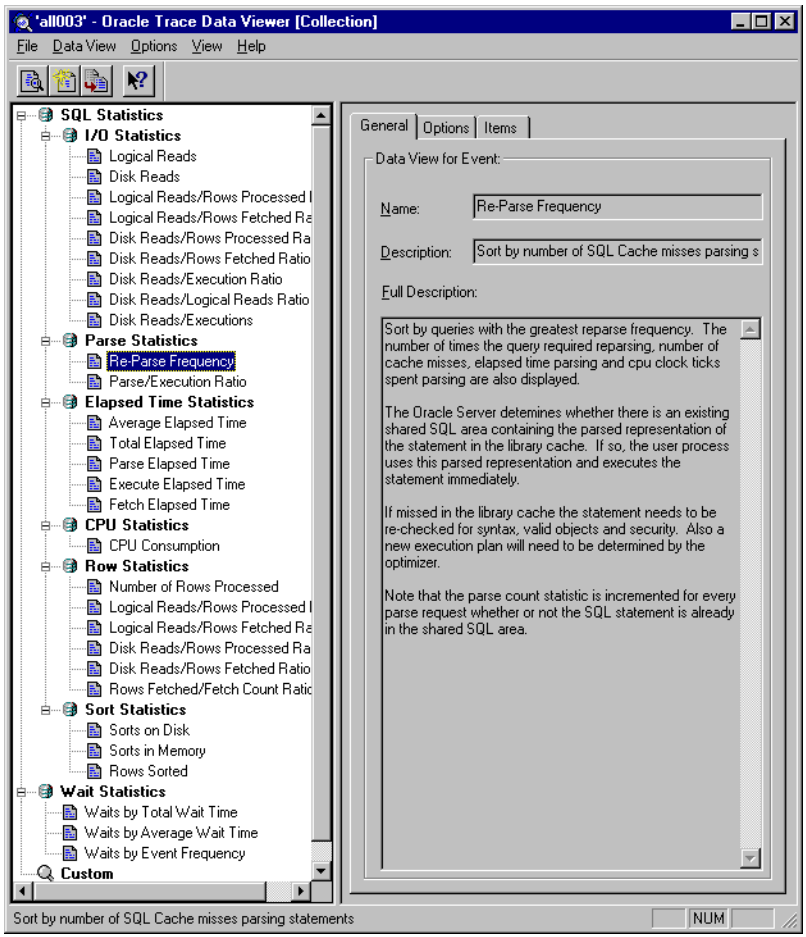


Table 13–1 explains the predefined data views shown in the previous figure as provided by Oracle Trace.

Table 13–1 Predefined Data Views Provided By Oracle Trace

View Name	Sort By	Data Displayed	Description
Logical Reads	Total number of logical reads performed for each distinct query.	Total number of blocks read during parses, executions, and fetches. Logical reads for parses, executions, and fetches of the query.	Logical data block reads include data block reads from both memory and disk. Input/output is one of the most expensive operations in a database system. I/O intensive statements can monopolize memory and disk usage, causing other database applications to compete for these resources.
Disk Reads	Queries that incur the greatest number of disk reads.	Disk reads for parses, executions, and fetches.	Disk reads, also known as physical I/O, are database blocks read from disk. The disk read statistic is incremented once per block read regardless of whether the read request was for a multiblock read or a single block read. Most physical reads load data, index, and rollback blocks from the disk into the buffer cache. A physical read count can indicate a high miss rate within the data buffer cache.
Logical Reads/ Rows Fetched Ratio	Number of logical reads divided by the number of rows fetched for all executions of the current query.	Total logical I/O. Total number of rows fetched.	The more blocks accessed relative to the number of rows actually returned, the more expensive each row is to return. This can be a rough indication of the relative expense of a query.
Disk Reads/ Rows Fetched Ratio	Number of disk reads divided by the number of rows fetched for all executions of the current query.	Total disk I/O. Total number of rows fetched.	The greater the number of blocks read from disk for each row returned, the more expensive each row is to return. This can be a rough indication of the relative expense of a query.
Disk Reads/ Execution Ratio	Total number of disk reads per distinct query divided by the number of executions of that query.	Total disk I/O. Logical I/O for the query as well as the number of executions of the query.	Indicates which statements incur the greatest number of disk reads per execution.

Table 13–1 Predefined Data Views Provided By Oracle Trace

View Name	Sort By	Data Displayed	Description
Disk Reads/ Logical Reads Ratio	Greatest miss rate ratio of disk to logical reads.	Individual logical reads. Disk reads for the query as well as the miss rate.	The miss rate indicates the percentage of times the Oracle server needed to retrieve a database block on disk as opposed to locating it in the data buffer cache in memory. The miss rate for the data block buffer cache is derived by dividing the physical reads by the number of accesses made to the block buffer to retrieve data in consistent mode plus the number of blocks accessed through single block gets. Memory access is much faster than disk access; the greater the hit ratio, the better the performance.
Reparse Frequency	Queries with the greatest reparse frequency.	Number of cache misses. Total number of parses. Total elapsed time parsing. Total CPU clock ticks spent parsing.	The Oracle server determines whether there is an existing shared SQL area containing the parsed representation of the statement in the library cache. If so, then the user process uses this parsed representation and runs the statement immediately. If missed in the library cache, then recheck the statement for syntax, valid objects, and security. Also, the optimizer must determine a new execution plan. The parse count statistic is incremented for every parse request, regardless of whether the SQL statement is already in the shared SQL area.
Parse/Execution Ratio	Number of parses divided by the number executions per statement.	Individual number of parses. Number of executions.	The count of parses to executions should be as close to one as possible. If there are a high number of parses per execution, then the statement has been needlessly reparsed. This could indicate the lack of use of bind variables in SQL statements or poor cursor reuse. Reparsing a query means that the SQL statement has to be rechecked for syntax, valid objects, and security. Also, a new execution plan needs to be determined by the optimizer.
Average Elapsed Time	Greatest average time spent parsing, executing, and fetching on behalf of the query.	Individual averages for parse, execution, and fetch.	The average elapsed time for all parses, executions, and fetches per execution are computed, then summed for each distinct SQL statement in the collection.

Table 13–1 Predefined Data Views Provided By Oracle Trace

View Name	Sort By	Data Displayed	Description
Total Elapsed Time	Greatest total elapsed time spent parsing, executing, and fetching on behalf of the query.	Individual elapsed times for parses, executions, and fetches.	The total elapsed time for all parses, executions, and fetches are computed. They are then summed for each distinct SQL statement in the collection.
Parse Elapsed Time	Total elapsed time for all parses associated with a distinct SQL statement.	SQL cache misses. Elapsed times for execution and fetching. Total elapsed time.	During parsing, the Oracle server determines whether there is an existing shared SQL area containing the parsed representation of the statement in the library cache. If so, then the user process uses this parsed representation and runs the statement immediately. If missed in the library cache, then the statement needs to be rechecked for syntax, valid objects, and security. Also, a new execution plan needs to be determined by the optimizer.
Execute Elapsed Time	Greatest total elapsed time for all executions associated with a distinct SQL statement.	Total elapsed time. Individual elapsed times for parsing and fetching.	The total elapsed time of all execute events for all occurrences of the query within an Oracle Trace collection.
Fetch Elapsed Time	Greatest total elapsed time for all fetches associated with a distinct SQL statement.	Number of rows fetched. Number of fetches. Number of executions. Total elapsed time. Individual elapsed times for parsing and executing.	The total elapsed time spent fetching data on behalf of all occurrences of the current query within the Oracle Trace collection.

Table 13–1 Predefined Data Views Provided By Oracle Trace

View Name	Sort By	Data Displayed	Description
CPU Statistics	Total CPU clock ticks spent parsing, executing, and fetching on behalf of the SQL statement.	CPU clock ticks for parses, executions, and fetches. Number of SQL cache misses and sorts in memory.	When SQL statements and other types of calls are made to an Oracle server, a certain amount of CPU time is necessary to process the call. Average calls require a small amount of CPU time. However, a SQL statement involving a large amount of data, a runaway query, in memory sorts, or excessive reparsing can potentially consume a large amount of CPU time. CPU time displayed is in terms of the number of CPU clock ticks on the operating system housing the database.
Number of Rows Returned	Greatest total number of rows returned during execution and fetch for the SQL statement.	Number of rows returned during the fetch operation as well as the execution rows.	Targets queries that manipulate the greatest number of rows during fetching and execution. This can mean that high gains can be made by tuning row-intensive queries.
Rows Fetched/Fetch Count Ratio	Number of rows fetched divided by the number of fetches.	Individual number of rows fetched. Number of fetches.	This ratio shows how many rows were fetched at a time. It indicates the level to which array fetch capabilities have been utilized. A ratio close to one indicates an opportunity to optimize code by using array fetches.
Sorts on Disk	Queries that did the greatest number of sorts on disk.	Sort statistics for SQL statements. Number of in-memory sorts. Total number of rows sorted.	Sorts on disk are sorts that could not be performed in memory; therefore, they are more expensive because memory access is faster than disk access.
Sorts in Memory	Queries that did the greatest number of sorts in memory.	Sort statistics for SQL statements. Number of disk sorts. Total number of rows sorted.	Sorts in memory are sorts that could be performed completely within the sort buffer in memory without using the temporary tablespace segments.
Rows Sorted	Queries that sorted the greatest number of rows.	Number of in memory sorts. Number of sorts on disk.	Returns sort statistics for SQL statements ordered by queries that sorted the greatest number of rows.

Table 13–1 Predefined Data Views Provided By Oracle Trace

View Name	Sort By	Data Displayed	Description
Waits by Total Wait Time	Highest total wait time per distinct type of wait.	Average wait time, total wait time, and number of waits per wait type.	Waits are sorted by the wait description or type that had the greatest cumulative wait time for all occurrences of the wait type within the collection.
Waits by Average Wait Time	Highest average wait time per wait type.	Average wait time, total wait time, and number of waits per wait type.	Waits are sorted by the wait description or type that had the greatest average wait time for all occurrences of the wait type within the collection.
Waits by Event Frequency	Frequency of waits per wait type.	Number of waits per wait type, average wait time, and total wait time.	Waits are sorted by the wait events or wait descriptions that appear most frequently within the collection.

Viewing Oracle Trace Data

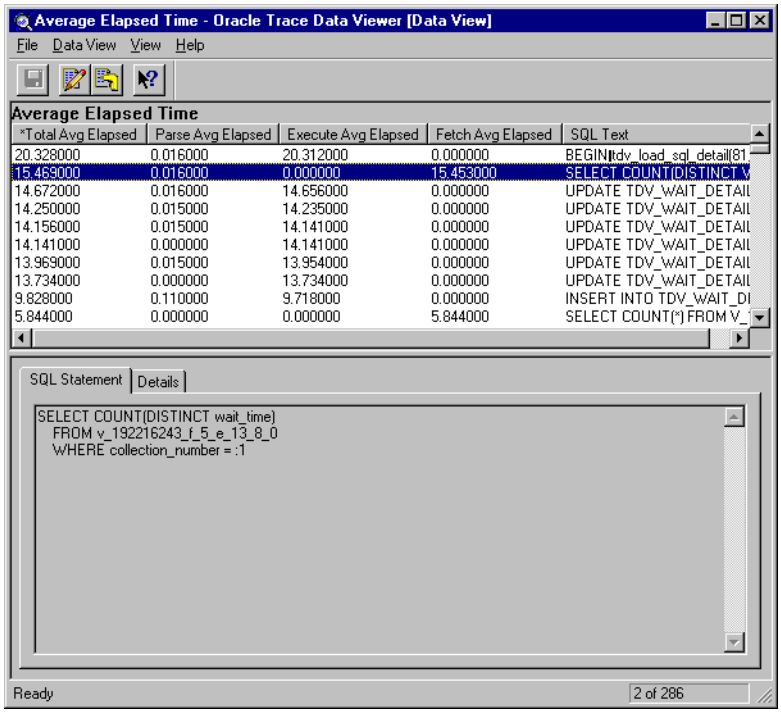
Double click on SQL or wait event data views to query the collection data and display data sorted by criteria described in the data view's description.

Double clicking the "Disk Reads/Log Reads Ratio" view returns data sorted by queries with the highest data buffer cache miss rate. This also displays the individual disk and logical read values.

Double clicking the "Average Elapsed Time" data view returns data sorted by queries that took the greatest average elapsed time to parse, execute, and fetch. It also displays the average elapsed times for parsing, execution, and fetching.

Figure 13–3 shows data in the "Average Elapsed Time" data view. Query text and statistics appear in the top portion of the window. Click any column headers to sort rows by the statistic in that column.

Figure 13–3 Oracle Trace Data Viewer - Data View Screen



The current data view's SQL text is shown in the lower portion of the window in the SQL Statement property sheet. Full statistical details about the current data view also appear in the Details property sheet.

When examining a data view like that shown in Figure 13–3, you can print either of the following:

- Data view statistics, located in the top portion of the screen
- Current SQL statement text in formatted output plus details on all statistical data collected for the selected query, located in the Details property page

Window focus at the time of printing determines which portion of the screen is printed. For example, if focus is on the top portion of the screen, then the tabular form of all statistics and SQL for this data view is printed.

Understanding the SQL Statement Property Page in Oracle Trace

The SQL Statement property page displays the selected query in a formatted output.

Understanding the Details Property Page in Oracle Trace

The Details property page displays a detailed report on statistics for all executions of a given query within an Oracle Trace collection. Text for the selected SQL statement is posted at the end of the property page.

Example of the Details Property Page in Oracle Trace

Statistics for all parses, executions, and fetches of the SQL statement.

The number of misses in library cache during parse: 1.000000

Elapsed time statistics for the SQL statement:

```
Average Elapsed Time:     0.843000
Total Elapsed Time:       0.843000

Total Elapsed Parse:      0.000000
Total Elapsed Execute:    0.843000
Total Elapsed Fetch:      0.000000

Average Elapsed Parse:    0.000000
Average Elapsed Execute:  0.843000
Average Elapsed Fetch:    0.000000
```

Number of times parse, execute, and fetch were called:

```
Number of Parses:       1
Number of Executions:   1
Number of Fetches:      0
```

Logical I/O statistics for parse, execute, and fetch calls:

```
Logical I/O for Parses:             1
Logical I/O for Executions:       247
Logical I/O for Fetches:            0
Logical I/O Total:                248
```

Disk I/O statistics for parse, execute, and fetch calls:

```
Disk I/O for Parses:                0
Disk I/O for Executions:           28
Disk I/O for Fetches:               0
Disk I/O Total:                    28
```

CPU statistics for parse, execute, and fetch calls:

```
CPU for Parses:               0
CPU for Executions:       62500
CPU for Fetches:              0
CPU Total:                62500
```

Row statistics for execute and fetch calls:

```
Rows processed during Executions:         104
Rows processed during Fetches:              0
Rows Total:                               104
```

Sort statistics for execute and fetch calls:

```
Sorts on disk:             0
Sorts in memory:           2
Sort rows:               667
```

Hit Rate - Disk I/O divided by Logical I/O: 0.112903

Logical I/O performed divided by rows actually processed: 2.384615

Disk I/O performed divided by number of executions: 28.000000

The number of parses divided by number of executions: 1.000000

The number of rows fetched divided by the number of fetches: 0.000000

```
INSERT INTO tdv_sql_detail
 (collection_number, sql_text_hash,
  "LIB_CACHE_ADDR")
 SELECT DISTINCT collection_number,
                 sql_text_hash,
                 "LIB_CACHE_ADDR"
     FROM v_192216243_f_5_e_7_8_0
     WHERE collection_number = :b1;
```

Getting More Information on a Selected Query in Oracle Trace

There are two convenient ways to obtain additional data for the selected SQL statement:

1. To modify a data view to add or remove statistics or items, select Modify from the Data View menu. You can add or remove statistics in the Items property sheet. These statistics appear as new columns in the data view. The selected query in Figure 13–3 is:

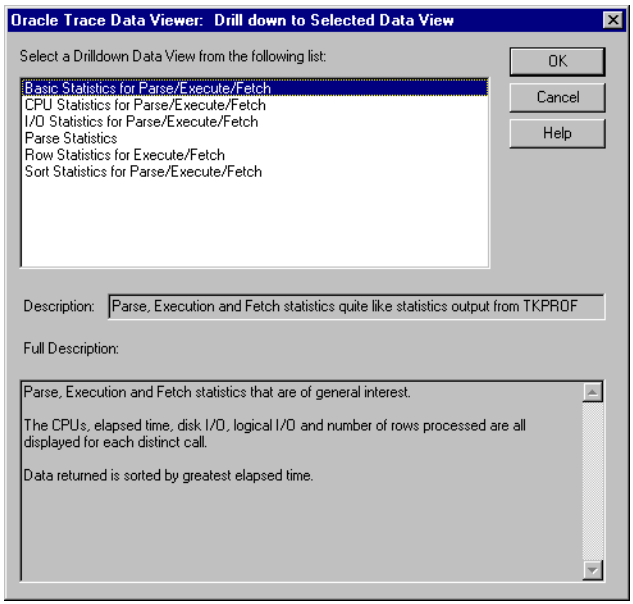
   ```
   SELECT COUNT(DISTINCT WAIT_TIME)
   FROM WAITS
   WHERE COLLECTION_NUMBER = :1;
   ```

   This query counts distinct values in the `wait_time` column of the `waits` table. By modifying the existing data view, you can add other statistics that might be of interest such as "Execute Rows", which is the number of rows processed during execution, or "Execute CPU", which is the number of CPU clock ticks during execution.

   You can also remove existing columns, change the sort order, or change the default number of rows to view. You can save the modified data view. Oracle stores user-defined data views in the Custom data view container following the Data Viewer supplied list of SQL and wait data views.

2. Drill to statistics on all parses, executions, and fetches of the selected query by clicking the Drill icon in the toolbar. The Drill down Data View dialog is displayed in Figure 13–4.

Figure 13–4 Oracle Trace Data Viewer - Drill Down Data View Screen



Drill-down data views show individual statistics for all parses, executions, and fetches.

In Figure 13–4, the "Basic Statistics for Parse/Execute/Fetch" drill-down data view is selected. It displays statistics similar to those from TKPROF.

> **See Also:** *Oracle8i Designing and Tuning for Performance* for more information on TKPROF

Table 13–2 Drill down Data Views

Drill down Name	Sort By	Data Displayed	Description
Basic Statistics for Parse/Execute/Fetch	Greatest elapsed time	For each distinct call: CPUs Elapsed time Disk I/O Logical I/O Number of rows processed	Parse, execution, and fetch statistics that are similar to statistics from TKPROF.
CPU Statistics for Parse/Execute/Fetch	Greatest number of CPUs	CPU total Pagefaults	CPU and pagefault statistics for parses, executions, and fetches of the current query. CPU total is the number of clock ticks in both user and system mode. The clock tick granularity is specific to the operating system on which the database resides.
I/O Statistics for Parse/Execute/Fetch	Greatest number of disk I/Os	Logical and Disk I/O statistics Pagefault I/O (number of hard pagefaults) Input I/O (number of times the file system performed input) Output I/O (number of times the file system performed output)	I/O statistics for parses, executions, and fetches.
Parse Statistics	Greatest elapsed time	Current user identifier Schema identifiers	Parse statistics (for example, whether the current statement was missed in library cache), Oracle optimizer mode, current user identifier, and schema identifier.

Table 13–2 Drill down Data Views

Drill down Name	Sort By	Data Displayed	Description
Row Statistics for Execute/Fetch	Greatest number of rows returned	Number of rows returned Number of rows sorted Number of rows returned during a full table scan	Execution and fetch row statistics.
Sort Statistics for Parse/Execute/Fetch	Greatest elapsed time	Sorts on disk Sorts in memory Number of rows sorted Number of rows returned from a full table scan	Parse, execution, and fetch sort statistics.
Wait Parameters	Wait_time	Description Wait_time P1 P2 P3	Investigating waits can help identify sources of contention. P1, P2, and P3 parameters are values that provide more information about specific wait events. The parameters are foreign keys to views that are wait event dependent. For example, for latch waits, P2 is the latch number that is a foreign key to V$LATCH. The meaning of each parameter is specific to each wait type.

Manually Collecting Oracle Trace Data

Although the Oracle Trace Manager GUI is currently the primary interface to Oracle Trace, you can optionally collect and format Oracle Trace data using one of the following non-GUI mechanisms:

- Oracle Trace Command-Line Interface (CLI)
- Database ORACLE_TRACE_* initialization parameters
- Oracle Trace stored procedures run from PL/SQL

Using the Oracle Trace Command-Line Interface

You can control Oracle Trace server collections with the Oracle Trace CLI (command-line interface). The CLI is invoked by the OTRCCOL executable for the following functions:

- `OTRCCOL START` *`job_id input_parameter_file`*
- `OTRCCOL STOP` *`job_id input_parameter_file`*
- `OTRCCOL CHECK` *`<collection_name>`*
- `OTRCCOL FORMAT` *`input_parameter_file`*
- `OTRCCOL DCF` *`col_name`* `cdf_file`
- `OTRCCOL DFD` *`col_name`* username password service [*`col_id`*]

The `job_id` parameter is a carryover from the Oracle Trace Manager use of the CLI, for managing multiple collection jobs. When the CLI is used directly, the `job_id` value is not used. (Therefore, it can be any positive numeric value greater than 0 and less than 10K). The `job_id` values used for CLI `START` and `STOP` commands do not need to be equal.

The input parameter file contains specific parameter values required for each function, as shown in the following examples. `Col_name` (collection name) and `cdf_file` (collection definition file) are initially defined in the `START` function input parameter file.

The `OTRCCOL START` statement invokes a collection based on parameter values contained in the input parameter file. For example:

```
OTRCCOL START 1234 my_start_input_file
```

where file `my_start_input_file` contains at least the following input parameters:

```
col_name= <collection name>
cdf_file= <collection name>.cdf
dat_file= <collection name>.dat
fdf_file= <facility definition file>.fdf
```

For Oracle database collections, one additional parameter is required:

```
regid= 1 192216243 0 0 5 <database SID>
```

Note: Older CLI versions required this syntax exactly, with no whitespace before '=' but at least some whitespace after '='. This is no longer true: CLI is not whitespace sensitive.

The `regid` parameter record identifies a database by SID where Oracle Trace collection is to be performed. The six elements making up the `regid` parameter record are as follows, in this order:

- flag (this should always be 1 for this usage)
- vendor number (Oracle's vendor number is 192216243)
- `cf_num` (more on this below)
- `cf_val` (more on this below)
- facility number (the Oracle server is facility number 5)
- database SID

The `cf_num` and `cf_val` elements should set to zero in this basic Oracle database collection `regid` above. However, additional `regid` records can be specified to reduce the amount of collected data, and non-zero `cf_num` and `cf_val` can be specified in those situations. In the Oracle server, Oracle Trace cross facility item 6 (`cf_num` = 6) is reserved to record database userID values.

For example, if you provide an additional `regid` record with `cf_num` = 6 and `cf_val` = <*some DB userID*>, then the collection of database event data is limited to only those events performed by that database user.

If you are interested only in collecting database activity for users 23 and 45, then you would provide the following 3 `regid` records:

```
regid= 1 192216243 0  0 5 ORCL
regid= 1 192216243 6 23 5 ORCL
regid= 1 192216243 6 45 5 ORCL
```

Note: Oracle Trace collections for Oracle databases prior to release 8.1.7 collected cross facility item 1 values at the end of each duration event (in other words, parse, execute, fetch, plus logical and physical transaction events), rather than the indicated `cross_fac_6_end` values. However, all cross facility values collected at the start of each duration event are correct. This includes `cross_fac_1_start`, which records an optional application ID if provided, and `cross_fac_6_start`, which records database userID values.

The input parameter file used by the CLI when starting a collection can also contain the following optional parameters, for both database and non-database Oracle Trace collections:

```
prores= <process restriction>
max_cdf= <maximum collection file size>
```

If no process restriction records are specified, then there are no restrictions on which processes can take part in the collection. If process restrictions are used, then one or more process ID (PID) values can be specified, as well as the operating system username for the owner of each process of interest.

The `max_cdf` parameter is often useful, in several different modes of use. This parameter specifies the maximum amount of Oracle Trace data that should be collected, in bytes (in other words, size of the <*collection*>.`dat` file).

A zero value indicates that no limit should be imposed; otherwise, a positive value up to 2 GB can be specified to stop the data collection when that size limit is reached. In addition, a negative value can be specified (but not less than -2 GB), which instructs Oracle Trace to collect data in its "circular data file" mode: when <*collection*>.`dat` reaches magnitude(`max_cdf`), then save that data (and delete any previously saved dat file), and then start collecting to a new <*collection*>.`dat` file. This limits the total amount of disk space used, but allows Oracle Trace data collection to continue until you manually stop collection.

The server event sets that can be used as values for the `fdf_file` parameter are `ORACLE`, `ORACLEC`, `ORACLED`, `ORACLEE`, and `ORACLESM`, plus `CONNECT`, `SQL_ONLY`, `SQL_PLAN`, `SQL_TXN`, `SQLSTATS`, `SQLWAITS`, and `WAITS`.

See Also: Table 13–4 for a description of the server event sets

The `OTRCCOL STOP` statement halts a running collection as follows:

```
OTRCCOL STOP 1234 my_stop_input_file
```

where `my_stop_input_file` contains the collection name and `cdf_file` name.

The `OTRCCOL FORMAT` statement formats the binary collection file to Oracle tables. An example of the `FORMAT` statement is:

```
OTRCCOL FORMAT my_format_input_file
```

where `my_format_input_file` contains the following input parameters:

```
username= <database username>
password= <database password>
```

```
service= <database service name>
cdf_file= <usually same as collection name>.cdf
full_format= <0/1>
```

A `full_format` value of 1 produces a full format. A `full_format` value of 0 produces a partial format, which only formats new data; in other words, data collected since any previous format.

> **See Also:** "Formatting Oracle Trace Data to Oracle Tables" on page 13-31 for more information on formatting part or all of an Oracle Trace collection.

The `OTRCCOL DCF` statement deletes collection files for a specific collection. The `OTRCCOL DFD` statement deletes formatted data from the Oracle Trace formatter tables for a specific collection.

Using Initialization Parameters to Control Oracle Trace

Six Oracle database initialization parameters are set up by default to control Oracle Trace. By logging into the administrator account in the database and executing the `SHOW PARAMETER ORACLE_TRACE` statement, you see the following parameters:

Table 13–3 Oracle Trace Initialization Parameters

Name	Type	Value
ORACLE_TRACE_COLLECTION_NAME	string	[null]
ORACLE_TRACE_COLLECTION_PATH	string	$ORACLE_HOME/otrace/admin/cdf
ORACLE_TRACE_COLLECTION_SIZE	integer	5242880
ORACLE_TRACE_ENABLE	boolean	FALSE
ORACLE_TRACE_FACILITY_NAME	string	oracled
ORACLE_TRACE_FACILITY_PATH	string	$ORACLE_HOME/otrace/admin/cdf

You can modify the Oracle Trace initialization parameters and use them by adding them to the initialization file.

Note: This chapter refers to file path names on UNIX-based systems. For the exact path on other operating systems, see your Oracle platform-specific documentation. A complete discussion of these parameters is provided in *Oracle8i Reference.*

Enabling Oracle Trace Collections

The `ORACLE_TRACE_ENABLE` database initialization parameter is `false` by default. This disables any collection of Oracle Trace data for that server, regardless of the mechanism used to control the collection.

Setting `ORACLE_TRACE_ENABLE` to `true` in <*`DBinit`*>.`ora` enables Oracle Trace collections for the server, but it does not necessarily start a collection when the database instance is started. If the database parameters alone are to be used to start an Oracle Trace collection of database event data, then all 6 `ORACLE_TRACE_*` parameters must be specified, or have non-null values by default. Typically, this means that both `ORACLE_TRACE_ENABLE` must be set to `true` and a non-null `ORACLE_TRACE_COLLECTION_NAME` must be provided (up to 16 characters in length).

Note: The collection name is also used to form the <*`collection name`*>.`cdf` and .`dat` binary file names, so 8.3 file naming conventions may apply on some platforms. 8.3 file naming means systems where filenames are restricted to 8 or fewer characters, plus a file extension of 3 or fewer characters.

This method for controlling the Oracle Trace collection is rather inflexible: the collection name cannot be changed without performing a database shutdown. (For Oracle releases prior to 8.1.7, the collection can only be stopped by doing a shutdown, then setting `ORACLE_TRACE_ENABLE` = `false` before restarting.) However, with `ORACLE_TRACE_ENABLE` = `true` but `ORACLE_TRACE_COLLECTION_NAME` = "" [in other words, empty name string], Oracle Trace collections of database event data can be performed using one of the other collection control mechanisms; for example, the Oracle Trace CLI or the Oracle Trace Manger GUI. These other mechanisms are more flexible than the database initialization parameters. They are generally preferred over using parameters for collection control.

`ORACLE_TRACE_ENABLE` is a dynamic parameter, so it can be set to `true` or `false` while the database is running. This can be done for the current database session or

for all sessions (including future ones), using ALTER SESSION or ALTER SYSTEM statements. When the database is subsequently shut down and then restarted, the <*DBinit*>.ora setting for ORACLE_TRACE_ENABLE is again used to initially enable or disable Oracle Trace collection of database event data.

Determining the Event Set that Oracle Trace Collects

The ORACLE_TRACE_FACILITY_NAME database initialization parameter specifies the event set that Oracle Trace collects, if the database parameters are used to control data collection. The default for this parameter is ORACLED (in other words, Oracle "default" event set).

> **Note:** The ORACLE_TRACE_FACILITY_NAME parameter does not use a file extension. So, the .fdf extension should not be specified as part of this parameter.

Table 13–4 Server Event Set File Names

Event Set File Name (.fdf)	Description
CONNECT	CONNECT_DISCONNECT event set. Collects statistics about connects to the database and disconnects from the database.
ORACLE	ALL event set. Collects all statistics for the Oracle Server including wait events.
ORACLEC	CACHEIO event set. Collects caching statistics for buffer cache I/O.
ORACLED	Oracle Server DEFAULT event set. Collects statistics for the Oracle Server.
ORACLEE	EXPERT event set. Collects statistics for the Oracle Expert application.
ORACLESM	SUMMARY event set. Collects workload statistics for the Summary Advisor application.

Table 13–4 Server Event Set File Names (Cont.)

Event Set File Name (.fdf)	Description
SQL_ONLY	SQL_TEXT_ONLY event set. Collects statistics about connects to the database, disconnects from the database, and SQL text.
SQL_PLAN	SQL_STATS_AND_PLAN event set. Collect statistics about connects to the database, disconnects from the database, SQL statistics, SQL text, and row source (EXPLAIN PLAN).
SQLSTATS	SQL_AND_STATS event set. Collects SQL text and statistics only.
SQL_TXN	SQL_TXNS_AND_STATS event set. Collects statistics about connects to the database, disconnects from the database, transactions, SQL text and statistics, and row source (EXPLAIN PLAN).
SQLWAITS	SQL_AND_WAIT_STATS event set. Collects statistics about connects to the database, disconnects from the database, row source (EXPLAIN PLAN), SQL text and statistics, and wait events.
WAITS	WAIT_EVENTS event set. Collects statistics about connects to the database, disconnects from the database, and wait events.

After it is restarted, if the database does not begin collecting data, then check the following:

- The event set file, identified by ORACLE_TRACE_FACILITY_NAME, with an .fdf extension, should be in the directory specified by the ORACLE_TRACE_FACILITY_PATH initialization parameter. The exact directory that this parameter specifies is platform-specific.
- The following files should exist in the Oracle Trace admin directory: COLLECT.DAT, FACILITY.DAT (or PROCESS.DAT for Oracle 7.3), and REGID.DAT. If they do not, then run the OTRCCREF executable to create or recreate them.

- The Oracle Trace parameters should be set to the values that you changed in the initialization file. Use Instance Manager to identify Oracle Trace parameter settings.
- Look for an EPC_ERROR.LOG file to see more information about why a collection failed. Oracle Trace creates the EPC_ERROR.LOG file in the current default directory of the Oracle Intelligent Agent when it runs the Oracle Trace Collection Services OTRCCOL image. Depending on whether you are running Oracle Trace from the Oracle Trace Manager or from the command-line interface, you can find the EPC_ERROR.LOG file in one of the following locations:
  - $ORACLE_HOME or $ORACLE_HOME/network/agent on UNIX
  - %ORACLE_HOME%\network\agent or %ORACLE_HOME%\net80\agent on NT
  - $ORACLE_HOME\rdbmsnn on NT or $ORACLE_HOME\rdbms on UNIX
  - In the current working directory, if you are using the command-line interface
  - To find the EPC_ERROR.LOG file on UNIX, change directories to the $ORACLE_HOME directory and run the statement:

    ```
    find . -name EPC_ERROR.LOG -print .
    ```

    Note: On UNIX, the EPC_ERROR.LOG file name is case sensitive and is in uppercase.

  - Look for *.trc files in the directory specified by the USER_DUMP_DEST initialization parameter. Searching for "epc" in the *.trc files might give errors. These errors and their descriptions are located in the $ORACLE_HOME/otrace/include/epc.h file.

Controlling Oracle Trace Collections from PL/SQL

Oracle provides an additional Oracle Trace library that allows control of both database and non-database Oracle Trace collections from PL/SQL.

Note: Older versions of the Oracle server, back to release 7.3.3, provided other stored procedures to start and stop Oracle Trace database collections, but with significant limitations. For example, only database sessions already active when the collection was started participated in the collection: no database event data was collected for sessions that began after the Oracle Trace collection started.

As of Oracle8, these limitations were eliminated for database collections started via the Oracle Trace Manager or CLI. However, these limitations still applied to database collections controlled via the older Oracle7 stored procedures. The new procedures provided with Oracle Trace 8.1.7 remove these limitations, permitting the same level of collection control as the Oracle Trace CLI, and for both database and non-database collections.

Both the name and the location of this new library are platform-dependent. On Unix platforms, the library is $ORACLE_HOME/lib/libtracepls8.so

On Win32 platforms (for example, Windows NT), the library is %ORACLE_HOME%\bin\oratracepls8.dll

The otrace/admin directory contains two new SQL scripts that can be used to define a database LIBRARY object for this library, and to define the procedures that can be used to call out to the library from PL/SQL:

- `OTRCPLSLIB.SQL`
- `OTRCPLSCMD.SQL`

Note: By default, `OTRCPLSLIB.SQL` grants access to the LIBRARY to all database users. You can edit this SQL script to restrict access as appropriate.

In addition, the otrace/demo directory contains several SQL scripts showing PL/SQL examples that start, stop, and then format an Oracle Trace collection. These are:

- `OTRCPLSSC.SQL`
- `OTRCPLSCC.SQL`

- OTRCPLSFC.SQL

In the "start collection" example script OTRCPLSSC.SQL, the regid_list contains only a single element: "1 192216243 0 0 5 ORCL". The inner double quotes are required to form a single regid string from its six components. These components are the following, in the order shown:

- flag (this should always be 1 for this usage)
- vendor number (Oracle's vendor number is 192216243)
- cf_num (see below)
- cf_val (see below)
- facility number (Oracle server is facility number 5)
- database SID

For an Oracle Trace database collection, a regid string like this example is required, basically to identify the database SID and to specify that you are collecting for an Oracle server. The cf_num and cf_val should be zero in this basic regid record.

Additional regid records can be specified in order to reduce the amount of collected data. This is when the cf_num and cf_val items are used. In the Oracle server, Oracle Trace cross facility item 6 (cf_num = 6) is reserved to record database userID values. So, if you provide an additional regid record with cf_num = 6 and a cf_val = *<some DB userID>*, then the collection of database event data is limited to only those events performed by that database user. For example, if you are only interested in collecting database activity for users 23 and 45, then the regid_list consists of three records:

```
regid_list   VARCHAR2(256) := '"1 192216243 0  0 5 ORCL",
                               "1 192216243 6 23 5 ORCL",
                               "1 192216243 6 45 5 ORCL"';
```

Note: For better readability, the layout of this regid string has been simplified.

Similarly, the fdf_list argument could specify the name of a single .fdf file (facility definition file). Typically, this is the case. However, more than one .fdf could be specified in fdf_list if multiple facilities are involved in the collection. Of course, only one .fdf can be specified for any given facility; for example, the database.

On the other hand, the process restriction list `prores_list` can be empty. This indicates that there are to be no restrictions on which processes can take part in the collection. If process restrictions are used, then one or more process ID (PID) values can be specified, as well as the operating system username for the owner of each process.

Other arguments in the "start collection" example in `OTRCPLSSC.SQL` are single numeric or string values, as shown. For example, the collection name and maximum collection data file size specified by the `col_name` and `maxsize` variables, respectively.

Accessing Oracle Trace Collection Results

Running an Oracle Trace collection produces the following collection files:

- `<`*`collection name`*`>.cdf` is the binary Oracle Trace collection definition file for the collection. It describes what is to be collected.
- `<`*`collection name`*`>.dat` is the output file containing the collected Oracle Trace event data in binary form.

You can access the Oracle Trace data in the collection files in the following ways:

- You can create Oracle Trace reports from the binary file.
- The data can be formatted to Oracle tables for Data Viewer, SQL access, and reporting.

Formatting Oracle Trace Data to Oracle Tables

You can format Oracle Trace binary collection data to Oracle database tables, and you can then access this formatted data using SQL or other tools, such as the Oracle Trace Data Viewer. The Oracle Trace format produces a separate table for each event type collected; for example, a parse event table is created to store data for all database parse events that were recorded during a server collection.

> **Note:** For Oracle server releases 7.3.4 and later, the Oracle Trace formatter automatically creates the formatter tables as needed.

Use the following syntax to format an Oracle Trace collection:

```
OTRCFMT [optional parameters] <collection_name>.cdf
[user/password@database]
```

If you omit `user/password@database`, then Oracle prompts you for this information.

Oracle Trace allows data to be formatted while a collection is occurring. By default, Oracle Trace formats only the portion of the collection that has not been formatted previously. If you want to reformat the entire collection file, then use the optional parameter `-f`.

Oracle Trace provides several SQL scripts that you can use to access the server event tables.

> **See Also:** *Oracle Trace User's Guide* for more information on server event tables and scripts for accessing event data and improving event table performance

Running the Oracle Trace Statistics Reporting Utility

The Oracle Trace statistics reporting utility displays statistics for all items associated with each occurrence of a server event. These reports can be quite large. You can control the report output by using statement parameters. Use the following statement and optional parameters to produce a report:

```
OTRCREP  [optional parameters]  collection_name.CDF
```

First, you might want to run a report called `PROCESS.txt`. You can produce this report to provide a listing of specific process identifiers for which you want to run another report.

You can manipulate the output of the Oracle Trace reporting utility by using the following optional report parameters:

`output_path`	Specifies a full output path for the report files. If this path is not specified, then the files are placed in the current directory.
`-p[<pid>]`	Organizes event data by process. If you specify a process ID (PID), then you have one file with all the events generated by that process in chronological order. If you omit the PID, then you have one file for each process that participated in the collection. The output files are named `<collection_name>_Ppid.txt`.
`-P`	Produces a report file name `PROCESS.txt` that lists all processes that participated in the collection. It does not include event data. You can produce this report first to determine the specific processes for which you want to produce more detailed reports.
`-w#`	Sets report width, such as -w132. The default is 80 characters.
`-l#`	Sets the number of report lines per page. The default is 63 lines per page.
`-h`	Suppresses all event and item report headers, producing a shorter report.
`-s`	Used with Net8 data only, this option creates a file similar to the SQL*Net Tracing file.
`-a`	Creates a report containing all the events for all products, in the order they occur in the data collection (`.dat`) file.

Index

Symbols

? operator, 2-36

A

ADD_INDEX new procedure, 2-38
ADD_STOPWORD procedure
  updated syntax, 2-47
ALTER DATABASE CHARACTER SET
    statement, 3-7
ALTER DATABASE NATIONAL CHARACTER
    SET statement, 3-7
ALTER DATABASE statement
  CREATE STANDBY CONTROLFILE
      clause, 12-22
ALTER INDEX, 2-20
  interMedia Text examples, 2-22
archive destinations
  multiple, 12-14
archived redo logs
  and multiple destinations, 12-14
array parameter
  Character Set Scanner Utility, 3-16
Average Elapsed Time data view, 13-10

B

backup command (RMAN), 7-42
backups
  standby database
    archived redo logs, 7-29
    interpreting RC_ARCHIVED_LOG
        view, 7-26
    overview of RMAN, 7-25
    restrictions, 7-26
    using RMAN, 7-25, 7-31, 7-32
  testing RMAN, 7-33
Basic Statistics for Parse/Execute/Fetch drilldown
    data view, 13-19
binary files
  formatting using Oracle Trace, 13-5
BLANK_TRIMMING initialization parameter, 3-4
boundaries parameter
  Character Set Scanner Utility, 3-16

C

capture parameter
  Character Set Scanner Utility, 3-17
catalog index
  about, 2-3
catproc.sql script, 6-2
CATSEARCH operator, 2-13
character set errors
  troubleshooting with RMAN, 7-38
character set migration
  data scanning, 3-7
Character Set Scanner
  scan modes, 3-11
Character Set Scanner Utility, 3-1, 3-9, 3-17
  array parameter, 3-16
  boundaries parameter, 3-16
  capture parameter, 3-17
  compatibility, 3-13
  feedback parameter, 3-17
  fromnchar parameter, 3-18
  full parameter, 3-18

help parameter, 3-18
invoking, 3-13
lastrpt parameter, 3-19
maxblocks parameter, 3-19
online help, 3-14
parameter file, 3-15
parameters, 3-16
parfile parameter, 3-20
scanner messages, 3-38
scanner parameters, 3-16
scanner tables, 3-36
suppress parameter, 3-20
table parameter, 3-21
tochar parameter, 3-21
tonchar parameter, 3-21
user parameter, 3-21
userid parameter, 3-22
character sets
choosing, 3-2
data loss, 3-4
migrating, 3-2
migration, 3-2
choosing character sets, 3-2
collections, 13-4, 13-25
commands, Recovery Manager
backup, 7-42
changes to, 7-41
copy, 7-45
debug, 7-46
duplicate, 7-2, 7-47
set, 7-49
compatibility
Recovery Manager, 7-39
COMPATIBLE initialization parameter, 12-16
snapshot base tables, 9-7
configure compatible command (RMAN)
obsolete in 8.1.7, 7-38
configuring
directory access, 5-2
domain hints to Oracle Names servers, 5-14
initialization parameter file, 12-14, 12-16, 12-24, 12-26
instance role, 5-11
listener.ora file, 12-12
network files, 12-12
session data unit (SDU), 5-9
tnsnames.ora file, 12-12
CONTEXT indextype, 2-16
control files
creating, 12-7, 12-22
snapshot
setting default location, 7-33
CONTROL_FILES initialization parameter, 12-9, 12-16
copy command (RMAN), 7-45
CPU Statistics data view, 13-12
CPU Statistics for Parse/Execute/Fetch drilldown data view, 13-19
CREATE INDEX, 2-16
CREATE PACKAGE BODY command, 6-3
CREATE PACKAGE command, 6-3
CREATE_INDEX_SET new procedure, 2-41
CREATE_STOPLIST procedure
updated syntax, 2-46
CREATE_TRANSLATION new procedure, 2-49
creating
control files, 12-22
packages, 6-3
standby databases, 12-6
crosschecking
catalog not required, 7-37
on multiple channels, 7-34
CSM$COLUMNS parameter, 3-34
CSM$ERRORS parameter, 3-35
CSM$TABLES parameter, 3-34
CSMIG user, 3-12
CSMINST.SQL script, 3-12
running, 3-13
CTX_DDL.ADD_INDEX, 2-38
CTX_DDL.ADD_STOPWORD, 2-47
CTX_DDL.CREATE_INDEX_SET, 2-41
CTX_DDL.CREATE_STOPLIST, 2-46
CTX_DDL.DROP_INDEX_SET, 2-42
CTX_DOC.GIST, 2-59
CTX_DOC.THEMES, 2-56
CTX_THES.CREATE_TRANSLATION, 2-49
CTX_THES.DROP_TRANSLATION, 2-50
CTX_THES.UPDATE_TRANSLATION, 2-52
CTXCAT indextype, 2-16
about, 2-3

CTXKBTC knowledge base compiler
updated syntax, 2-12

D

data conversion
database character set, 3-7
data expansion during character set migration, 3-2
data inconsistencies
causing data loss, 3-5
data loss
caused by data inconsistencies, 3-5
during character set migration, 3-4
from mixed character sets, 3-6
data scanning
character set migration, 3-7
data truncation, 3-2
restrictions, 3-3
data views in Oracle Trace, 13-7
Average Elapsed Time, 13-10
CPU Statistics, 13-12
Disk Reads, 13-9
Disk Reads/Execution Ratio, 13-9
Disk Reads/Logical Reads Ratio, 13-10
Disk Reads/Rows Fetched Ratio, 13-9
Execute Elapsed Time, 13-11
Fetch Elapsed Time, 13-11
Logical Reads, 13-9
Logical Reads/Rows Fetched Ratio, 13-9
Number of Rows Processed, 13-12
Parse Elapsed Time, 13-11
Parse/Execution Ratio, 13-10
Re-Parse Frequency, 13-10
Rows Fetched/Fetch Count Ratio, 13-12
Rows Sorted, 13-12
Sorts in Memory, 13-12
Sorts on Disk, 13-12
Total Elapsed Time, 13-11
Waits by Average Wait Time, 13-13
Waits by Event Frequency, 13-13
Waits by Total Wait Time, 13-13
database character set
conversion, 3-7
database character set migration, 3-6
database conections
RMAN and Oracle Parallel Server, 4-2
Database Connection event, 13-5
Database Scan Summary Report, 3-25, 3-26
databases
standby
creating, procedures for, 12-6
DB_FILE_NAME_CONVERT initialization parameter, 12-10, 12-16
DB_FILES initialization parameter, 12-16
DB_NAME initialization parameter, 12-16
DBA Studio
Replication Management tool included, 9-15
DBMS_JOB package, 6-5
DBMS_REPCAT_INSTANTIATE package
INSTANTIATE_OFFLINE_REPAPI function, 10-2
DBMS_REPCAT_RGT package
INSTANTIATE_OFFLINE_REPAPI function, 10-6
debug command (RMAN), 7-46
deleting
using RMAN
catalog not required, 7-37
deleting backups
on multiple channels, 7-34
DESDecrypt procedure, 6-70, 6-74
DESEncrypt procedure, 6-69, 6-73
detail report in Oracle Trace, 13-6
details property sheet in Oracle Trace, 13-15
directory access configuration, 5-2
ldapmodify tool, 5-8
ldap.ora file, 5-3
Net8 Configuration Assistant options, 5-5
Oracle Context, 5-3
OracleDBCreators group, 5-3, 5-6
OracleNetAdmins group, 5-3, 5-6, 5-8
OracleSecurityAdmins group, 5-3, 5-6
performing after installation, 5-5
performing during installation, 5-2
client, 5-4
server custom, 5-2
Disk Reads data view, 13-9
Disk Reads/Execution Ratio data view, 13-9
Disk Reads/Logical Reads Ratio data view, 13-10
Disk Reads/Rows Fetched Ratio data view, 13-9

document formats
supported, 2-9
document tokens
about, 2-8
DOMAIN_HINT command, 5-14
drilldown data views in Oracle Trace, 13-17
Basic Statistics for Parse/Execute/Fetch, 13-19
CPU Statistics for Parse/Execute/Fetch, 13-19
Parse Statistics, 13-19
Row Statistics for Execute/Fetch, 13-20
DROP_INDEX_SET new procedure, 2-42
DROP_TRANSLATION new procedure, 2-50
duplicate command (RMAN), 7-47
duplicate databases
creating
on a remote host with same directory structure, 7-13
on local host, 7-20
on remote host with different directory structure, 7-15
duration events in Oracle Trace, 13-5

E

ENABLE option
LOG_ARCHIVE_DEST_*n* initialization parameter, 12-15
events in Oracle Trace, 13-5
Execute Elapsed Time data view, 13-11

F

fast refresh
avoiding problems, 9-6
feedback parameter
Character Set Scanner Utility, 3-17
Fetch Elapsed Time data view, 13-11
filtering documents
interMedia Text, 2-9
FORMAT statement
in Oracle Trace, 13-21
formatter tables
in Oracle Trace, 13-5
French knowledge base, 2-10
fromchar parameter, 3-17
Character Set Scanner Utility, 3-17
fromnchar parameter
Character Set Scanner Utility, 3-18
full parameter
Character Set Scanner Utility, 3-18
fuzzy operator, 2-36
about, 2-9

G

gap sequences
applying the logs to standby database, 12-20, 12-27
copying the logs in, 12-19
identifying the logs in, 12-18
gist
generating, 2-59
GIST procedure
example, 2-62
updated syntax, 2-59

H

help parameter
Character Set Scanner Utility, 3-18

I

index
creating interMedia Text, 2-16
index maintenance
interMedia Text, 2-20
index optimization, 2-21
Individual Exception Report, 3-25, 3-32
initialization parameter files
configuring, 12-14, 12-16, 12-24, 12-26
initialization parameters
BLANK_TRIMMING, 3-4
COMPATIBLE, 12-16
CONTROL_FILES, 12-9, 12-16
DB_FILE_NAME_CONVERT, 12-10, 12-16
DB_FILES, 12-16
DB_NAME, 12-16
in Oracle Trace, 13-24
LOCK_NAME_SPACE, 12-9, 12-16

LOG_ARCHIVE_DEST_*n*, 12-15, 12-16
LOG_ARCHIVE_FORMAT, 12-15
LOG_ARCHIVE_START, 12-25
LOG_FILE_NAME_CONVERT, 12-10, 12-25
ORACLE_TRACE_ENABLE, 8-2
STANDBY_ARCHIVE_DEST, 12-16
Inso filter
newly supported formats, 2-9
supported platforms, 2-10
installation
directory access configuration, 5-2
instance role configuration, 5-11
connections in TAF, 5-13
connections to primary and secondary instances, 5-11
connections to specific instances, 5-12
INSTANCE_ROLE
use of in secondary instance connections, 4-2
INSTANCE_ROLE parameter, 5-11
interconnects
and the OPS_INTERCONNECTS parameter, 4-2
interMedia Text
new features and enhancements, 2-3
I/O
Statistics for Parse/Execute/Fetch view, 13-19

K

keyless index
about, 2-5
knowledge base
French and English, 2-10
supported character set, 2-11
user-defined, 2-10

L

lastrpt parameter
Character Set Scanner Utility, 3-19
ldapmodify tool, 5-8
ldap.ora file, 5-3
LISTENER.ORA file
configuring, 12-12
format, 12-13, 12-23
LOCK_NAME_SPACE initialization parameter, 12-9, 12-16
LOG_ARCHIVE_DEST_*n* initialization parameter, 12-15, 12-16
ENABLE option, 12-15
OPTIONAL option, 12-15
REOPEN option, 12-15
SERVICE option, 12-15
specifying destinations using, 12-15
LOG_ARCHIVE_FORMAT initialization parameter, 12-15
LOG_ARCHIVE_START initialization parameter, 12-25
LOG_FILE_NAME_CONVERT initialization parameter, 12-10, 12-25
Logical Reads data view, 13-9
Logical Reads/Rows Fetched Ratio data view, 13-9

M

master sites
scheduled links for guidelines, 9-9
scheduled purge for guidelines, 9-12
maxblocks parameter
Character Set Scanner Utility, 3-19
migrating character sets, 3-2
migration
database character set, 3-6
post-migration actions, 6-66
mixed character sets
causing data loss, 3-6
MULTI_COLUMN_DATASTORE, 2-24
multi-column datastore
about, 2-6
multi-language stoplist
about, 2-7
multimaster replication, 6-1

N

NAMES.DOMAIN_HINTS parameter, 5-14
Net8 Configuration Assistant
configuring directory access, 5-2

OracleDBCreators group, 5-3, 5-6
OracleNetAdmins group, 5-3, 5-6
OracleSecurityAdmins group, 5-3, 5-6
network files
configuring, 12-12
Number of Rows Processed data view, 13-12
numthemes new parameter
CTX_DOC.GIST, 2-62
CTX_DOC.THEMES, 2-57

O

operator
fuzzy, 2-36
OPS_INTERCONNECTS
for Oracle Parallel Server, 4-2
optimization
token, 2-22
optimizing index, 2-21
OPTIONAL option
LOG_ARCHIVE_DEST_*n* initialization parameter, 12-15
OR REPLACE clause
for creating packages, 6-3
Oracle Context creation, 5-3
Oracle Names Control utility
DOMAIN_HINT command, 5-14
REGISTER command, 5-17
REGISTER_NS command, 5-17
UNREGISTER command, 5-17
UNREGISTER_NS command, 5-19
Oracle Parallel Server
during database character set migration, 3-8
using RMAN, 7-36
Oracle Server
events, 13-5
Oracle Trace
accessing collected data, 13-5
binary files, 13-5
collection results, 13-31
collections, 13-4, 13-25
command-line interface, 13-20
data views, 13-7
Average Elapsed Time, 13-10
CPU Statistics, 13-12
Disk Reads, 13-9
Disk Reads/Execution Ratio, 13-9
Disk Reads/Logical Reads Ratio, 13-10
Disk Reads/Rows Fetched Ratio, 13-9
Execute Elapsed Time, 13-11
Fetch Elapsed Time, 13-11
Logical Reads, 13-9
Logical Reads/Rows Fetched Ratio, 13-9
Number of Rows Processed, 13-12
Parse Elapsed Time, 13-11
Parse/Execution Ratio, 13-10
Re-Parse Frequency, 13-10
Rows Fetched/Fetch Count Ratio, 13-12
Rows Sorted, 13-12
Sorts in Memory, 13-12
Sorts on Disk, 13-12
Total Elapsed Time, 13-11
Waits by Average Wait Time, 13-13
Waits by Event Frequency, 13-13
Waits by Total Wait Time, 13-13
deleting files, 13-24
details property sheet, 13-15
drilldown data views, 13-17, 13-19
Basic Statistics for Parse/Execute/Fetch view, 13-19
CPU Statistics for Parse/Execute/Fetch view, 13-19
Parse Statistics view, 13-19
Row Statistics for Execute/Fetch view, 13-20
duration events, 13-5
events, 13-5
FORMAT statement, 13-21
formatter tables, 13-5
Oracle Trace Data Viewer, 13-6
parameters, 13-24
point events, 13-5
predefined data views, 13-7
reporting utility, 13-6, 13-32
SQL statement property sheet, 13-15
START statement, 13-21
STOP statement, 13-21, 13-23
using to collect workload data, 13-3
viewing data, 13-13
Oracle Trace Data Viewer, 13-6
Oracle Trace Manager, 13-4

used for formatting collections, 13-5
ORACLE_TRACE_COLLECTION_NAME
initialization parameter, 13-24
ORACLE_TRACE_COLLECTION_PATH
initialization parameter, 13-24
ORACLE_TRACE_COLLECTION_SIZE
initialization parameter, 13-24
ORACLE_TRACE_ENABLE initialization
parameter, 8-2, 13-24
ORACLE_TRACE_FACILITY_NAME initialization
parameter, 13-24, 13-26
ORACLE_TRACE_FACILITY_PATH initialization
parameter, 13-24
OracleDBCreators group, 5-3, 5-6
OracleNetAdmins group, 5-3, 5-6, 5-8
OracleSecurityAdmins group, 5-3, 5-6

P

package overview, 6-2
packages
creating, 6-3
referencing, 6-5
parallel propagation
replication, 9-14
parameters
CSM$COLUMNS, 3-34
CSM$ERRORS, 3-35
CSM$TABLES, 3-34
parfile parameter
Character Set Scanner Utility, 3-20
Parse Elapsed Time data view, 13-11
Parse Statistics drilldown data view, 13-19
Parse/Execution Ratio data view, 13-10
point events in Oracle Trace, 13-5
prefix indexing, 2-7, 2-33
primary and secondary instances, 5-11
procedure filter, 2-7, 2-28

R

raw partitions
tablespace size requirements for Oracle Parallel
Server, 4-2
RECOVER statement
specifying the TIMEOUT option, 12-21
recovery catalog
crosschecking, 7-37
deleting, 7-37
views, 7-50
Recovery Manager
and Oracle Parallel Server, 4-2
backups
standby database, 7-32
testing, 7-33
character set errors
troubleshooting, 7-38
commands
changes to, 7-41
duplicate, 7-2
compatibility, 7-39
snapshot control file
setting default location, 7-33
standby database
about creating, 7-2
backing up, 7-25
backing up archived logs, 7-29
backing up overview, 7-25
backing up the database, 7-31
backup restrictions, 7-26
copying standby control file, 7-6
creating, 7-9
creating standby control file, 7-3
creating standby control file using RMAN
backup command, 7-4
creating standby control file using RMAN
copy command, 7-5
creating standby control file using SQL, 7-5
creating using image copies, 7-20
creation overview, 7-9
DB_FILE_NAME_CONVERT initialization
parameter, 7-8
interpreting the RC_ARCHIVED_LOG
view, 7-26
LOG_FILE_NAME_CONVERT initialization
parameter, 7-8
naming online redo logs, 7-8
naming standby datafiles, 7-6
preparing using RMAN, 7-2
restrictions when creating, 7-12

starting RMAN and standby instance, 7-12
views, 7-50
REGISTER command, 5-17
REGISTER_NS command, 5-17
REOPEN option
LOG_ARCHIVE_DEST_*n* initialization parameter, 12-15
Re-Parse Frequency data view, 13-10
replication, 9-1
adding snapshot sites
avoiding problems, 9-6
COMPATIBLE initialization parameter
snapshot base tables, 9-7
continuous push
scheduling, 9-11, 9-13
creating an environment, 9-40
deferred transactions
continuous purge, 9-13
continuous push, 9-11
periodic purge, 9-12
periodic push, 9-10
deployment templates
offline instantiation, 10-2, 10-6
documentation
additions, 9-5, 10-10
updates, 9-7
flowchart for creating environment, 9-40
generated files
location, 9-5
instantiate offline
INSTANTIATE_OFFLINE_REPAPI function, 10-2, 10-6
master sites
switching, 10-10
multimaster, 6-1
new features, 9-2
offline instantiation
INSTANTIATE_OFFLINE_REPAPI function, 10-2, 10-6
RepAPI clients, 10-2, 10-6
procedural replication
snapshot sites, 9-5
propagation
serial and parallel, 9-14
quiesce reduced, 9-2
replication management API, 10-1
Replication Management tool, 9-15
Copy Template Wizard, 9-38
Deployment Template Wizard, 9-34
first login, 9-17
interface, 9-18
navigator pane, 9-19
Offline Instantiation Wizard, 9-36
right pane, 9-24
Setup Wizard, 9-30
Snapshot Group Wizard, 9-33
Topology tab, 9-26
usage scenarios, 9-16
wizards, 9-29
scheduled links
continuous push, 9-11, 9-13
guidelines for, 9-9
parallel propagation, 9-14
serial propagation, 9-14
See Also multimaster replication
sequences
not supported, 9-7
snapshot sites
procedural replication, 9-5
snapshots
base table, 9-7
UTL_FILE_DIR initialization parameter, 9-5
restrictions
data truncation, 3-3
passwords, 3-3
space padding during export, 3-4
usernames, 3-3
RMAN. *See* Recovery Manager
Row Statistics for Execute/Fetch drilldown data views, 13-20
Rows Fetched/Fetch Count Ratio data view, 13-12
Rows Sorted data view, 13-12
RowSource event, 13-5

S

scan modes
Character Set Scanner Utility, 3-11
database character sets
full database scan, 3-11

single table scan, 3-11
user tables scan, 3-11
SDU parameter, 5-9
secondary instances
connections to in Oracle Parallel Server, 4-2
security with datastores, 2-6
sequences
replication not supported, 9-7
SERVICE option
LOG_ARCHIVE_DEST_*n* initialization parameter, 12-15
session data unit (SDU) configuration, 5-9
set command (RMAN), 7-49
snapshot sites
adding
avoiding problems, 9-6
scheduled links for
guidelines, 9-9
scheduled purge for
guidelines, 9-12
snapshots
base table, 9-7
Sorts in Memory data view, 13-12
Sorts on Disk data view, 13-12
space padding
during export, 3-4
SQL Parse event, 13-5
SQL statement property sheet in Oracle Trace, 13-15
SQL statements
ALTER INDEX, 2-20
CREATE INDEX, 2-16
SQL*Plus
creating a sequence, 6-5
standby databases
about creating using RMAN, 7-2
backing up using RMAN, 7-25, 7-31, 7-32
guidelines for backing up logs, 7-29
interpreting the RC_ARCHIVED_LOG view, 7-26
overview, 7-25
restrictions, 7-26
control files
creating, 12-7, 12-22
copying control file using RMAN, 7-6
copying standby control file, 7-6
creating
on remote host, 12-21
on same host, 12-9
procedures, 12-6
creating control file
using RMAN, 7-3, 7-4, 7-5
using SQL, 7-5
creating using RMAN, 7-9
overview, 7-9
restrictions, 7-12
using image copies, 7-20
DB_FILE_NAME_CONVERT initialization parameter, 7-8
initialization parameters, 12-16
LOG_FILE_NAME_CONVERT initialization parameter, 7-8
naming datafiles using RMAN, 7-6
naming online redo logs using RMAN, 7-8
preparing using RMAN, 7-2
set auxname command (RMAN), 7-7, 7-8
set newname command (RMAN), 7-8
starting RMAN and standby instance, 7-12
STANDBY_ARCHIVE_DEST initialization parameter, 12-16
START statement in Oracle Trace, 13-21
STOP statement in Oracle Trace, 13-21, 13-23
stoplist
creating, 2-46
stopword
defining, 2-47
subsets
and supersets, 3-45
supersets
and subsets, 3-45
suppress parameter
Character Set Scanner Utility, 3-20

T

table parameter
Character Set Scanner Utility, 3-21
tables
formatter in Oracle Trace, 13-5
tablespace size requirements

for raw partitions in Oracle Parallel Server, 4-2
target database connections
RMAN and Oracle Parallel Server, 4-2
testing RMAN backups, 7-33
theme functionality
supported languages, 2-10
theme summary
generating, 2-59
generating top n, 2-62
themes
generating for document, 2-56
obtaining top n, 2-58
THEMES procedure
updated syntax, 2-56
thesaurus translation
about, 2-8
TNSNAMES.ORA file
configuring, 12-12
format, 12-12, 12-23
tochar parameter
Character Set Scanner Utility, 3-21
token index optimization
about, 2-8
example, 2-22
interMedia Text, 2-21
tonchar parameter
Character Set Scanner Utility, 3-21
Total Elapsed Time data view, 13-11
translations
adding to thesaurus, 2-49
dropping, 2-50
updating, 2-52
Transparent Application Failover (TAF) with instance role, 5-13
troubleshooting RMAN
character set errors, 7-38

U

UNREGISTER command, 5-17
UNREGISTER_NS command, 5-19
UPDATE_TRANSLATION new procedure, 2-52
upgrading
post-upgrade actions, 6-66
URL datastore
enhancements, 2-6
user parameter
Character Set Scanner Utility, 3-21
userid parameter
Character Set Scanner Utility, 3-22
UTL_FILE_DIR initialization parameter, 9-5

V

V$ARCHIVED_LOG view, 8-2, 12-28
V$BACKUP_DATAFILE view, 8-3
V$BACKUP_SET view, 8-4
V$DATABASE view, 12-7
V$DATAFILE view, 12-22
V$DATAFILE_COPY view, 8-5
V$LOG_HISTORY view, 12-19, 12-27
V$PROXY_DATAFILE view, 8-7
V$SQL_SHARED_CURSOR view, 8-8
views
recovery catalog, 7-50
V$ARCHIVED_LOG, 12-28
V$DATABASE, 12-7
V$DATAFILE, 12-22
V$LOG_HISTORY, 12-19, 12-27

W

Waits by Average Wait Time data view, 13-13
Waits by Event Frequency data view, 13-13
Waits by Total Wait Time data view, 13-13