Oracle® Database

Backup and Recovery User's Guide

11*g* Release 1 (11.1)

B28270-03

August 2008

A guide to backup and recovery of Oracle databases, including RMAN backup and recovery, RMAN data transfer, Oracle Flashback Technology, and user-managed backup and recovery

ORACLE®

Oracle Database Backup and Recovery User's Guide, 11*g* Release 1 (11.1)

B28270-03

Primary Author: Lance Ashdown

Contributing Author: Antonio Romero, Katherine Weill

Contributors: Tammy Bednar, Anand Beldalker, Timothy Chien, Mark Dilman, Senad Dizdar, Raymond Guzman, Stephan Haisley, Wei Hu, Alex Hwang, Ashok Joshi, Vasudha Krishnaswamy, J. William Lee, Valarie Moore, Muthu Olagappan, Vsevolod Panteleenko, Cris Pedregal-Martin, Samitha Samaranayake, Francisco Sanchez, Vivian Schupmann, Vinay Srihari, Margaret Susairaj, Mike Stewart, Steven Wertheimer, Wanli Yang, Rob Zijlstra

Contents

Preface xxi

What's New in Backup and Recovery? xxiii

Part I Overview of Backup and Recovery

1 Introduction to Backup and Recovery

Purpose of Backup and Recovery 1-1
Data Protection 1-2
Data Preservation 1-3
Data Transfer 1-3
Oracle Backup and Recovery Solutions 1-3
Oracle Flashback Technology 1-5
Logical Flashback Features 1-5
Flashback Database 1-7
Data Recovery Advisor 1-7
Backup and Recovery Documentation Roadmap 1-8
Recovery Manager Documentation Roadmap 1-10
User-Managed Backup and Recovery Documentation Roadmap 1-10

2 Getting Started with RMAN

Overview of the RMAN Environment 2-1
Starting RMAN and Connecting to Database 2-2
Showing the Default RMAN Configuration 2-3
Backing Up a Database 2-4
Backing Up a Database in ARCHIVELOG Mode 2-4
Backing Up a Database in NOARCHIVELOG Mode 2-4
Typical Backup Options 2-5
Making Incremental Backups 2-5
Validating Database Files and Backups 2-7
Scripting RMAN Operations 2-8
Reporting on RMAN Operations 2-8
Listing Backups 2-8
Reporting on Database Files and Backups 2-9
Maintaining RMAN Backups 2-10

Crosschecking Backups 2-10
Deleting Obsolete Backups 2-10
Diagnosing and Repairing Failures with Data Recovery Advisor 2-11
Listing Failures and Determining Repair Options 2-11
Repairing Failures 2-12
Rewinding a Database with Flashback Database 2-13
Restoring and Recovering Database Files 2-13
Preparing to Restore and Recover Database Files 2-14
Recovering the Whole Database 2-14
Recovering Tablespaces 2-15
Recovering Individual Data Blocks 2-16

Part II Starting and Configuring RMAN

3 Recovery Manager Architecture

About the RMAN Environment 3-1
RMAN Command-Line Client 3-3
RMAN Channels 3-3
Channels and Devices 3-4
Automatic and Manual Channels 3-4
RMAN Repository 3-5
Media Management 3-6
RMAN Interaction with a Media Manager 3-6
Oracle Secure Backup 3-6
Backup Solutions Program 3-7
Flash Recovery Area 3-7
RMAN in a Data Guard Environment 3-7
RMAN Configuration in a Data Guard Environment 3-7
RMAN File Management in a Data Guard Environment 3-8

4 Starting and Interacting with the RMAN Client

Starting and Exiting RMAN 4-1
Specifying the Location of RMAN Output 4-2
Setting Globalization Support Environment Variables for RMAN 4-2
Entering RMAN Commands 4-2
Entering RMAN Commands at the RMAN Prompt 4-3
Using Command Files with RMAN 4-3
Entering Comments in RMAN Command Files 4-4
Using Substitution Variables in Command Files 4-4
Checking RMAN Syntax 4-5
Making Database Connections with RMAN 4-7
About RMAN Database Connections 4-7
Making RMAN Database Connections from the Operating System Command Line 4-8
Making Database Connections from the RMAN Prompt 4-10
Connecting RMAN to an Auxiliary Database 4-10
Making RMAN Database Connections Within Command Files 4-11

Diagnosing RMAN Connection Problems 4-12
Using the RMAN Pipe Interface 4-12
Executing Multiple RMAN Commands In Succession Through a Pipe: Example 4-13
Executing RMAN Commands In a Single Job Through a Pipe: Example 4-13

5 Configuring the RMAN Environment

Configuring the Environment for RMAN Backups 5-1
Showing and Clearing Persistent RMAN Configurations 5-2
Configuring the Default Device for Backups: Disk or SBT 5-3
Configuring the Default Type for Backups: Backup Sets or Copies 5-4
Configuring Channels 5-4
Configuring Control File and Server Parameter File Autobackups 5-7
Configuring RMAN to Make Backups to a Media Manager 5-8
Prerequisites for Using a Media Manager with RMAN 5-9
Determining the Location of the Media Management Library 5-9
Configuring Media Management Software for RMAN Backups 5-10
Testing Whether the Media Manager Library Is Integrated Correctly 5-10
Configuring SBT Channels for Use with a Media Manager 5-12
Configuring the Flash Recovery Area 5-13
Overview of the Flash Recovery Area 5-14
Enabling the Flash Recovery Area 5-16
Configuring Locations for Control Files and Redo Logs 5-19
Configuring RMAN File Creation in the Flash Recovery Area 5-20
Configuring the Backup Retention Policy 5-21
Configuring a Redundancy-Based Retention Policy 5-21
Configuring a Recovery Window-Based Retention Policy 5-22
Disabling the Retention Policy 5-22
Configuring Backup Optimization 5-23
Overview of Backup Optimization 5-23
Effect of Retention Policies on Backup Optimization for SBT Backups 5-24
Configuring Backup Optimization 5-25
Configuring an Archived Redo Log Deletion Policy 5-26
About Archived Redo Log Deletion Policies 5-26
Enabling an Archived Redo Log Deletion Policy 5-27
Configuring Oracle Flashback Database and Restore Points 5-27
About Restore Points and Flashback Database 5-28
Prerequisites for Flashback Database and Guaranteed Restore Points 5-32
Enabling Flashback Database 5-33
Creating Normal and Guaranteed Restore Points 5-33
Configuring the Environment for Optimal Flashback Database Performance 5-34
Configuring RMAN in a Data Guard Environment 5-35

6 Configuring the RMAN Environment: Advanced Topics

Configuring Advanced Channel Options 6-1
About Channel Control Options 6-1
Configuring Specific Channel Parameters 6-2

Configuring Advanced Backup Options 6-3
Configuring the Maximum Size of Backup Sets 6-4
Configuring the Maximum Size of Backup Pieces 6-4
Configuring Backup Duplexing 6-5
Configuring Tablespaces for Exclusion from Whole Database Backups 6-6
Configuring the Backup Compression Algorithm 6-6
Configuring Backup Encryption 6-7
Configuring Auxiliary Instance Datafile Names 6-10
Configuring the Snapshot Control File Location 6-11
Viewing the Configured Location of the Snapshot Control File 6-11
Setting the Location of the Snapshot Control File 6-11
Configuring RMAN for Use with a Shared Server 6-11
Enabling Lost Write Detection 6-13

Part III Backing Up and Archiving Data

7 RMAN Backup Concepts

Consistent and Inconsistent RMAN Backups 7-1
Consistent Backups 7-1
Inconsistent Backups 7-2
Online Backups and Backup Mode 7-2
Backup Sets 7-3
Backup Sets and Backup Pieces 7-3
Compression for Backup Sets 7-4
Encryption for Backup Sets 7-4
Filenames for Backup Pieces 7-5
Number and Size of Backup Pieces 7-5
Number and Size of Backup Sets 7-6
Multiplexed Backup Sets 7-6
Proxy Copies 7-8
Image Copies 7-8
RMAN-Created Image Copies 7-8
User-Managed Image Copies 7-9
Multiple Copies of RMAN Backups 7-10
Duplexed Backup Sets 7-10
Backups of Backups 7-11
Control File and Server Parameter File Autobackups 7-12
When RMAN Performs Control File Autobackups 7-12
How RMAN Performs Control File Autobackups 7-12
Incremental Backups 7-13
Multilevel Incremental Backups 7-13
Block Change Tracking 7-15
Incremental Backup Algorithm 7-16
Recovery with Incremental Backups 7-16
Backup Retention Policies 7-17
Recovery Window 7-17
Backup Redundancy 7-19

Batch Deletes of Obsolete Backups 7-19
Backup Retention Policy and Flash Recovery Area Deletion Rules 7-20

8 Backing Up the Database

Overview of RMAN Backups 8-1
Purpose of RMAN Backups 8-1
Basic Concepts of RMAN Backups 8-1
Specifying Backup Output Options 8-2
Specifying the Device Type for an RMAN Backup 8-2
Specifying Backup Set or Copy for an RMAN Backup to Disk 8-3
Specifying a Format for RMAN Backups 8-3
Specifying Tags for an RMAN Backup 8-4
Making Compressed Backups 8-6
Backing Up Database Files with RMAN 8-7
Making Whole Database Backups with RMAN 8-7
Backing Up Tablespaces and Datafiles with RMAN 8-8
Backing Up Control Files with RMAN 8-8
Backing Up Server Parameter Files with RMAN 8-10
Backing Up a Database in NOARCHIVELOG Mode 8-10
Backing Up Archived Redo Logs with RMAN 8-11
About Backups of Archived Redo Logs 8-11
Backing Up Archived Redo Log Files 8-12
Backing Up Only Archived Redo Logs That Need Backups 8-13
Deleting Archived Redo Logs After Backups 8-13
Making and Updating Incremental Backups 8-14
Purpose of Incremental Backups 8-14
Planning an Incremental Backup Strategy 8-15
Making Incremental Backups 8-16
Incrementally Updating Backups 8-17
Using Block Change Tracking to Improve Incremental Backup Performance 8-20
Making Database Backups for Long-Term Storage 8-23
Purpose of Archival Backups 8-23
Basic Concepts of Archival Backups 8-23
Making an Archival Backup for Long-Term Storage 8-24
Making a Temporary Archival Backup 8-25
Backing Up RMAN Backups 8-26
About Backups of Backups 8-26
Backing Up Backup Sets with RMAN 8-28
Backing Up Image Copy Backups with RMAN 8-29

9 Backing Up the Database: Advanced Topics

Limiting the Size of RMAN Backup Sets 9-1
About Backup Set Size 9-1
Limiting the Size of Backup Sets with BACKUP ... MAXSETSIZE 9-2
Dividing the Backup of a Large Datafile into Sections 9-2
Using Backup Optimization to Skip Files 9-3

Optimizing a Daily Archived Log Backup to a Single Tape: Scenario 9-4
Optimizing a Daily Archived Log Backup to Multiple Media Families: Scenario 9-4
Creating a Weekly Secondary Backup of Archived Logs: Example 9-5
Skipping Offline, Read-Only, and Inaccessible Files 9-6
Duplexing Backup Sets 9-6
Duplexing Backup Sets with CONFIGURE BACKUP COPIES 9-7
Duplexing Backup Sets with BACKUP ... COPIES 9-8
Making Split Mirror Backups with RMAN 9-8
Encrypting RMAN Backups 9-10
About RMAN Backup Encryption Settings 9-10
Making Transparent-Mode Encrypted Backups 9-11
Making Password-Mode Encrypted Backups 9-11
Making Dual-Mode Encrypted Backups 9-11
Restarting RMAN Backups 9-12
About Restartable Backups 9-12
Restarting a Backup After It Partially Completes 9-13
Managing Backup Windows 9-13
About Backup Windows 9-13
Specifying a Backup Duration 9-13
Permitting Partial Backups in a Backup Window 9-14
Minimizing Backup Load and Duration 9-14

Part IV Managing RMAN Backups

10 Reporting on RMAN Operations

Overview of RMAN Reporting 10-1
Purpose of RMAN Reporting 10-1
Basic Concepts of RMAN Reporting 10-1
Listing Backups and Recovery-Related Objects 10-3
About the LIST Command 10-3
Listing Backups and Copies 10-5
Listing Selected Backups and Copies 10-7
Listing Database Incarnations 10-9
Listing Restore Points 10-9
Reporting on Backups and Database Schema 10-10
About Reports of RMAN Backups 10-10
Reporting on Files Needing a Backup Under a Retention Policy 10-11
Reporting on Datafiles Affected by Unrecoverable Operations 10-12
Reporting on Obsolete Backups 10-13
Reporting on the Database Schema 10-14
Using V$ Views to Query Backup Metadata 10-15
Querying Details of Past and Current RMAN Jobs 10-15
Determining the Encryption Status of Backup Pieces 10-16
Querying Recovery Catalog Views 10-17
About Recovery Catalog Views 10-17
Querying Catalog Views for the Target DB_KEY or DBID Values 10-18
Querying RC_BACKUP_FILES 10-19

11 Maintaining RMAN Backups and Repository Records

Overview of RMAN Backup and Repository Maintenance 11-1
Purpose of Backup and Repository Maintenance 11-1
Basic Concepts of Backup and Repository Maintenance 11-2
Maintaining the Control File Repository 11-3
About Control File Records 11-3
Preventing the Loss of Control File Records 11-5
Protecting the Control File 11-5
Maintaining the Flash Recovery Area 11-6
Deletion Rules for the Flash Recovery Area 11-6
Monitoring Flash Recovery Area Space Usage 11-7
Managing Space For Flashback Logs in the Flash Recovery Area 11-7
Responding to a Full Flash Recovery Area 11-8
Dropping Restore Points 11-9
Changing the Flash Recovery Area to a New Location 11-9
Disabling the Flash Recovery Area 11-9
Responding to an Instance Crash During File Creation 11-10
Monitoring Flashback Database Performance Impact 11-10
Flashback Writer (RVWR) Behavior With I/O Errors 11-11
Updating the RMAN Repository 11-11
Crosschecking the RMAN Repository 11-12
Changing the Repository Status of Backups and Copies 11-14
Adding Backup Records to the RMAN Repository 11-16
Removing Records from the RMAN Repository 11-18
Deleting RMAN Backups and Archived Redo Logs 11-19
Overview of RMAN Deletion 11-19
Deleting All Backups and Copies 11-21
Deleting Specified Backups and Copies 11-22
Deleting Expired RMAN Backups and Copies 11-23
Deleting Obsolete RMAN Backups Based on Retention Policies 11-23
Dropping a Database 11-24

12 Managing a Recovery Catalog

Overview of the Recovery Catalog 12-1
Purpose of the Recovery Catalog 12-1
Basic Concepts for the Recovery Catalog 12-2
Basic Steps of Managing a Recovery Catalog 12-3
Creating a Recovery Catalog 12-4
Configuring the Recovery Catalog Database 12-4
Creating the Recovery Catalog Schema Owner 12-6
Executing the CREATE CATALOG Command 12-6
Registering a Database in the Recovery Catalog 12-7
About Registration of a Database in the Recovery Catalog 12-7
Registering a Database with the REGISTER DATABASE Command 12-8
Cataloging Backups in the Recovery Catalog 12-9
Creating and Managing Virtual Private Catalogs 12-9

About Virtual Private Catalogs ... 12-9
Creating and Granting Privileges to a Virtual Private Catalog Owner ... 12-10
Creating a Virtual Private Catalog ... 12-11
Revoking Privileges from a Virtual Private Catalog Owner ... 12-12
Dropping a Virtual Private Catalog ... 12-12
Protecting the Recovery Catalog ... 12-13
Backing Up the Recovery Catalog ... 12-13
Recovering the Recovery Catalog ... 12-15
Managing Stored Scripts ... 12-15
About Stored Scripts ... 12-15
Creating Stored Scripts ... 12-16
Replacing Stored Scripts ... 12-17
Executing Stored Scripts ... 12-17
Creating and Executing Dynamic Stored Scripts ... 12-18
Printing Stored Scripts ... 12-19
Listing Stored Script Names ... 12-20
Deleting Stored Scripts ... 12-20
Executing a Stored Script at RMAN Startup ... 12-21
Maintaining a Recovery Catalog ... 12-21
About Recovery Catalog Maintenance ... 12-21
Resynchronizing the Recovery Catalog ... 12-22
Updating the Recovery Catalog After Changing a DB_UNIQUE_NAME ... 12-25
Unregistering a Target Database from the Recovery Catalog ... 12-26
Resetting the Database Incarnation in the Recovery Catalog ... 12-28
Upgrading the Recovery Catalog ... 12-29
Importing and Moving a Recovery Catalog ... 12-31
Dropping a Recovery Catalog ... 12-33

Part V Diagnosing and Responding to Failures

13 RMAN Data Repair Concepts

Overview of RMAN Data Repair ... 13-1
Problems Requiring Data Repair ... 13-1
RMAN Data Repair Techniques ... 13-2
RMAN Restore Operations ... 13-3
Backup Selection ... 13-3
Restore Failover ... 13-4
Restore Optimization ... 13-4
RMAN Media Recovery ... 13-5
Selection of Incremental Backups and Archived Redo Logs ... 13-5
Database Incarnations ... 13-5

14 Diagnosing and Repairing Failures with Data Recovery Advisor

Overview of Data Recovery Advisor ... 14-1
Purpose of Data Recovery Advisor ... 14-1
Basic Concepts of Data Recovery Advisor ... 14-2

Basic Steps of Diagnosing and Repairing Failures 14-6
Listing Failures 14-6
Listing All Failures 14-7
Listing a Subset of Failures 14-7
Checking for Block Corruptions by Validating the Database 14-8
Determining Repair Options 14-10
Determining Repair Options for All Failures 14-10
Determining Repair Options for a Subset of Failures 14-11
Repairing Failures 14-12
About Repairing Failures 14-12
Repairing a Failure 14-13
Changing Failure Status and Priority 14-14

15 Validating Database Files and Backups

Overview of RMAN Validation 15-1
Purpose of RMAN Validation 15-1
Basic Concepts of RMAN Validation 15-1
Checking for Block Corruption with the VALIDATE Command 15-4
Parallelizing the Validation of a Datafile 15-5
Validating Database Files with BACKUP VALIDATE 15-6
Validating Backups Before Restoring Them 15-6

16 Performing Flashback and Database Point-in-Time Recovery

Overview of Flashback Technology and Database Point-in-Time Recovery 16-1
Purpose of Flashback and Database Point-in-Time-Recovery 16-1
Basic Concepts of Point-in-Time Recovery and Flashback Features 16-1
Rewinding a Table with Flashback Table 16-4
Prerequisites for Flashback Table 16-4
Performing a Flashback Table Operation 16-5
Rewinding a DROP TABLE Operation with Flashback Drop 16-7
About Flashback Drop 16-7
Prerequisites of Flashback Drop 16-7
Performing a Flashback Drop Operation 16-8
Rewinding a Database with Flashback Database 16-11
Prerequisites of Flashback Database 16-11
Performing a Flashback Database Operation 16-12
Performing Database Point-in-Time Recovery 16-14
Prerequisites of Database Point-in-Time Recovery 16-15
Performing Database Point-in-Time Recovery 16-15
Flashback and Database Point-in-Time Recovery Scenarios 16-17
Rewinding an OPEN RESETLOGS Operation with Flashback Database 16-17
Rewinding the Database to an SCN in an Abandoned Incarnation Branch 16-18
Recovering the Database to an Ancestor Incarnation 16-20

17 Performing Complete Database Recovery

Overview of Complete Database Recovery 17-1

Purpose of Complete Database Recovery 17-1
Scope of This Chapter 17-1
Preparing for Complete Database Recovery 17-2
Identifying the Database Files to Restore or Recover 17-3
Determining the DBID of the Database 17-5
Previewing Backups Used in Restore Operations 17-5
Validating Backups Before Restoring Them 17-8
Restoring Archived Redo Logs Needed for Recovery 17-8
Performing Complete Database Recovery 17-9
About Complete Database Recovery 17-10
Performing Complete Recovery of the Whole Database 17-10
Performing Complete Recovery of a Tablespace 17-13
Performing Complete Recovery After Switching to a Copy 17-16

18 Performing Block Media Recovery

Overview of Block Media Recovery 18-1
Purpose of Block Media Recovery 18-1
Basic Concepts of Block Media Recovery 18-2
Prerequisites for Block Media Recovery 18-3
Recovering Individual Blocks 18-4
Recovering All Blocks in V$DATABASE_BLOCK_CORRUPTION 18-5

19 Performing RMAN Recovery: Advanced Scenarios

Recovering a NOARCHIVELOG Database with Incremental Backups 19-1
Restoring the Server Parameter File 19-2
Restoring the Server Parameter File from a Control File Autobackup 19-3
Creating an Initialization Parameter File with RMAN 19-4
Performing Recovery with a Backup Control File 19-4
About Recovery with a Backup Control File 19-4
Performing Recovery with a Backup Control File and No Recovery Catalog 19-6
Performing Disaster Recovery 19-8
Prerequisites of Disaster Recovery 19-8
Recovering the Database After a Disaster 19-8
Restoring a Database on a New Host 19-10
Preparing to Restore a Database on a New Host 19-11
Testing the Restore of a Database on a New Host 19-12

20 Performing RMAN Tablespace Point-in-Time Recovery (TSPITR)

Overview of RMAN TSPITR 20-1
Purpose of RMAN TSPTIR 20-1
Basic Concepts of RMAN TSPITR 20-2
Basic Steps of RMAN TSPITR 20-3
Prerequisites and Consequences of TSPITR 20-3
Consequences of TSPITR 20-4
Special Considerations When Not Using a Recovery Catalog 20-4
Planning and Preparing for TSPITR 20-5

Choosing the Right Target Time for TSPITR 20-5
Determining the Recovery Set 20-5
Identifying and Preserving Objects That Will Be Lost After TSPITR 20-7
Performing Fully Automated RMAN TSPITR 20-8
Performing Customized RMAN TSPITR with an RMAN-Managed Auxiliary Instance 20-10
Renaming Oracle Managed Files in TSPITR 20-10
Renaming TSPITR Recovery Set Datafiles with SET NEWNAME 20-10
Naming TSPITR Auxiliary Set Datafiles 20-11
Using Image Copies for Faster RMAN TSPITR Performance 20-14
Customizing Initialization Parameters for the Automatic Auxiliary Instance in TSPITR . 20-16
Performing RMAN TSPITR Using Your Own Auxiliary Instance 20-18
Preparing Your Own Auxiliary Instance for RMAN TSPITR 20-18
Preparing RMAN Commands for TSPITR with Your Own Auxiliary Instance 20-20
Executing TSPITR with Your Own Auxiliary Instance 20-20
Performing TSPITR with Your Own Auxiliary Instance: Scenario 20-21
Troubleshooting RMAN TSPITR 20-23
Troubleshooting Filename Conflicts 20-23
Troubleshooting Identification of Tablespaces with Undo Segments 20-23
Troubleshooting the Restart of a Manual Auxiliary Instance After TSPITR Failure 20-23

Part VI Tuning and Troubleshooting

21 Tuning RMAN Performance

Purpose of RMAN Performance Tuning 21-1
Basic Concepts of RMAN Performance Tuning 21-1
Read Phase 21-3
Copy Phase 21-6
Write Phase for SBT 21-6
Write Phase for Disk 21-9
Using V$ Views to Diagnose RMAN Performance Problems 21-10
Monitoring RMAN Job Progress with V$SESSION_LONGOPS 21-10
Identifying Bottlenecks with V$BACKUP_SYNC_IO and V$BACKUP_ASYNC_IO 21-12
Tuning RMAN Backup Performance 21-13
Step 1: Remove the RATE Parameter from Channel Settings 21-13
Step 2: If You Use Synchronous Disk I/O, Set DBWR_IO_SLAVES 21-14
Step 3: If You Fail to Allocate Shared Memory, Set LARGE_POOL_SIZE 21-14
Step 4: Tune the Read, Write, and Copy Phases 21-15

22 Troubleshooting RMAN Operations

Interpreting RMAN Message Output 22-1
Identifying Types of Message Output 22-1
Recognizing RMAN Error Message Stacks 22-2
Identifying Error Codes 22-2
Interpreting RMAN Error Stacks 22-5
Identifying RMAN Return Codes 22-7
Using V$ Views for RMAN Troubleshooting 22-7

Monitoring RMAN Interaction with the Media Manager 22-8
Correlating Server Sessions with RMAN Channels 22-9
Testing the Media Management API 22-11
Obtaining the sbttest Utility 22-11
Obtaining Online Documentation for the sbttest Utility 22-11
Using the sbttest Utility 22-12
Terminating an RMAN Command 22-13
Terminating the Session with ALTER SYSTEM KILL SESSION 22-13
Terminating the Session at the Operating System Level 22-13
Terminating an RMAN Session That Is Hung in the Media Manager 22-13

Part VII Transferring Data with RMAN

23 Duplicating a Database

Overview of RMAN Database Duplication 23-1
Purpose of Database Duplication 23-1
Basic Concepts of Database Duplication 23-2
Basic Steps of Database Duplication 23-3
Making Backups and Archived Logs Accessible to the Duplicate Instance 23-3
Making SBT Backups Accessible to the Duplicate Instance 23-4
Making Disk Backups Accessible to the Duplicate Instance 23-4
Choosing a Strategy for Naming Duplicate Files 23-6
Preparing the Auxiliary Instance 23-7
Step 1: Create an Oracle Password File for the Auxiliary Instance 23-7
Step 2: Establish Oracle Net Connectivity to the Auxiliary Instance 23-8
Step 3: Create an Initialization Parameter File for the Auxiliary Instance 23-8
Step 4: Start the Auxiliary Instance with SQL*Plus 23-10
Starting and Configuring RMAN Before Duplication 23-10
Step 1: Start RMAN and Connect to the Database Instances 23-10
Step 2: Mount or Open the Source Database 23-11
Step 3: Configure RMAN Channels for Use in the Duplication 23-11
Duplicating a Database 23-11
Duplicating a Database to a Remote Host with the Same Directory Structure 23-12
Duplicating a Database to a Remote Host with a Different Directory Structure 23-13
Creating a Duplicate Database on the Local Host 23-14
Duplicating a Database with Oracle Managed Files or Automatic Storage Management 23-14
Naming Duplicate Files with Alternative Techniques 23-16
Naming Duplicate Control Files 23-17
Naming Duplicate Online Redo Log Files 23-17
Naming Duplicate Datafiles 23-18
Naming Duplicate Tempfiles 23-19
RMAN Duplication Scenarios 23-19
Duplicating a Subset of the Source Database Tablespaces 23-19
Using DUPLICATE to Restore an Archival Backup 23-21
Using SET NEWNAME to Name Duplicate Files 23-22
Using CONFIGURE AUXNAME to Name Duplicate Files 23-23

24 Creating Transportable Tablespace Sets

Overview of Creating Transportable Tablespace Sets 24-1
Purpose of Creating Transportable Tablespace Sets 24-1
Basic Concepts of Transportable Tablespace Sets 24-2
Basic Steps of Creating Transportable Tablespace Sets 24-4
Customizing Initialization Parameters for the Auxiliary Instance 24-5
Setting Initialization Parameters for the Auxiliary Instance 24-5
Setting the Location of the Auxiliary Instance Parameter File 24-6
Creating a Transportable Tablespace Set 24-7
Troubleshooting Creation of Transportable Tablespace Sets 24-8
Transportable Tablespace Set Scenarios 24-8
Creating a Transportable Tablespace Set at a Specified Time or SCN 24-8
Specifying Locations for Data Pump Files 24-9
Specifying Auxiliary File Locations 24-10

25 Transporting Data Across Platforms

Overview of Cross-Platform Data Transportation 25-1
Purpose of Cross-Platform Data Transportation 25-1
Basic Concepts of Cross-Platform Data Transportation 25-2
Performing Cross-Platform Tablespace Conversion on the Source Host 25-3
Performing Cross-Platform Datafile Conversion on the Destination Host 25-4
About Cross-Platform Datafile Conversion on the Destination Host 25-4
Using CONVERT DATAFILE to Convert Datafile Formats 25-5
Checking the Database Before Cross-Platform Database Conversion 25-7
Converting Datafiles on the Source Host When Transporting a Database 25-9
Converting Datafiles on the Destination Host When Transporting the Database 25-11
Performing Preliminary Datafile Conversion Steps on the Source Host 25-11
Converting Datafiles on the Destination Host 25-13

26 Performing ASM Data Migration

Overview of ASM Data Migration 26-1
Purpose of ASM Data Migration 26-1
Basic Concepts of ASM Data Migration 26-1
Basics Steps of Data Migration to ASM 26-2
Preparing to Migrate the Database to ASM 26-2
Migrating the Database to ASM 26-4
Migrating a Database from ASM to Alternative Storage 26-8
Moving Datafiles Between ASM Disk Groups 26-8

Part VIII Performing User-Managed Backup and Recovery

27 Making User-Managed Database Backups

Querying V$ Views to Obtain Backup Information 27-1
Listing Database Files Before a Backup 27-1
Determining Datafile Status for Online Tablespace Backups 27-2

Making User-Managed Backups of the Whole Database ... 27-3
Making Consistent Whole Database Backups ... 27-3
Making User-Managed Backups of Tablespaces and Datafiles ... 27-4
Making User-Managed Backups of Offline Tablespaces and Datafiles ... 27-4
Making User-Managed Backups of Online Tablespaces and Datafiles ... 27-5
Making User-Managed Backups of the Control File ... 27-10
Backing Up the Control File to a Binary File ... 27-10
Backing Up the Control File to a Trace File ... 27-11
Making User-Managed Backups of Archived Redo Logs ... 27-11
Making User-Managed Backups in SUSPEND Mode ... 27-11
About the Suspend/Resume Feature ... 27-12
Making Backups in a Suspended Database ... 27-12
Making User-Managed Backups to Raw Devices ... 27-14
Backing Up to Raw Devices on Linux and UNIX ... 27-14
Backing Up to Raw Devices on Windows ... 27-16
Making Backups with the Volume Shadow Copy Service (VSS) ... 27-17
Verifying User-Managed Datafile Backups ... 27-17
Testing the Restore of Datafile Backups ... 27-17
Running the DBVERIFY Utility ... 27-17

28 Performing User-Managed Database Flashback and Recovery

Performing Flashback Database with SQL*Plus ... 28-1
Overview of User-Managed Media Recovery ... 28-2
About User-Managed Restore and Recovery ... 28-2
Automatic Recovery with the RECOVER Command ... 28-3
Recovery When Archived Logs Are in the Default Location ... 28-5
Recovery When Archived Logs Are in a Nondefault Location ... 28-5
Recovery Cancellation ... 28-6
Parallel Media Recovery ... 28-7
Performing Complete Database Recovery ... 28-7
Performing Closed Database Recovery ... 28-8
Performing Open Database Recovery ... 28-11
Performing Incomplete Database Recovery ... 28-13
Performing Cancel-Based Incomplete Recovery ... 28-14
Performing Time-Based or Change-Based Incomplete Recovery ... 28-16
Recovering a Database in NOARCHIVELOG Mode ... 28-16
Troubleshooting Media Recovery ... 28-17
About User-Managed Media Recovery Problems ... 28-18
Investigating the Media Recovery Problem: Phase 1 ... 28-19
Trying to Fix the Recovery Problem Without Corrupting Blocks: Phase 2 ... 28-20
Deciding Whether to Allow Recovery to Corrupt Blocks: Phase 3 ... 28-21
Allowing Recovery to Corrupt Blocks: Phase 4 ... 28-23
Performing Trial Recovery ... 28-23

29 Performing User-Managed Recovery: Advanced Scenarios

Responding to the Loss of a Subset of the Current Control Files ... 29-1
Copying a Multiplexed Control File to a Default Location ... 29-1

Copying a Multiplexed Control File to a Nondefault Location 29-2
Recovering After Loss of All Current Control Files 29-2
Recovering with a Backup Control File in the Default Location 29-3
Recovering with a Backup Control File in a Nondefault Location 29-4
Recovering Through an Added Datafile with a Backup Control File 29-4
Recovering Read-Only Tablespaces with a Backup Control File 29-5
Re-Creating a Control File 29-6
Recovering Through a RESETLOGS with a Created Control File 29-7
Recovery of Read-Only Files with a Re-Created Control File 29-8
Re-Creating Datafiles When Backups Are Unavailable 29-8
Recovering NOLOGGING Tables and Indexes 29-9
Recovering Transportable Tablespaces 29-10
Recovering After the Loss of Online Redo Log Files 29-10
Recovering After Losing a Member of a Multiplexed Online Redo Log Group 29-11
Recovering After the Loss of All Members of an Online Redo Log Group 29-12
Recovering from a Dropped Table Without Using Flashback Features 29-15
Dropping a Database with SQL*Plus 29-16

Glossary

Index

Preface

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documentation
- Conventions

Audience

Backup and Recovery User's Guide is intended for database administrators who perform the following tasks:

- Back up, restore, and recover Oracle databases
- Perform maintenance on backups of database files
- Transfer data between a file system and ASM or between platforms when installing Oracle Database

To use this document, you need to know the following:

- Relational database concepts and basic database administration as described in *Oracle Database Concepts* and the *Oracle Database Administrator's Guide*
- The operating system environment under which you are running the database

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an

otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Related Documentation

For more information, see these Oracle resources:

- *Oracle Database Backup and Recovery Reference*
- *Oracle Database Utilities*
- *Oracle Database Storage Administrator's Guide*

You can access information about the Backup Solutions Program at

`http://www.oracle.com/technology/deploy/availability`

Many books in the documentation set use the sample schemas of the seed database, which is installed by default when you install Oracle Database. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

Oracle error message documentation is only available in HTML. If you only have access to the Oracle Documentation CD, then you can browse the error messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in Backup and Recovery?

This section describes the new backup and recovery features in Oracle Database 11*g* Release 1. The new features for this release greatly improve the manageability of Oracle Database backup and recovery. In particular, manageability is increased by the introduction of Data Recovery Advisor, better integration of RMAN with Data Guard, expansion of the recovery catalog functionality, and improved management of archived redo logs.

The new features in this release include:

- Data Recovery Advisor

  Data Recovery Advisor is a built-in tool to automatically diagnose data failures and recommend repairs. You can repair failures manually or request that they be repaired automatically. Data Recovery Advisor supports the `LIST FAILURE`, `CHANGE FAILURE`, `ADVISE FAILURE`, and `REPAIR FAILURE` commands.

  > **See Also:** Chapter 14, "Diagnosing and Repairing Failures with Data Recovery Advisor", and *Oracle Database Backup and Recovery Reference* to learn about the Data Recovery Advisor commands

- Improved integration with Data Guard

  You can now set persistent RMAN configurations for a primary or **physical standby database** when RMAN is not connected as `TARGET` to the database. RMAN works seamlessly on all databases in the Data Guard environment, enabling you to use backups made on one database for restore and recovery on another database. The same **recovery catalog** can manage metadata for all primary and standby databases.

  > **See Also:** "Configuring RMAN in a Data Guard Environment" on page 5-35, and *Oracle Data Guard Concepts and Administration*

- Improved handling of long-term backups

  You can create a long-term or **archival backup** with `BACKUP . . . KEEP` that retains only the archived log files needed to make the backup consistent.

  > **See Also:** "Making Database Backups for Long-Term Storage" on page 8-23, and *Oracle Database Backup and Recovery Reference* to learn about the `BACKUP` command

- Backup failover for archived redo logs in the **flash recovery area**

When backing up archived redo log files located in the flash recovery area, RMAN can fail over to archiving destinations outside the recovery area. RMAN can use an intact copy of an archived log in an alternative location to continue writing backups when a log in the recovery area is missing or corrupted.

See Also: "Archived Redo Log Failover" on page 8-11

- Archived log deletion policy enhancements

  When you `CONFIGURE` an archived log deletion policy, the configuration applies to all archiving destinations, including the flash recovery area. Both `BACKUP ... DELETE INPUT` and `DELETE ... ARCHIVELOG` obey this configuration, as does the flash recovery area. You can also `CONFIGURE` an **archived redo log deletion policy** so that logs are eligible for deletion only after being applied to or transferred to standby database destinations. You can set the policy for mandatory standby destinations only, or for any standby destinations.

  See Also:

  - "Configuring an Archived Redo Log Deletion Policy" on page 5-26,
  - *Oracle Database Backup and Recovery Reference* to learn about the `CONFIGURE ARCHIVELOG DELETION POLICY` command
  - *Oracle Data Guard Concepts and Administration* to learn how to use RMAN in a Data Guard environment

- Network-enabled database duplication without backups

  You can use the `DUPLICATE` command to create a **duplicate database** or **physical standby database** over the network without a need for pre-existing database backups. This form of duplication is called **active database duplication**.

  See Also: Chapter 23, "Duplicating a Database," and *Oracle Database Backup and Recovery Reference* to learn about the `DUPLICATE` command

- Recovery catalog enhancements

  The owner of a recovery catalog can `GRANT` or `REVOKE` access to a subset of the catalog to other database users in the same recovery catalog database. This subset is called a **virtual private catalog**. You can also use the `IMPORT CATALOG` command to merge one recovery catalog (or metadata for specific databases in the catalog) into another recovery catalog.

  See Also: Chapter 12, "Managing a Recovery Catalog," and *Oracle Database Backup and Recovery Reference* to learn about the recovery catalog commands

- Multisection backups

  RMAN can back up a single file in parallel by dividing the work among multiple channels. Each channel backs up one **file section**. You create a **multisection backup** by specifying `SECTION SIZE` on the `BACKUP` command. Restoring a multisection backup in parallel is automatic and requires no option.

  You can parallelize validations of a file with `VALIDATE ... SECTION SIZE`.

See Also: "Dividing the Backup of a Large Datafile into Sections" on page 9-2, and *Oracle Database Backup and Recovery Reference* to learn about the `BACKUP` and `VALIDATE` commands

- Undo optimization

  The `BACKUP` command does not back up undo that is not needed for recovery of a backup. Undo is not needed if it was generated for a transaction that has already committed. This undo can represent the majority of undo in the database.

  See Also: "Overview of Backup Optimization" on page 5-23, and *Oracle Database Backup and Recovery Reference* to learn about the `BACKUP` command

- Improved block media recovery performance

  When performing **block media recovery**, RMAN automatically searches the **flashback logs**, if they are available, for the required blocks before searching backups. Using blocks from the flashback logs can significantly improve block media recovery performance.

  See Also: "Overview of Block Media Recovery" on page 18-1

- Improved block corruption detection

  Several database components and utilities, including RMAN, can now detect a **corrupt block** and record it in `V$DATABASE_BLOCK_CORRUPTION`. When instance recovery detects a corrupt block, it records it in this view automatically. Oracle Database automatically updates this view when block corruptions are detected or repaired. The `VALIDATE` command is enhanced with many new options such as `VALIDATE ... BLOCK` and `VALIDATE DATABASE`.

  See Also: Chapter 15, "Validating Database Files and Backups," and *Oracle Database Backup and Recovery Reference* to learn about the `VALIDATE` command

- Faster backup compression

  In addition to the existing `BZIP2` algorithm for **binary compression** of backups, RMAN also supports the `ZLIB` algorithm. `ZLIB` runs faster than `BZIP2`, but produces larger files. `ZLIB` requires the Oracle Advanced Compression option. You can use the `CONFIGURE COMPRESSION ALGORITHM` command to choose between `BZIP2` (default) and `ZLIB` for RMAN backups.

  See Also: "Configuring the Backup Compression Algorithm" on page 6-6, and *Oracle Database Backup and Recovery Reference* to learn about the `CONFIGURE` command

- Block change tracking support for standby databases

  You can enable **block change tracking** on a physical standby database. When you back up the standby database, RMAN can use the **block change tracking file** to quickly identify the blocks that changed since the last incremental backup.

  See Also: "Making and Updating Incremental Backups" on page 8-14

- Improved scripting with RMAN substitution variables

  You can create RMAN command files and stored scripts that accept user input at runtime. Thus, backup scripts can use RMAN substitution variables for tags, filenames, restore point names, and so on.

  > **See Also:** "Using Substitution Variables in Command Files" on page 4-4 and "Creating and Executing Dynamic Stored Scripts" on page 12-18

- Integration with VSS-enabled applications

  The **Oracle VSS writer** is integrated with applications that use the **Volume Shadow Copy Service (VSS)** infrastructure on Windows. You can use VSS-enabled software and storage systems to back up and restore an Oracle database. A key benefit is the ability to make a shadow copy of an open database.

  > **See Also:** *Oracle Database Platform Guide for Microsoft Windows* to learn how to perform backup and recovery with VSS-enabled applications, and *Oracle Database Backup and Recovery Reference* to learn about the `BACKUP` command

- Lost write detection

  You can enable the `DB_LOST_WRITE_PROTECT` initialization parameter to detect a **lost write** during managed recovery of a standby database or media recovery of a primary database. Lost write detection is disabled by default.

  > **See Also:** "Enabling Lost Write Detection" on page 6-13

- Backup of read-only transportable tablespaces

  In previous releases, RMAN could not back up transportable tablespaces until they were made read/write at the destination database. Now RMAN can back up transportable tablespaces when they are not read/write and restore these backups.

  > **See Also:** "Backing Up Tablespaces and Datafiles with RMAN" on page 8-8

- Backup and recovery enhancements in Oracle Enterprise Manager

  Enterprise Manager includes an interface for Data Recovery Advisor.

  > **See Also:** *Oracle Database 2 Day DBA* for details about Enterprise Manager enhancements related to backup and recovery

- Oracle Flashback Transaction

  You can reverse a transaction. Oracle Database determines the dependencies between transactions and in effect creates a compensating transaction that reverses the unwanted changes. The database rewinds to a state as if the transaction, and any transactions that could be dependent on it, never occurred.

  > **See Also:** *Oracle Database Advanced Application Developer's Guide* to learn how to use this feature

- Flashback data archive

A **flashback data archive** enables the database to automatically track and store all transactional changes to a table for the duration of its lifetime. Thus, you do not need to build this functionality into database applications.

A flashback data archive is especially useful for compliance, audit reports, data analysis, and DSS (Decision Support Systems). You can use some of the **logical flashback features** with a flashback data archive to access data from far in the past.

> **See Also:** *Oracle Database Advanced Application Developer's Guide* to learn how to configure and use a flashback data archive, and "Oracle Flashback Technology" on page 1-5 for an overview of flashback features

- Improved media recovery performance for databases on SMP systems

  Media recovery of databases on symmetric multiprocessing (SMP) computers is now faster. The performance improvements include the following:

  – More parallelism
  – More efficient asynchronous redo read, parse, and apply
  – Fewer synchronization points in the parallel apply algorithm
  – The media recovery **checkpoint** at a redo log boundary no longer blocks the apply of the next log

  No configuration is necessary, although you can use new **parallel recovery** wait events for tuning if the default apply rate is not satisfactory.

  > **See Also:** *Oracle Database Reference* to learn about Oracle wait events

Part I

Overview of Backup and Recovery

The chapters in this part introduce backup and recovery and explain how to devise a backup and recovery strategy:

- Chapter 1, "Introduction to Backup and Recovery"
- Chapter 2, "Getting Started with RMAN"

1

Introduction to Backup and Recovery

This chapter explains the purpose of Oracle Database backup and recovery and summarizes the Oracle solutions. This chapter includes the following topics:

- Purpose of Backup and Recovery
- Oracle Backup and Recovery Solutions
- Oracle Flashback Technology
- Data Recovery Advisor
- Backup and Recovery Documentation Roadmap

> **Note:** To get started with Recovery Manager (RMAN) right away, proceed to Chapter 2, "Getting Started with RMAN."

Purpose of Backup and Recovery

As a backup administrator, your principal duty is to devise, implement, and manage a backup and recovery strategy. In general, the purpose of a **backup and recovery** strategy is to protect the database against data loss and reconstruct the database after data loss. Typically, backup administration tasks include the following:

- Planning and testing responses to different kinds of failures
- Configuring the database environment for backup and recovery
- Setting up a backup schedule
- Monitoring the backup and recovery environment
- Troubleshooting backup problems
- Recovering from data loss if the need arises

As a backup administrator, you may also be asked to perform other duties that are related to backup and recovery:

- Data preservation, which involves creating a database copy for long-term storage
- Data transfer, which involves moving data from one database or one host to another

The purpose of this manual is to explain how to perform the preceding tasks.

Data Protection

As a backup administrator, your primary job is making and monitoring backups for data protection. A **backup** is a copy of data of a database that you can use to reconstruct data. A backup can be either a **physical backup** or a **logical backup**.

Physical backups are copies of the physical files used in storing and recovering a database. These files include datafiles, control files, and archived redo logs. Ultimately, every physical backup is a copy of files that store database information to another location, whether on disk or on offline storage media such as tape.

Logical backups contain logical data such as tables and stored procedures. You can use Oracle Data Pump to export logical data to binary files, which you can later import into the database. The Data Pump command-line clients `expdp` and `impdp` use the `DBMS_DATAPUMP` and `DBMS_METADATA` PL/SQL packages.

Physical backups are the foundation of any sound backup and recovery strategy. Logical backups are a useful supplement to physical backups in many circumstances but are not sufficient protection against data loss without physical backups.

Unless otherwise specified, the term **backup** as used in the backup and recovery documentation refers to a physical backup. Backing up a database is the act of making a physical backup. The focus in the backup and recovery documentation set is almost exclusively on physical backups.

While several problems can halt the normal operation of an Oracle database or affect database I/O operations, only the following typically require DBA intervention and data recovery: **media failure**, user errors, and application errors. Other failures may require DBA intervention without causing data loss or requiring recovery from backup. For example, you may need to restart the database after an instance failure or allocate more disk space after statement failure because of a full datafile.

Media Failures

A media failure is a physical problem with a disk that causes a failure of a read or write of a disk file required to run the database. Any database file can be vulnerable to a media failure. The appropriate recovery technique following a media failure depends on the files affected and the types of backup available.

One particularly important aspect of backup and recovery is developing a **disaster recovery** strategy to protect against catastrophic data loss, for example, the loss of an entire database host.

User Errors

User errors occur when, either due to an error in application logic or a manual mistake, data in a database is changed or deleted incorrectly. User errors are estimated to be the greatest single cause of database downtime.

Data loss due to user error can be either localized or widespread. An example of localized damage is deleting the wrong `SMITH` from the employees table. This type of damage requires surgical detection and repair. An example of widespread damage is a batch job that deletes the company orders for the current month. In this case, drastic action is required to avoid a extensive database downtime.

While user training and careful management of privileges can prevent most user errors, your backup strategy determines how gracefully you recover the lost data when user error does cause data loss.

Application Errors

Sometimes a software malfunction can corrupt data blocks. In a **physical corruption**, which is also called a media corruption, the database does not recognize the block at all: the **checksum** is invalid, the block contains all zeros, or the header and footer of the block do not match. If the corruption is not extensive, then you can often repair it easily with **block media recovery**.

See Also:

- Chapter 8, "Backing Up the Database"
- *Oracle Database Utilities* to learn how to use Data Pump

Data Preservation

Data preservation is related to data protection, but serves a different purpose. For example, you may need to preserve a copy of a database as it existed at the end of a business quarter. This backup is not part of the disaster recovery strategy. The media to which these backups are written are often unavailable after the backup is complete. You may send the tape into fire storage or ship a portable hard drive to a testing facility. RMAN provides a convenient way to create a backup and exempt it from your **backup retention policy**. This type of backup is known as an **archival backup**.

See Also: "Making Database Backups for Long-Term Storage" on page 8-23

Data Transfer

In some situations you may need to take a backup of a database or database component and move it to another location. For example, you can use Recovery Manager (RMAN) to create a database copy, create a tablespace copy that can be imported into another database, or move an entire database from one platform to another. These tasks are not strictly speaking part of a backup and recovery strategy, but they do require the use of database backups, and so may be included in the duties of a backup administrator.

See Also: The chapters in Part VII, "Transferring Data with RMAN"

Oracle Backup and Recovery Solutions

When implementing a backup and recovery strategy, you have the following solutions available:

- **Recovery Manager (RMAN)**

  This tool integrates with sessions running on an Oracle database to perform a range of backup and recovery activities, including maintaining an **RMAN repository** of historical data about backups. You can access RMAN through the command line or through Oracle Enterprise Manager.

- User-managed backup and recovery

  In this solution, you perform backup and recovery with a mixture of host operating system commands and SQL*Plus recovery commands.

Both of the preceding solutions are supported by Oracle and are fully documented, but RMAN is the preferred solution for database backup and recovery. RMAN performs the same types of backup and recovery available through user-managed techniques more easily, provides a common interface for backup tasks across different host

operating systems, and offers a number of backup techniques not available through user-managed methods.

Most of this manual focuses on RMAN-based backup and recovery. User-managed backup and recovery techniques are covered in Section VIII, "Performing User-Managed Backup and Recovery." RMAN gives you access to several backup and recovery techniques and features not available with user-managed backup and recovery. The most noteworthy are the following:

- Incremental backups

  An **incremental backup** stores only blocks changed since a previous backup. Thus, they provide more compact backups and faster recovery, thereby reducing the need to apply redo during **datafile media recovery**. If you enable **block change tracking**, then you can improve performance by avoiding full scans of every input datafile. You use the `BACKUP INCREMENTAL` command to perform incremental backups.

- Block media recovery

  You an repair a datafile with only a small number of corrupt data blocks without taking it offline or restoring it from backup. You use the `RECOVER` command to perform **block media recovery**.

- Unused block compression

  In **unused block compression**, RMAN can skip data blocks that have never been used and, in some cases, used blocks that are currently unused.

- Binary compression

  A **binary compression** mechanism integrated into Oracle Database reduces the size of backups.

- Encrypted backups

  RMAN uses **backup encryption** capabilities integrated into Oracle Database to store backup sets in an encrypted format. To create encrypted backups on disk, the database must use the Advanced Security Option. To create encrypted backups directly on tape, RMAN must use the Oracle Secure Backup SBT interface, but does not require the Advanced Security Option.

Whether you use RMAN or user-managed methods, you can supplement physical backups with logical backups of schema objects made with Data Pump Export utility. You can later use Data Pump Import to re-create data after restore and recovery. Logical backups are for the most part beyond the scope of the backup and recovery documentation.

Table 1–1 summarizes the features of the different backup techniques.

Table 1–1 Feature Comparison of Backup Techniques

Feature	Recovery Manager	User-Managed	Data Pump Export
Closed database backups	Supported. Requires instance to be mounted.	Supported.	Not supported.
Open database backups	Supported. No need to use `BEGIN/END BACKUP` statements.	Supported. Must use `BEGIN/END BACKUP` statements.	Requires rollback or undo segments to generate consistent backups.
Incremental backups	Supported.	Not supported.	Not supported.

Table 1–1 (Cont.) Feature Comparison of Backup Techniques

Feature	Recovery Manager	User-Managed	Data Pump Export
Corrupt block detection	Supported. Identifies corrupt blocks and logs in `V$DATABASE_BLOCK_CORRUPTION`.	Not supported.	Supported. Identifies corrupt blocks in the export log.
Automatic specification of files to include in a backup	Supported. Establishes the name and locations of all files to be backed up (whole database, tablespaces, datafiles, control files, and so on).	Not supported. Files to be backed up must be located and copied manually.	Not applicable.
Backup repository	Supported. Backups are recorded in the control file, which is the main repository of RMAN metadata. Additionally, you can store this metadata in a **recovery catalog**, which is a schema in a different database.	Not supported. DBA must keep own records of backups.	Not supported.
Backups to a media manager	Supported. Interfaces with a **media manager**. RMAN also supports proxy copy, a feature that allows a media manager to manage completely the transfer of data between disk and backup media.	Supported. Backup to tape is manual or controlled by a media manager.	Not supported.
Backup of initialization parameter file	Supported.	Supported.	Not supported.
Backup of password and networking files	Not supported.	Supported.	Not supported.
Platform-independent language for backups	Supported.	Not supported.	Supported.

Oracle Flashback Technology

As explained in *Oracle Database Concepts*, **Oracle Flashback Technology** complements your physical backup and recovery strategy. This set of features provides an additional layer of data protection. Specifically, you can use flashback features to view past states of data and rewind your database without restoring backups or performing **point-in-time recovery**. In general, flashback features are more efficient and less disruptive than media recovery in most situations in which they apply.

Logical Flashback Features

Most of the flashback features of Oracle operate at the logical level, enabling you to view and manipulate database objects. The logical-level flashback features of Oracle do not depend on RMAN and are available whether or not RMAN is part of your backup strategy. With the exception of Flashback Drop, the logical flashback features rely on **undo data**, which are records of the effects of each database update and the values overwritten in the update.

Oracle Database includes the following logical flashback features:

- Oracle Flashback Query

You can specify a target time and run queries against a database, viewing results as they would have appeared at the target time. To recover from an unwanted change like an update to a table, you could choose a target time before the error and run a query to retrieve the contents of the lost rows. *Oracle Database Advanced Application Developer's Guide* explains how to use this feature.

- Oracle Flashback Version Query

  You can view all versions of all rows that ever existed in one or more tables in a specified time interval. You can also retrieve metadata about the differing versions of the rows, including start and end time, operation, and transaction ID of the transaction that created the version. You can use this feature to recover lost data values and to audit changes to the tables queried. *Oracle Database Advanced Application Developer's Guide* explains how to use this feature.

- Oracle Flashback Transaction Query

  You can view changes made by a single transaction, or by all the transactions during a period of time. *Oracle Database Advanced Application Developer's Guide* explains how to use this feature.

- Oracle Flashback Transaction

  You can reverse a transaction. Oracle Database determines the dependencies between transactions and in effect creates a compensating transaction that reverses the unwanted changes. The database rewinds to a state as if the transaction, and any transactions that could be dependent on it, had never happened. *Oracle Database Advanced Application Developer's Guide* explains how to use this feature.

- Oracle Flashback Table

  You can recover a table or set of tables to a specified point in time in the past without taking any part of the database offline. In many cases, Flashback Table eliminates the need to perform more complicated point-in-time recovery operations. Flashback Table restores tables while automatically maintaining associated attributes such as current indexes, triggers, and constraints, and in this way enabling you to avoid finding and restoring database-specific properties. "Rewinding a Table with Flashback Table" on page 16-4 explains how to use this feature.

- Oracle Flashback Drop

  You can reverse the effects of a `DROP TABLE` statement. "Rewinding a DROP TABLE Operation with Flashback Drop" on page 16-7 explains how to use this feature.

A **flashback data archive** enables you to use some of the logical flashback features to access data from far back in the past. A flashback data archive consists of one or more tablespaces or parts of tablespaces. When you create a flashback data archive, you specify the name, retention period, and tablespace. You can also specify a default flashback data archive. The database automatically purges old historical data the day after the retention period expires.

You can turn flashback archiving on and off for individual tables. By default, flashback archiving is turned off for every table.

See Also:

- Chapter 16, "Performing Flashback and Database Point-in-Time Recovery" to learn how to perform Flashback Table and Flashback Drop
- *Oracle Database Advanced Application Developer's Guide* for more information on the logical flashback features

Flashback Database

At the physical level, **Oracle Flashback Database** provides a more efficient data protection alternative to **database point-in-time recovery (DBPITR)**. If the current datafiles have unwanted changes, then you can use the RMAN command `FLASHBACK DATABASE` to revert the datafiles to their contents at a past time. The end product is much like the result of a DBPITR, but is generally much faster because it does not require restoring datafiles from backup and requires less redo than media recovery.

Flashback Database uses **flashback logs** to access past versions of data blocks and some information from archived redo logs. Flashback Database requires that you configure a **flash recovery area** for a database because the flashback logs can only be stored there. Flashback logging is not enabled by default. Space used for flashback logs is managed automatically by the database and balanced against space required for other files in the flash recovery area.

Oracle Database also supports restore points in conjunction with Flashback Database and backup and recovery. A **restore point** is an alias corresponding to a **system change number (SCN)**. You can create a restore point at any time if you anticipate needing to return part or all of a database to its contents at that time. A **guaranteed restore point** ensures that you can use Flashback Database to return a database to the time of the restore point.

See Also: "Rewinding a Database with Flashback Database" on page 16-11 to learn how to perform Flashback Database with the `FLASHBACK DATABASE` command

Data Recovery Advisor

Oracle Database includes a **Data Recovery Advisor** tool that automatically diagnoses persistent data failures, presents appropriate repair options, and executes repairs at your request. Data Recovery Advisor provides a single point of entry for Oracle backup and recovery solutions. You can use Data Recovery Advisor through the Enterprise Manager Database Control or Grid Control console or through the RMAN command-line client.

A database failure usually manifests itself as a set of symptoms: error messages, alerts, trace files and dumps, and failed data integrity checks. Data Recovery Advisor automatically diagnoses and informs you of these failures. Within the context of Data Recovery Advisor, a **failure** is a persistent data corruption that can be directly mapped to a set of repair actions. Each failure has a status of open or closed. Each failure also has a priority of critical, high, or low.

Failures are detected by data integrity checks, which are diagnostic procedures executed to assess the health of the database or its components. If a **data integrity check** reveals a failure, then Data Recovery Advisor automatically assesses the effect of a set of failures and maps it to a set of repair options. In most cases, Data Recovery Advisor presents both automated and manual repair options.

Data Recovery Advisor determines the best automated repair option and its effect on the database. The **repair option** may include repairs such as datafile restore and recovery, media recovery, Flashback Database, and so on. Before presenting an automated repair option, Data Recovery Advisor validates it with respect to the specific environment and the availability of media components required to complete the proposed repair.

If you choose an automated repair option, then RMAN coordinates sessions on the Oracle database to perform the **repair** for you. The Data Recovery Advisor tool verifies the repair success and closes the appropriate failures.

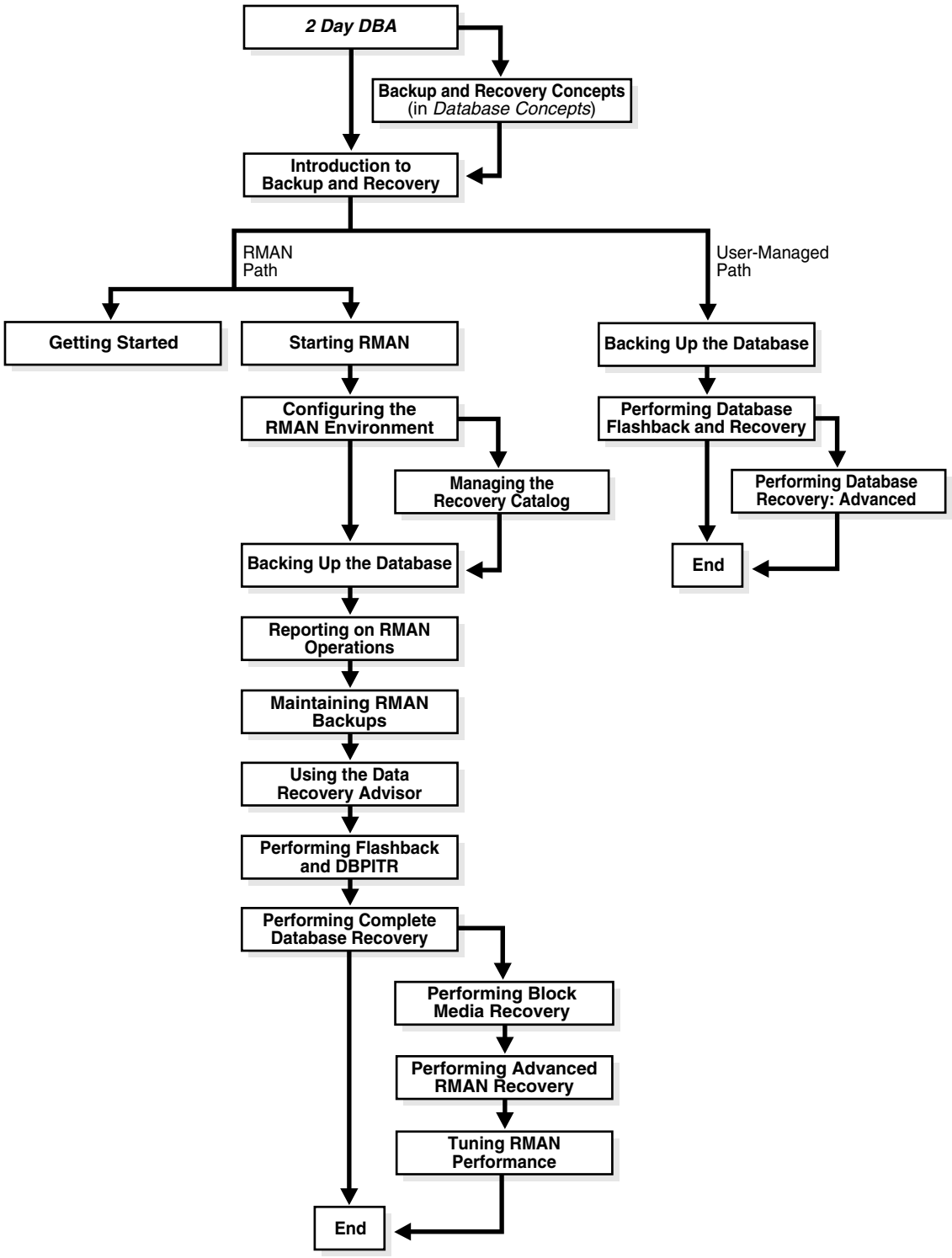
> **See Also:** Chapter 14, "Diagnosing and Repairing Failures with Data Recovery Advisor," to learn how to use Data Recovery Advisor

Backup and Recovery Documentation Roadmap

Figure 1–1 illustrates the recommended way to navigate the backup and recovery documentation. The roadmap is divided into two main paths: RMAN and user-managed backup and recovery. Optional paths are shown as splitting off and then rejoining each main path.

If you are new to Oracle Database and want to learn about backup recovery, then the best entry point is *Oracle Database 2 Day DBA*. The backup and recovery chapter explains how to use Enterprise Manager to perform basic operations. Optionally, you can expand your knowledge of basic backup and recovery principles by reading the relevant chapter in *Oracle Database Concepts*.

Figure 1–1 Backup and Recovery Documentation Roadmap



As shown in Figure 1–1, you can either implement your backup and recovery strategy with RMAN, which is recommended, or with user-managed tools.

Recovery Manager Documentation Roadmap

If you use RMAN as your principal backup and recovery solution, then begin by reading "Getting Started with RMAN" on page 2-1. This brief chapter, which explains the most basic RMAN techniques, may be adequate for your purposes. For a more comprehensive explanation of how to implement a backup and recovery strategy with RMAN, read the chapters in the following order (optional chapters are not listed):

1. Read Chapter 4, "Starting and Interacting with the RMAN Client."

   This chapter explains how to start the RMAN client and connect to databases.

2. Read Chapter 5, "Configuring the RMAN Environment."

   This chapter explains how to perform basic tasks such as configuring a **flash recovery area**, **backup retention policy**, and **archived redo log deletion policy**.

3. Read Chapter 8, "Backing Up the Database."

   This chapter explains how to implement a basic backup strategy.

4. Read Chapter 10, "Reporting on RMAN Operations."

   This chapter explains how to monitor RMAN backup and recovery operations. Specifically, the chapter explains how to use the reporting commands (`LIST`, `REPORT`, and `SHOW`) and the relevant V$ and recovery catalog views.

5. Read Chapter 11, "Maintaining RMAN Backups and Repository Records."

   This chapter explains how to verify the existence of backups, change the repository status of backups, delete backups, and perform other maintenance tasks.

6. Read Chapter 14, "Diagnosing and Repairing Failures with Data Recovery Advisor."

   This chapter explains how to use the Data Recovery Advisor tool. You can use it to list failures, obtain advice about to respond to these failures, and in some cases automatically repair the failures.

7. Read Chapter 16, "Performing Flashback and Database Point-in-Time Recovery."

   This chapter explains how to use the `FLASHBACK DATABASE` command and perform point-in-time recovery with the `RECOVER DATABASE` command.

8. Read Chapter 17, "Performing Complete Database Recovery."

   This chapter explains how to recover individual tablespaces or the database.

User-Managed Backup and Recovery Documentation Roadmap

If you do *not* use RMAN as your principal backup and recovery solution, then you must use third-party tools to make your backups and SQL or SQL*Plus commands to perform recovery. Read the chapters in the following order:

1. Read Chapter 27, "Making User-Managed Database Backups."

   This chapter explains how to make backups with third-party tools.

2. Read Chapter 28, "Performing User-Managed Database Flashback and Recovery."

   This chapter explains how to use the SQL statement `FLASHBACK DATABASE` and to perform recovery with the SQL*Plus `RECOVER` command.

3. Read Chapter 29, "Performing User-Managed Recovery: Advanced Scenarios."

   This chapter explains various recovery scenarios.

2

Getting Started with RMAN

This chapter is intended for new users who want to start using RMAN right away without first reading the more detailed chapters in this book. This chapter provides the briefest possible digest of the most important RMAN concepts and tasks and is not a substitute for the rest of the backup and recovery documentation set.

This chapter contains the following topics:

- Overview of the RMAN Environment
- Starting RMAN and Connecting to Database
- Showing the Default RMAN Configuration
- Backing Up a Database
- Reporting on RMAN Operations
- Maintaining RMAN Backups
- Diagnosing and Repairing Failures with Data Recovery Advisor
- Rewinding a Database with Flashback Database
- Restoring and Recovering Database Files

Overview of the RMAN Environment

Recovery Manager (RMAN) is an Oracle Database client that performs backup and recovery tasks on your databases and automates administration of your backup strategies. It greatly simplifies backing up, restoring, and recovering database files.

The RMAN environment consists of the utilities and databases that play a role in backing up your data. At a minimum, the environment for RMAN must include the following components:

- A **target database**

  An Oracle database to which RMAN is connected with the `TARGET` keyword. A target database is a database on which RMAN is performing backup and recovery operations. RMAN always maintains metadata about its operations on a database in the control file of the database. The RMAN metadata is known as the **RMAN repository**.

- The **RMAN client**

  An Oracle Database executable that interprets commands, directs server sessions to execute those commands, and records its activity in the target database control file. The RMAN executable is automatically installed with the database and is

typically located in the same directory as the other database executables. For example, the RMAN client on Linux is located in `$ORACLE_HOME/bin`.

Some environments use the following optional components:

- A **flash recovery area**

  A disk location in which the database can store and manage files related to backup and recovery. You set the flash recovery area location and size with the `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` initialization parameters.

- A **media manager**

  An application required for RMAN to interface with sequential media devices such as tape libraries. A media manager controls these devices during backup and recovery, managing the loading, labeling, and unloading of media. Media management devices are sometimes called **SBT** (system backup to tape) devices.

- A **recovery catalog**

  A separate database schema used to record RMAN activity against one or more target databases. A recovery catalog preserves RMAN repository metadata if the control file is lost, making it much easier to restore and recover following the loss of the control file. The database may overwrite older records in the control file, but RMAN maintains records forever in the catalog unless deleted by the user.

This chapter explains how to use RMAN in the most basic configuration, which is *without* a recovery catalog or media manager.

> **See Also:** Chapter 3, "Recovery Manager Architecture" for a more detailed overview of the RMAN environment

Starting RMAN and Connecting to Database

The RMAN client is started by issuing the `rman` command at the command prompt of your operating system. After being started, RMAN displays a prompt for your commands as shown in the following example:

```
% rman
RMAN>
```

RMAN connections to a database are specified and authenticated in the same way as SQL*Plus connections to a database. The only difference is that RMAN connections to a target or auxiliary database require the `SYSDBA` privilege. The `AS SYSDBA` keywords are implied and cannot be explicitly specified. See *Oracle Database Administrator's Guide* to learn about database connection options when using SQL*Plus.

> **Caution:** Good security practice requires that passwords should not be entered in plain text on the command line. You should enter passwords in RMAN only when requested by an RMAN prompt. See *Oracle Database Security Guide* to learn about password protection.

You can connect to a database with command-line options or by using the `CONNECT TARGET` command. The following example starts RMAN and then connects to a target database through Oracle Net (note that `AS SYSDBA` is not specified because it is implied). RMAN prompts for a password.

```
% rman
```

```
RMAN> CONNECT TARGET SYS@prod

target database Password: password
connected to target database: PROD (DBID=39525561)
```

The following variation starts RMAN and then connects to a target database by using operating system authentication:

```
% rman
RMAN> CONNECT TARGET /

connected to target database: PROD (DBID=39525561)
```

To quit the RMAN client, enter `EXIT` at the RMAN prompt:

```
RMAN> EXIT
```

Syntax of Common RMAN Command-line Options

```
RMAN
[ TARGET connectStringSpec
| { CATALOG connectStringSpec }
| LOG ['] filename ['] [ APPEND ]
.
.
.
]...

connectStringSpec::=
['] [userid] [/ [password]] [@net_service_name] ['] 
```

The following example appends the output from an **RMAN session** to a text file at `/tmp/msglog.log`

```
% rman TARGET / LOG /tmp/msglog.log APPEND
```

> **See Also:** Chapter 4, "Starting and Interacting with the RMAN Client," to learn more about starting and using the RMAN client

Showing the Default RMAN Configuration

The RMAN backup and recovery environment is preconfigured for each target database. The configuration is persistent and applies to all subsequent operations on this target database, even if you exit and restart RMAN.

RMAN configured settings can specify backup devices, configure a connection to a backup device (known as a **channel**), policies affecting backup strategy, and others. The default configuration is adequate for most purposes.

To show the current configuration for a database:

1. Start RMAN and connect to a target database.
2. Run the `SHOW ALL` command.

   For example, enter the command at the RMAN prompt as follows:

   ```
   RMAN> SHOW ALL;
   ```

The output lists the `CONFIGURE` commands to re-create this configuration.

See Also: Chapter 5, "Configuring the RMAN Environment," and Chapter 6, "Configuring the RMAN Environment: Advanced Topics," to learn how to configure the RMAN environment

Backing Up a Database

Use the BACKUP command to back up files. RMAN backs up data to the configured default device for the type of backup requested. By default, RMAN creates backups on disk. If a flash recovery area is enabled, and if you do not specify the FORMAT parameter (see Table 2–1), then RMAN creates backups in the recovery area and automatically gives them unique names.

By default, RMAN creates backup sets rather than image copies. A **backup set** consists of one or more backup pieces, which are physical files written in a format that only RMAN can access. A **multiplexed backup set** contains the blocks from multiple input files. RMAN can write backup sets to disk or tape.

If you specify BACKUP AS COPY, then RMAN copies each file as an **image copy**, which is a bit-for-bit copy of a database file created on disk. Image copies are identical to copies created with operating system commands like cp on Linux or COPY on Windows, but are recorded in the RMAN repository and so are usable by RMAN. You can use RMAN to make image copies while the database is open.

See Also:

- Chapter 7, "RMAN Backup Concepts," to learn concepts relating to RMAN backups
- Chapter 8, "Backing Up the Database," to learn how to back up database files with RMAN
- *Oracle Database Backup and Recovery Reference* for BACKUP command syntax and semantics

Backing Up a Database in ARCHIVELOG Mode

If a database runs in ARCHIVELOG mode, then you can back up the database while it is open. The backup is called an **inconsistent backup** because redo is required during recovery to bring the database to a consistent state. As long as you have the archived redo logs needed to recover the backup, open database backups are as effective a means of data protection as consistent backups.

To back up the database and archived redo logs while the database is open:

1. Start RMAN and connect to a target database.
2. Run the BACKUP DATABASE command.

   For example, enter the following command at the RMAN prompt to back up the database and all archived redo log files to the default backup device:

   ```
   RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
   ```

Backing Up a Database in NOARCHIVELOG Mode

If a database runs in NOARCHIVELOG mode, then the only valid database backup is a **consistent backup**. For the backup to be consistent, the database must be mounted after a consistent shutdown. No recovery is required after restoring the backup.

To make a consistent database backup:

1. Start RMAN and connect to a target database.
2. Shut down the database consistently and then mount it.

   For example, enter the following commands to *guarantee* that the database is in a consistent state for a backup:

   ```
   RMAN> SHUTDOWN IMMEDIATE;
   RMAN> STARTUP FORCE DBA;
   RMAN> SHUTDOWN IMMEDIATE;
   RMAN> STARTUP MOUNT;
   ```

3. Run the `BACKUP DATABASE` command.

   For example, enter the following command at the RMAN prompt to back up the database to the default backup device:

   ```
   RMAN> BACKUP DATABASE;
   ```

   The following variation of the command creates image copy backups of all datafiles in the database:

   ```
   RMAN> BACKUP AS COPY DATABASE;
   ```

4. Open the database and resume normal operations.

   The following command opens the database:

   ```
   RMAN> ALTER DATABASE OPEN;
   ```

Typical Backup Options

The `BACKUP` command includes a host of options, parameters, and clauses that control backup output. The following table lists some typical backup options.

Table 2–1 Common Backup Options

Option	Description	Example

`FORMAT`	Specifies a location and name for backup pieces and copies. You must use substitution variables to generate unique filenames.
	The most common substitution variable is `%U`, which generates a unique name. Others include `%d` for the DB_NAME, `%t` for the backup set time stamp, `%s` for the backup set number, and `%p` for the backup piece number.
		`BACKUP`
		`  FORMAT 'AL_%d/%t/%s/%p'`
		`  ARCHIVELOG LIKE '%arc_dest%';`

`TAG`	Specifies a user-defined string as a label for the backup. If you do not specify a **tag**, then RMAN assigns a default tag with the date and time. Note that tags are always stored in the RMAN repository in uppercase.	`BACKUP`
		`  TAG 'weekly_full_db_bkup'`
		`  DATABASE MAXSETSIZE 10M;`

See Also: "Specifying Backup Output Options" on page 8-2

Making Incremental Backups

If you specify `BACKUP INCREMENTAL`, then RMAN creates an **incremental backup** of a database. Incremental backups capture block-level changes to a database made after a previous incremental backup. Incremental backups are generally smaller and faster

to make than full database backups. Recovery with incremental backups is faster than using redo logs alone.

The starting point for an incremental backup strategy is a **level 0 incremental backup**, which backs up all blocks in the database. An incremental backup at level 0 is identical in content to a **full backup**, but unlike a full backup the level 0 backup is considered a part of the incremental backup strategy.

A level 1 incremental backup contains only blocks changed after a previous incremental backup. If no level 0 backup exists in either the current or parent database **incarnation** when you run a level 1 backup, then RMAN makes a level 0 backup automatically.

> **Note:** You cannot make incremental backups when a `NOARCHIVELOG` database is open, although you can make incremental backups when the database is mounted after a consistent shutdown.

A level 1 backup can be a **cumulative incremental backup**, which includes all blocks changed since the most recent level 0 backup, or a **differential incremental backup**, which includes only blocks changed since the most recent incremental backup. Incremental backups are differential by default.

When restoring incremental backups, RMAN uses the level 0 backup as the starting point, then updates changed blocks based on level 1 backups where possible to avoid reapplying changes from redo one at a time. Recovering with incremental backups requires no additional effort on your part. If incremental backups are available, then RMAN uses them during recovery.

To make incremental backups of the database:

1. Start RMAN and connect to a target database.
2. Run the `BACKUP INCREMENTAL` command.

   The following example creates a level 0 incremental backup to serve as a base for an incremental backup strategy:

   ```
   BACKUP INCREMENTAL LEVEL 0 DATABASE;
   ```

   The following example creates a level 1 cumulative incremental backup:

   ```
   BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE;
   ```

   The following example creates a level 1 differential incremental backup:

   ```
   BACKUP INCREMENTAL LEVEL 1 DATABASE;
   ```

> **See Also:** "Incremental Backups" on page 7-13 for a more detailed conceptual overview of incremental backups and "Making and Updating Incremental Backups" on page 8-14

Making Incrementally Updated Backups

The RMAN **incrementally updated backup** feature is an efficient incremental backup routine. Changes from level 1 backups roll forward an image copy level 0 incremental backup, so that it includes all changes as of the SCN at which the level 1 incremental backup was created. Recovery of the updated level 0 incremental backup is faster because all changes from the level 1 incremental backup have already been applied.

The BACKUP FOR RECOVER OF COPY command specifies that an incremental backup should contain all changes since the SCN of a specified datafile copy (level 0 incremental backup) of your database. The following table explains which options to use with FOR RECOVER OF COPY to implement an incrementally updated backup strategy.

Table 2–2 FOR RECOVER OF COPY Options

BACKUP Option	Description	Example
FOR RECOVER OF COPY WITH TAG '*tag_name*'	Use the TAG parameter to identify the level 0 incremental backup serving as the basis of the incremental backup. If no level 0 datafile copy with the specified tag exists in either the current or parent database **incarnation**, then RMAN creates a level 0 datafile copy with the specified tag.	BACKUP INCREMENTAL LEVEL 1 FOR RECOVER OF COPY WITH TAG 'incr_update' DATABASE;
FOR RECOVER OF COPY DATAFILECOPY FORMAT '*format*'	Identifies the datafile copies to use as the basis for this incremental backup.	BACKUP INCREMENTAL LEVEL 1 FOR RECOVER OF COPY DATAFILECOPY FORMAT '/disk2/df1.cpy' DATABASE;

To implement an incrementally updated backup strategy:

1. Start RMAN and connect to a target database.
2. Run the RECOVER COPY and BACKUP INCREMENTAL commands.

   The following script, run on a regular basis, is all that is required to implement a strategy based on incrementally updated backups.

```
RECOVER COPY OF DATABASE
  WITH TAG 'incr_update';
BACKUP
  INCREMENTAL LEVEL 1
  FOR RECOVER OF COPY WITH TAG 'incr_update'
  DATABASE;
```

See Also: "Incrementally Updating Backups" on page 8-17

Validating Database Files and Backups

You can use the VALIDATE command to confirm that all database files exist, are in their correct location, and are free of physical corruption. The CHECK LOGICAL option also checks for logical block corruption.

To validate database files:

1. Start RMAN and connect to a target database.
2. Run the VALIDATE command for the desired files.

   For example, enter the following commands to validate all database files and archived redo log files for physical and logical corruption:

```
BACKUP VALIDATE CHECK LOGICAL
  DATABASE ARCHIVELOG ALL;
```

   You can also use the VALIDATE command to individual data blocks, as shown in the following example:

```
VALIDATE DATAFILE 4 BLOCK 10 TO 13;
```

You can also validate backup sets, as shown in the following example:

```
VALIDATE BACKUPSET 3;
```

You specify backup sets by primary key, which is shown in the output of the `LIST BACKUP` command.

See Also: Chapter 15, "Validating Database Files and Backups"

Scripting RMAN Operations

RMAN supports the use of command files to manage recurring tasks such as weekly backups. A **command file** is a client-side text file containing RMAN commands, exactly as you enter them at the RMAN prompt. You can use any file extension.The `RUN` command provides a degree of flow-of-control in your scripts.

To create and run a command file:

1. Use a text editor to create a command file.

   For example, create a command file with the following contents:

   ```
   # my_command_file.txt
   CONNECT TARGET /
   BACKUP DATABASE PLUS ARCHIVELOG;
   LIST BACKUP;
   EXIT;
   ```

2. Start RMAN and then execute the contents of a command file by running the `@` command at the RMAN prompt:

   ```
   % rman
   RMAN> @/my_dir/my_command_file.txt  # runs specified command file
   ```

   You can also launch RMAN with a command file to run, as shown here:

   ```
   % rman @/my_dir/my_command_file.txt
   ```

See Also: "Using Command Files with RMAN" on page 4-3 to learn more about command files, and "Using Substitution Variables in Command Files" on page 4-4 to learn how to use substitution variables in command files and pass parameters at runtime

Reporting on RMAN Operations

The RMAN `LIST` and `REPORT` commands generate reports on backup activities based on the RMAN repository. Use the `SHOW ALL` command to display the current RMAN configuration.

Listing Backups

Run the `LIST BACKUP` and `LIST COPY` commands to display information about backups and datafile copies listed in the repository. For backups, you can control the format of `LIST` output with the options in the following tables.

Table 2–3 LIST Options for Backups

Option	Example	Explanation
BY BACKUP	LIST BACKUP OF DATABASE BY BACKUP	Organizes the output by backup set. This is the default mode of presentation.
BY FILE	LIST BACKUP BY FILE	Lists the backups according to which file was backed up.
SUMMARY	LIST BACKUP SUMMARY	Displays summary output. By default, the output is VERBOSE.

For both backups and copies you have the following additional options.

Table 2–4 Additional LIST Options

Option	Example	Explanation
EXPIRED	LIST EXPIRED COPY	Lists backups that are recorded in the RMAN repository but that were not present at the expected location on disk or tape during the last CROSSCHECK command. An **expired backup** may have been deleted by an operating system utility.
RECOVERABLE	LIST BACKUP RECOVERABLE	Lists datafile backups or copies that have status AVAILABLE in the RMAN repository and that can be restored and recovered.

To list backups and copies:

1. Start RMAN and connect to a target database.
2. Run the LIST command at the RMAN prompt.

   You can display specific objects, as in the following examples:

```
LIST BACKUP OF DATABASE;
LIST COPY OF DATAFILE 1, 2;
LIST BACKUP OF ARCHIVELOG FROM SEQUENCE 10;
LIST BACKUPSET OF DATAFILE 1;
```

See Also: "Listing Backups and Recovery-Related Objects" on page 10-3 to learn more about the LIST command

Reporting on Database Files and Backups

The REPORT command performs more complex analysis than LIST. Some of the main options are shown in the following table.

Table 2–5 REPORT Options

Option	Example	Explanation
NEED BACKUP	REPORT NEED BACKUP DATABASE	Shows which files need backing up under current retention policy. Use optional REDUNDANCY and RECOVERY WINDOW parameters to specify different criteria.
OBSOLETE	REPORT OBSOLETE	Lists backups that are obsolete under the configured **backup retention policy**. Use the optional REDUNDANCY and RECOVERY WINDOW parameters to override the default.

Table 2–5 (Cont.) REPORT Options

Option	Example	Explanation
SCHEMA	REPORT SCHEMA	Reports the tablespaces and datafiles in the database at the current time (default) or a different time.
UNRECOVERABLE	REPORT UNRECOVERABLE	Lists all datafiles for which an unrecoverable operation has been performed against an object in the datafile since the last backup of the datafile.

To generate reports of database files and backups:

1. Start RMAN and connect to a target database.
2. Run the REPORT command at the RMAN prompt.

   The following example reports backups that are obsolete according to the currently configured backup retention policy:

   ```
   REPORT OBSOLETE;
   ```

   The following example reports the datafiles and tempfiles in the database:

   ```
   REPORT SCHEMA;
   ```

> **See Also:** "Reporting on Backups and Database Schema" on page 10-10 to learn how to use the REPORT command for RMAN reporting

Maintaining RMAN Backups

RMAN repository metadata is always stored in the control file of the target database. The **RMAN maintenance commands** use this metadata when managing backups.

Crosschecking Backups

The CROSSCHECK command synchronizes the logical records of RMAN backups and copies with the files on storage media. If a backup is on disk, then CROSSCHECK determines whether the header of the file is valid. If a backup is on tape, then RMAN queries the RMAN repository for the names and locations of the backup pieces. It is a good idea to crosscheck backups and copies before deleting them.

To crosscheck all backups and copies on disk:

1. Start RMAN and connect to a target database.
2. Run the CROSSCHECK command, as shown in the following example:

   ```
   CROSSCHECK BACKUP;
   CROSSCHECK COPY;
   ```

> **See Also:** "Crosschecking the RMAN Repository" on page 11-12 to learn how to crosscheck RMAN backups

Deleting Obsolete Backups

The DELETE command removes RMAN backups and copies from disk and tape, updates the status of the files to DELETED in the control file repository, and removes the records from the recovery catalog (if you use a catalog). If you run RMAN

interactively, and if you do not specify the NOPROMPT option, then DELETE displays a list of files and prompts for confirmation before deleting any file in the list.

The DELETE OBSOLETE command is particular useful because RMAN deletes backups and datafile copies recorded in the RMAN repository that are obsolete, that is, no longer needed. You can use options on the DELETE command to specify what is obsolete or use the configured **backup retention policy**.

To delete obsolete backups and copies:

1. Start RMAN and connect to a target database.
2. Run the DELETE OBSOLETE command, as shown in the following example:

```
DELETE OBSOLETE;
```

> **See Also:** "Deleting RMAN Backups and Archived Redo Logs" on page 11-19 to learn how to use the DELETE command

Diagnosing and Repairing Failures with Data Recovery Advisor

The simplest way to diagnose and repair database problems is to use the **Data Recovery Advisor**. This Oracle Database tool provides an infrastructure for diagnosing persistent data failures, presenting repair options to the user, and automatically executing repairs.

> **See Also:** "Overview of Data Recovery Advisor" on page 14-1

Listing Failures and Determining Repair Options

A **failure** is a persistent data corruption detected by the Health Monitor. Examples include physical and logical data block corruptions and missing datafiles. Each failure has a **failure priority** and **failure status**. The priority can be CRITICAL, HIGH, or LOW. The status can be OPEN or CLOSED.

You can run the LIST FAILURE command to show all known failures. If failures exist, then run the ADVISE FAILURE command in the same session to determine manual and automated repair options. The following example illustrates these two commands (sample output included).

Example 2–1 LIST FAILURE and ADVISE FAILURE

```
RMAN> LIST FAILURE;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
142        HIGH     OPEN      23-APR-07     One or more non-system datafiles are missing
101        HIGH     OPEN      23-APR-07     Datafile 1: '/disk1/oradata/prod/system01.dbf'
                                            contains one or more corrupt blocks

RMAN> ADVISE FAILURE;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
142        HIGH     OPEN      23-APR-07     One or more non-system datafiles are missing
```

```
101        HIGH     OPEN      23-APR-07     Datafile 1: '/disk1/oradata/prod/system01.dbf'
                                            contains one or more corrupt blocks

analyzing automatic repair options; this may take some time
using channel ORA_DISK_1
analyzing automatic repair options complete

Mandatory Manual Actions
========================
no manual actions available

Optional Manual Actions
=======================
1. If file /disk1/oradata/prod/users01.dbf was unintentionally renamed or moved, restore it

Automated Repair Options
========================
Option Repair Description
------ ------------------
1      Restore and recover datafile 28; Perform block media recovery of
       block 56416 in file 1
  Strategy: The repair includes complete media recovery with no data loss
  Repair script: /disk1/oracle/log/diag/rdbms/prod/prod/hm/reco_660500184.hm
```

The `ADVISE FAILURE` output shows both manual and automated repair options. First try to fix the problem manually. If you cannot fix the problem manually, then review the automated repair section.

An automated **repair option** describes a server-managed repair for one or more failures. Repairs are consolidated when possible so that a single repair can fix multiple failures. The repair option indicates which repair will be performed and whether data will be lost by performing the repair.

In Example 2–1, the output indicates the filename of a repair script containing RMAN commands. If you do not want to use Data Recovery Advisor to repair the failure automatically, then you can use the script as the basis of your own recovery strategy.

> **See Also:** "Listing Failures" on page 14-6 and "Determining Repair Options" on page 14-10

Repairing Failures

After running `LIST FAILURE` and `ADVISE FAILURE` in an **RMAN session**, you can run `REPAIR FAILURE` to execute a repair option. If you execute `REPAIR FAILURE` with no other command options, then RMAN uses the first repair option of the most recent `ADVISE FAILURE` command in the current session. Alternatively, specify the repair option number obtained from the most recent `ADVISE FAILURE` command. Example 2–2 illustrates how to repair the failures identified in Example 2–1.

Example 2–2 REPAIR FAILURE

```
RMAN> REPAIR FAILURE;
```

By default, `REPAIR FAILURE` prompts for confirmation before it begins executing. After executing a repair, Data Recovery Advisor reevaluates all existing failures on the possibility that they may also have been fixed. Data Recovery Advisor always verifies that failures are still relevant and automatically closes fixed failures. If a repair fails to complete because of an error, then the error triggers a new assessment and re-evaluation of existing failures and repairs.

> **See Also:** "Repairing Failures" on page 14-12

Rewinding a Database with Flashback Database

You can use the **Oracle Flashback Database** to rewind the whole database to a past time. Unlike media recovery, you do not need to restore datafiles to return the database to a past state.

To use the RMAN `FLASHBACK DATABASE` command, your database must have been previously configured to generate **flashback logs**. This configuration task is described in "Configuring Oracle Flashback Database and Restore Points" on page 5-27. Flashback Database works by rewinding changes to the datafiles that exist at the moment that you run the command. You cannot use the command to repair media failures or missing datafiles.

The database must be mounted when you issue `FLASHBACK DATABASE`. Note that if you have previously created a **restore point**, then you can flash back to this restore point if it falls within the **flashback database window**.

To rewind a database with Flashback Database:

1. Start RMAN and connect to a target database.
2. Ensure that the database is in a mounted state.

   The following commands shut down and then mount the database:

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT;
   ```

3. Run the `FLASHBACK DATABASE` command.

   The following examples illustrate different forms of the command:

   ```
   FLASHBACK DATABASE TO SCN 861150;

   FLASHBACK DATABASE
     TO RESTORE POINT BEFORE_CHANGES;

   FLASHBACK DATABASE TO TIME
     "TO_DATE('06/20/07','MM/DD/YY')";
   ```

4. After performing the Flashback Database, open the database read-only in SQL*Plus and run some queries to verify the database contents.

   Open the database read-only as follows:

   ```
   SQL "ALTER DATABASE OPEN READ ONLY";
   ```

5. If satisfied with the results, then issue the following sequence of commands to shut down and then open the database:

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT;
   ALTER DATABASE OPEN RESETLOGS;
   ```

See Also: "Rewinding a Database with Flashback Database" on page 16-11

Restoring and Recovering Database Files

Use the `RESTORE` and `RECOVER` commands for RMAN restore and recovery of physical database files. Restoring datafiles is retrieving them from backups as needed for a recovery operation. Media recovery is the application of changes from redo logs

and incremental backups to a restored datafile to bring the datafile forward to a desired SCN or point in time.

See Also: Chapter 17, "Performing Complete Database Recovery"

Preparing to Restore and Recover Database Files

If you need to recover the database because a media failure damages database files, then you should first ensure that you have the necessary backups. You can use the `RESTORE ... PREVIEW` command to report, but not restore, the backups that RMAN could use to restore to the specified time. RMAN queries the metadata and does not actually read the backup files. The database can be open when you run this command.

To preview a database restore and recovery:

1. Start RMAN and connect to the target database.
2. Optionally, list the current tablespaces and datafiles, as shown in the following command:

   ```
   RMAN> REPORT SCHEMA;
   ```

3. Run the `RESTORE DATABASE` command with the `PREVIEW` option.

   The following command specifies `SUMMARY` so that the backup metadata is not displayed in verbose mode (sample output included):

   ```
   RMAN> RESTORE DATABASE PREVIEW SUMMARY;

   Starting restore at 21-MAY-07
   allocated channel: ORA_DISK_1
   channel ORA_DISK_1: SID=80 device type=DISK

   List of Backups
   ===============
   Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
   ------- -- -- - ----------- --------------- ------- ------- ---------- ---
   11      B  F  A DISK        18-MAY-07       1       2       NO
   TAG20070518T181114
   13      B  F  A DISK        18-MAY-07       1       2       NO
   TAG20070518T181114
   using channel ORA_DISK_1

   List of Archived Log Copies for database with db_unique_name PROD
   =====================================================================

   Key     Thrd Seq     S Low Time
   ------- ---- ------- - ---------
   47      1    18      A 18-MAY-07
           Name: /disk1/oracle/dbs/db1r_60ffa882_1_18_0622902157.arc

   Media recovery start SCN is 586534
   Recovery must be done beyond SCN 587194 to clear datafile fuzziness
   validation succeeded for backup piece
   Finished restore at 21-MAY-07
   ```

Recovering the Whole Database

Use the `RESTORE DATABASE` and `RECOVER DATABASE` commands to recover the whole database. You must have previously made backups of all needed files. This scenario assumes that you can restore all datafiles to their original locations. If the

original locations are inaccessible, then use the SET NEWNAME command as described in "Restoring Datafiles to a Nondefault Location" on page 17-10.

To recover the whole database:

1. Prepare for recovery as explained in "Preparing to Restore and Recover Database Files" on page 2-14.
2. Place the database in a mounted state.

   The following example terminates the database instance (if it is started) and mounts the database:

   ```
   RMAN> STARTUP FORCE MOUNT;
   ```

3. Restore the database.

   The following example uses the preconfigured disk channel to restore the database:

   ```
   RMAN> RESTORE DATABASE;
   ```

4. Recover the database, as shown in the following example:

   ```
   RMAN> RECOVER DATABASE;
   ```

5. Open the database, as shown in the following example:

   ```
   RMAN> ALTER DATABASE OPEN;
   ```

Recovering Tablespaces

Use the RESTORE TABLESPACE and RECOVER TABLESPACE commands on individual tablespaces when the database is open. In this case, must take the tablespace that needs recovery offline, restore and then recover the tablespace, and bring the recovered tablespace online.

If you cannot restore a datafile to a new location, then use the RMAN SET NEWNAME command within a RUN command to specify the new filename. Afterward, use a SWITCH DATAFILE ALL command, which is equivalent to using the SQL statement ALTER DATABASE RENAME FILE, to update the control file to reflect the new names for all datafiles for which a SET NEWNAME has been issued in the RUN command.

Unlike in user-managed media recovery, you should *not* place an online tablespace in **backup mode**. Unlike user-managed tools, RMAN does not require extra logging or backup mode because it knows the format of data blocks.

To recover an individual tablespace when the database is open:

1. Prepare for recovery as explained in "Preparing to Restore and Recover Database Files" on page 2-14.
2. Take the tablespace to be recovered offline:

   The following example takes the users tablespace offline:

   ```
   RMAN> SQL 'ALTER TABLESPACE users OFFLINE';
   ```

3. Restore and recover the tablespace.

   The following RUN command, which you execute at the RMAN prompt, sets a new name for the datafile in the users tablespace:

   ```
   RUN
   ```

```
{
  SET NEWNAME FOR DATAFILE '/disk1/oradata/prod/users01.dbf'
    TO '/disk2/users01.dbf';
  RESTORE TABLESPACE users;
  SWITCH DATAFILE ALL;   # update control file with new filenames
  RECOVER TABLESPACE users;
}
```

4. Bring the tablespace online, as shown in the following example:

```
RMAN> SQL 'ALTER TABLESPACE users ONLINE';
```

You can also use `RESTORE DATAFILE` and `RECOVER DATAFILE` for recovery at the datafile level.

See Also:

- "Performing Complete Recovery of a Tablespace" on page 17-13
- "Online Backups and Backup Mode" on page 7-2

Recovering Individual Data Blocks

RMAN can recover individual corrupted datafile blocks. When RMAN performs a complete scan of a file for a backup, any corrupted blocks are listed in `V$DATABASE_BLOCK_CORRUPTION`. Corruption is usually reported in alert logs, trace files, or results of SQL queries.

To recover data blocks:

1. Obtain the block numbers of the corrupted blocks if you do not already have this information.

   The easiest way to locate trace files and the alert log is to connect SQL*Plus to the target database and execute the following query:

   ```
   SQL> SELECT NAME, VALUE
     2  FROM V$DIAG_INFO;
   ```

2. Start RMAN and connect to the target database.
3. Run the `RECOVER` command to repair the blocks.

   The following RMAN command recovers all corrupted blocks:

   ```
   RMAN> RECOVER CORRUPTION LIST;
   ```

   You can also recover individual blocks, as shown in the following example:

   ```
   RMAN> RECOVER DATAFILE 1 BLOCK 233, 235 DATAFILE 2 BLOCK 100 TO 200;
   ```

See Also: Chapter 18, "Performing Block Media Recovery"

Part II

Starting and Configuring RMAN

The chapters in this part explain the basic components of the RMAN environment and how to configure it. This part contains the following chapters:

- Chapter 3, "Recovery Manager Architecture"
- Chapter 4, "Starting and Interacting with the RMAN Client"
- Chapter 5, "Configuring the RMAN Environment"
- Chapter 6, "Configuring the RMAN Environment: Advanced Topics"

3

Recovery Manager Architecture

This chapter describes the Recovery Manager (RMAN) interface and the basic components of the RMAN environment. This chapter contains the following topics:

- About the RMAN Environment
- RMAN Command-Line Client
- RMAN Channels
- RMAN Repository
- Media Management
- Flash Recovery Area
- RMAN in a Data Guard Environment

About the RMAN Environment

The **Recovery Manager environment** consists of the various applications and databases that play a role in a backup and recovery strategy.

Table 3–1 lists some of the components in a typical RMAN environment.

Table 3–1 Components of the RMAN Environment

Component	Description
RMAN client	The client application that manages backup and recovery operations for a target database. The RMAN client can use Oracle Net to connect to a target database, so it can be located on any host that is connected to the target host through Oracle Net.
target database	A database containing the control files, datafiles, and optional archived redo logs that RMAN backs up or restores. RMAN uses the target database control file to gather metadata about the target database and to store information about its own operations. The work of backup and recovery is performed by server sessions running on the target database.
recovery catalog database	A database containing a **recovery catalog**, which contains metadata that RMAN uses to perform backup and recovery. You can create one recovery catalog that contains the RMAN metadata for multiple target databases. Unless you are using RMAN with a **physical standby database**, a recovery catalog is optional when using RMAN because RMAN stores its metadata in the control file of each target database.
recovery catalog schema	The user within the recovery catalog database that owns the metadata tables maintained by RMAN. RMAN periodically propagates metadata from the target database control file into the recovery catalog.

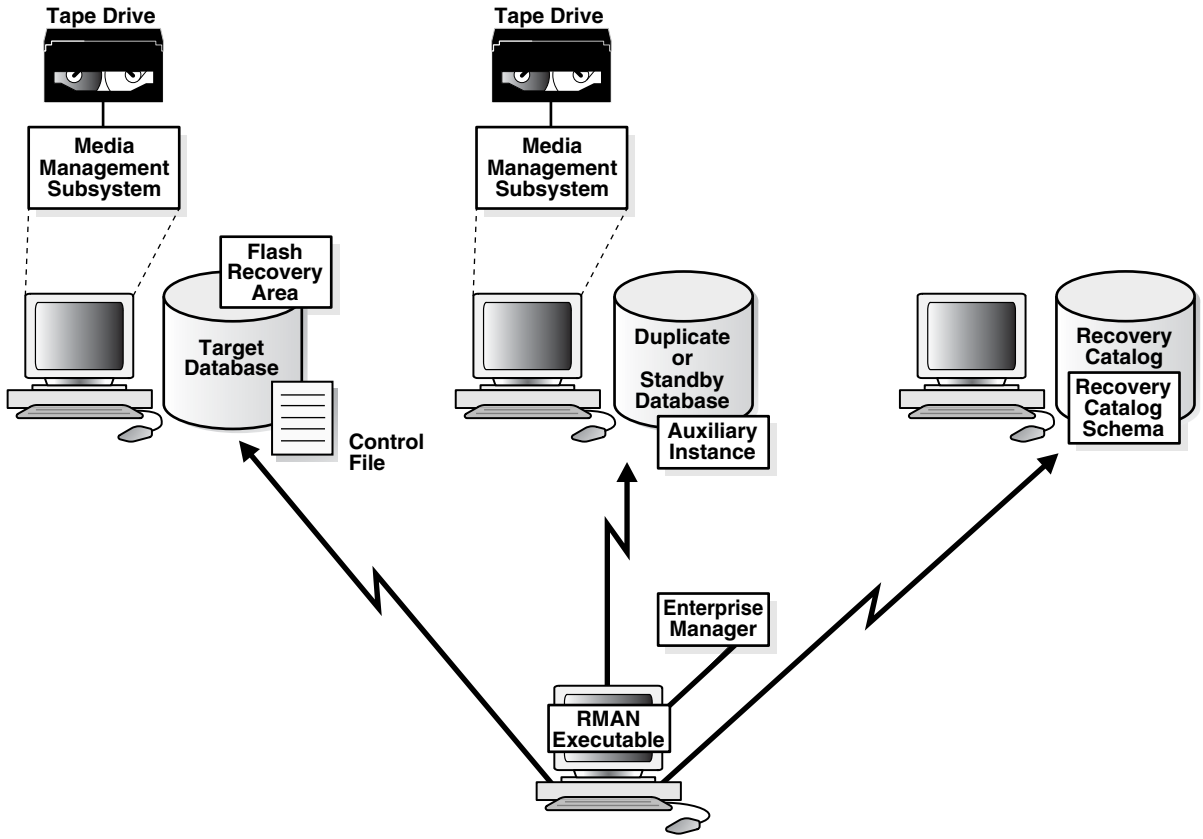
Table 3–1 (Cont.) Components of the RMAN Environment

Component	Description
physical standby database	A copy of the primary database that is updated with archived redo logs generated by the primary database. A physical standby database has the same `DBID` and `DB_NAME` values as the primary database, but a different `DB_UNIQUE_NAME`. You can fail over to the standby database if the primary database becomes inaccessible. RMAN can create, back up, or recover a standby database. Backups that you make at a standby database are usable at primary database or another standby database for same production database. A recovery catalog is required when you use RMAN in a Data Guard environment. **Note:** A logical standby database is treated as a separate database by RMAN because it has different DBID from its primary database. **See Also:** *Oracle Data Guard Concepts and Administration* to learn how to use RMAN in a Data Guard environment
duplicate database	A copy of the primary database that you can use for testing purposes. The DBID is different from the database from which it was created.
flash recovery area	A disk location that you can use to store recovery-related files such as control file and online redo log copies, archived redo logs, **flashback logs**, and RMAN backups. Oracle and RMAN manage the files in the flash recovery area automatically.
media manager	A vendor-specific application that enables RMAN to back up to a storage system such as tape.
media management catalog	A vendor-specific repository of metadata about a media management application.
Oracle Enterprise Manager	A browser-based interface to the database, including backup and recovery through RMAN.

The only required components in an RMAN environment are a target database and RMAN client, but most real-world configurations are more complicated. For example, you could use an RMAN client connecting to multiple media managers and multiple target databases and auxiliary databases, all accessed through Enterprise Manager.

Figure 3–1 illustrates components in a possible RMAN environment. The graphic shows that the primary database, standby database, and recovery catalog databases all reside on different computers. The primary and standby database hosts use a locally attached tape drive. The RMAN client and Enterprise Manager console run on a separate computer.

Figure 3–1 Sample RMAN Environment



See Also: *Oracle Database Net Services Administrator's Guide* to learn about Oracle Net

RMAN Command-Line Client

Use the RMAN command-line client to enter commands that you can use to manage all aspects of backup and recovery operations. RMAN uses a command language interpreter that can execute commands in interactive or batch mode. Even when you use the backup and recovery features in Enterprise Manager that are built on top of RMAN, an RMAN client executes behind the scenes.

RMAN Channels

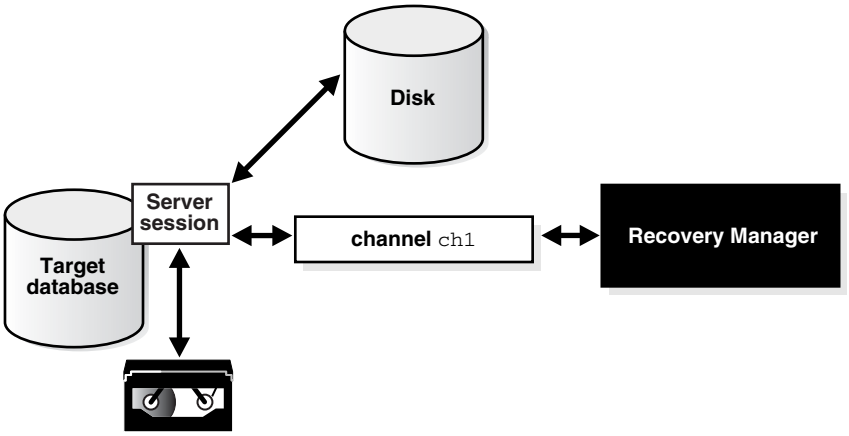
The RMAN client directs database server sessions to perform all backup and recovery tasks. What constitutes a session depends on the operating system. For example, on Linux, a server session corresponds to a server process, whereas on Windows it corresponds to a thread within the database service.

The RMAN client itself does not perform backup, restore, or recovery operations. When you connect the RMAN client to a target database, RMAN allocates server sessions on the target instance and directs them to perform the operations.

An RMAN **channel** represents one stream of data to a device, and corresponds to one database server session. The channel reads data into PGA memory, processes it, and writes it to the output device. See "Basic Concepts of RMAN Performance Tuning" on page 21-1 for a low-level description of how channels work.

Most RMAN commands are executed by channels, which must be either configured to persist across RMAN sessions, or manually allocated in each **RMAN session**. As illustrated in Figure 3–2, a channel establishes a connection from the RMAN client to a target or auxiliary database instance by starting a server session on the instance.

Figure 3–2 Channel Allocation



Channels and Devices

The RMAN-supported device types are disk and **SBT** (system backup to tape). An SBT device is controlled by a third-party **media manager**. Typically, SBT devices are tape libraries and tape drives.

If you use a disk channel for a backup, then the channel creates the backup on disk in the file name space of the target database instance creating the backup. You can make a backup on any device that can store a datafile. RMAN does not call a media manager when making disk backups.

To create backups on nondisk media, you must use media management software such as Oracle Secure Backup and allocate channels supported by this software. RMAN contacts the media manager whenever the channel type allocated is not disk. How and when the SBT channels cause the media manager to allocate resources is vendor-specific. Some media managers allocate resources when you issue the command; others do not allocate resources until you open a file for reading or writing.

> **See Also:** "Configuring the Default Device for Backups: Disk or SBT" on page 5-3

Automatic and Manual Channels

You can use the `CONFIGURE CHANNEL` command to configure channels for use with disk or tape across RMAN sessions. This technique is known as **automatic channel allocation**. RMAN comes preconfigured with one `DISK` channel that you can use for backups to disk.

When you run a command that can use automatic channels, RMAN automatically allocates the channels with the options you specified in the `CONFIGURE` command. For the `BACKUP` command, RMAN allocates only the type of channel required to back up to the specified media. For the `RESTORE` command and **RMAN maintenance commands**, RMAN allocates all necessary channels for the device types required to execute the command. RMAN determines the names for automatic channels.

You also have the option of manually allocating channels. Each manually allocated channel uses a separate connection to the database. When you manually allocate a channel, you give it a user-defined name such as `dev1` or `ch2`.

The number of channels available for use with a device when you run a command determines whether RMAN will read from or write to this device in parallel while carrying out the command. In parallelism, the backup of the files is performed by more than one channel. Each channel may back up more than one file, but unless a **multisection backup** is performed, no file is backed up by more than one channel.

See Also:

- *Oracle Database Backup and Recovery Reference* for `ALLOCATE CHANNEL` syntax
- *Oracle Database Backup and Recovery Reference* on `ALLOCATE CHANNEL FOR MAINTENANCE`
- "Configuring Channels for Disk" on page 5-5 and "Configuring SBT Channels for Use with a Media Manager" on page 5-12

RMAN Repository

The **RMAN repository** is the collection of metadata about the target databases that RMAN uses for backup, recovery, and maintenance. RMAN always stores its metadata in the control file. The version of this metadata in the control file is the authoritative record of RMAN backups of your database. This is one reason why protecting your control file is a important part of your backup strategy. RMAN can conduct all necessary backup and recovery operations using just the control file to store the RMAN repository information, and maintains all records necessary to meet your configured retention policy.

You can also create a **recovery catalog**, which is a repository of RMAN metadata stored in an Oracle database schema. The control file has finite space for records of backup activities, whereas a recovery catalog can store a much longer history. You can simplify backup and recovery administration by creating a single recovery catalog that contains the RMAN metadata for all of your databases.

The owner of a recovery catalog can grant or revoke restricted access to the catalog to other database users. Each restricted user has full read/write access to his own metadata, which is called a **virtual private catalog**. When one or more virtual private catalogs exist in a database, the database contains just one set of catalog tables. These tables are owned by the base recovery catalog owner. The owner of the base recovery catalog controls which databases each virtual private catalog user can access.

Some RMAN features only function when you use a recovery catalog. For example, you can create a **stored script** in the recovery catalog and use this script to execute RMAN jobs. Other RMAN commands are specifically related to managing the recovery catalog and so are not available (and not needed) if RMAN is not connected to a recovery catalog.

The recovery catalog is maintained solely by RMAN. A target database instance never accesses the catalog directly. RMAN propagates information about the database structure, archived redo logs, backup sets, and datafile copies into the recovery catalog from the target database control file after any operation that updates the repository, and also before certain operations.

> **See Also:** Chapter 11, "Maintaining RMAN Backups and Repository Records" and Chapter 12, "Managing a Recovery Catalog"

Media Management

Oracle's Media Management Layer (MML) API lets third-party vendors build a **media manager**, software that works with RMAN and the vendor's hardware to allow backups to sequential media devices such as tape drives. A media manager handles loading, unloading and labeling of sequential media such as tapes. You must install media manager software to use RMAN with sequential media devices.

When backing up or restoring, the RMAN client connects to a target database instance and directs the instance to send requests to its media manager. No direct communication occurs between the RMAN client and media manager.

RMAN Interaction with a Media Manager

Before performing backup or restore to a media manager, you must allocate one or more channels to handle the communication with the media manager. You can also configure default channels for use with the media manager, which will be applied for all backup and recovery tasks that use the media manager where you do not explicitly allocate channels.

RMAN does not issue specific commands to load, label, or unload tapes. When backing up, RMAN gives the media manager a stream of bytes and associates a unique name with this stream. When RMAN needs to restore the backup, it asks the media manager to retrieve the byte stream. All details of how and where that stream is stored are handled entirely by the media manager. For example, the media manager labels and keeps track of the tape and names of files on each tape, and automatically loads and unloads tapes, or signals an operator to do so.

Some media managers support **proxy copy** functionality, in which they handle the entire data movement between datafiles and the backup devices. Such products may use technologies such as high-speed connections between storage and media subsystems to reduce load on the primary database server. RMAN provides a list of files requiring backup or restore to the media manager, which in turn makes all decisions regarding how and when to move the data.

> **See Also:** "Configuring SBT Channels for Use with a Media Manager" on page 5-12

Oracle Secure Backup

Oracle Secure Backup is a media manager that provides reliable and secure data protection through file system backup to tape. All major tape drives and tape libraries in SAN, Gigabit Ethernet, and SCSI environments are supported.

Although Oracle Secure Backup has no specialized knowledge of database backup and recovery algorithms, it can serve as a media management layer for RMAN through the SBT interface. In this capacity, Oracle Secure Backup provides the same services for RMAN as other supported third-party SBT libraries. Oracle Secure Backup has some features, however, that are not available in other media managers.

> **See Also:** *Oracle Secure Backup Administrator's Guide* to learn how to use Oracle Secure Backup

Backup Solutions Program

The Oracle Backup Solutions Program (BSP), part of the Oracle Partner Program, is a group of media manager vendors whose products are compliant with Oracle's MML specification. Several products may be available for your platform from media management vendors. For more information, you can contact your Oracle representative for a list of available products, contact individual vendors to ask them if they participate, or access the Backup Solutions Program Web site at:

`http://www.oracle.com/technology/deploy/availability`

Note that Oracle does not certify media manager vendors for compatibility with RMAN. Questions about availability, version compatibility, and functionality can only be answered by the media manager vendor, not Oracle.

Flash Recovery Area

The component that creates different backup and recovery-related files have no knowledge of each other or of the size of the file systems where they store their data. With automatic disk-based backup and recovery, you can create a **flash recovery area** (also simply called the **recovery area**), which automates management of backup-related files.

A flash recovery area minimizes the need to manually manage disk space for backup-related files and balance the use of space among the different types of files. In this way, a flash recovery area simplifies the ongoing administration of your database. Oracle recommends that you enable a recovery area to simplify backup management.

When you create a recovery area, you choose a location on disk and an upper bound for storage space. You also set a **backup retention policy** that governs how long backup files are needed for recovery. The database manages the storage used for backups, archived redo logs, and other recovery-related files for the database within this space. Files no longer needed are eligible for deletion when RMAN needs to reclaim space for new files.

> **See Also:** "Configuring the Flash Recovery Area" on page 5-13 to learn about the flash recovery area and how to configure it

RMAN in a Data Guard Environment

When using RMAN in a Data Guard environment, a recovery catalog is required. The recovery catalog can store the metadata for all primary and standby databases.

A database in a Data Guard environment is uniquely identified by means of the `DB_UNIQUE_NAME` parameter in the initialization parameter file. The `DB_UNIQUE_NAME` must be unique across all the databases with the same DBID for RMAN to work correctly in a Data Guard environment.

> **See Also:** *Oracle Data Guard Concepts and Administration* to learn how to use RMAN in a Data Guard environment

RMAN Configuration in a Data Guard Environment

To simplify ongoing use of RMAN for backup and recovery, you can set a number of persistent configuration settings for each primary and **physical standby database** in a Data Guard environment. These settings control many aspects of RMAN behavior. For example, you can configure the **backup retention policy**, default destinations for backups to tape or disk, default backup device type, and so on.

You can use the CONFIGURE command with the FOR DB_UNIQUE_NAME clause to create a persistent configuration for a database in a Data Guard environment without connecting to the standby database or primary database as TARGET. For example, you connect RMAN to the recovery catalog, run the SET DBID command, and then can create a configuration for a physical standby database before its creation so that the RMAN configuration applies when the database is created.

RMAN updates the control file of the database when connected to it as TARGET during a recovery catalog **resynchronization**. If you use FOR DB_UNIQUE_NAME for a database without being connected as TARGET to this database, however, then RMAN changes configurations in the recovery catalog only.

> **See Also:** "Configuring RMAN in a Data Guard Environment" on page 5-35

RMAN File Management in a Data Guard Environment

RMAN uses a recovery catalog to track filenames for all database files in a Data Guard environment. The catalog also records where the online redo log files, standby redo log files, tempfiles, archived redo log files, backup sets, and image copies are created.

Interchangeability of Backups in a Data Guard Environment

RMAN commands use the recovery catalog metadata to behave transparently across different physical databases in the Data Guard environment. For example, you can back up a tablespace on a physical standby database and restore and recover it on the primary database. Similarly, you can back up a tablespace on a primary database and restore and recover it on a physical standby database.

> **Note:** Backups of logical standby databases are not usable at the primary database.

Backups of standby control files and nonstandby control files are interchangeable. For example, you can restore a standby control file on a primary database and a primary control file on a physical standby database. This interchangeability means that you can offload control file backups to one database in a Data Guard environment. RMAN automatically updates the filenames for database files during restore and recovery at the databases.

Association of Backups in a Data Guard Environment

The recovery catalog tracks the files in the Data Guard environment by associating every database file or backup file with a DB_UNIQUE_NAME. The database that creates a file is associated with the file. For example, if RMAN backs up the database with the unique name of standby1, then standby1 is associated with this backup. A backup remains associated with the database that created it unless you use the CHANGE . . . RESET DB_UNIQUE_NAME to associate the backup with a different database.

Accessibility of Backups in a Data Guard Environment

The accessibility of a backup is different from its association. In a Data Guard environment, the recovery catalog considers disk backups as accessible only to the database with which it is associated, whereas tape backups created on one database are accessible to all databases. If a backup file is not associated with any database, then the row describing it in the recovery catalog view shows null for the SITE_KEY

column. By default, RMAN associates files whose SITE_KEY is null with the database to which it is connected as TARGET.

RMAN commands such as BACKUP, RESTORE, and CROSSCHECK work on any accessible backup. For example, for a RECOVER COPY operation, RMAN considers only image copies that are associated with the database as eligible to be recovered. RMAN considers the incremental backups on disk and tape as eligible to recover the image copies. In a database recovery, RMAN considers only the disk backups associated with the database and all files on tape as eligible to be restored.

To illustrate the differences in backup accessibility, assume that databases prod and standby1 reside on different hosts. RMAN backs up datafile 1 on prod to /prmhost/disk1/df1.dbf on the production host and also to tape. RMAN backs up datafile 1 on standby1 to /sbyhost/disk2/df1.dbf on the standby host and also to tape. If RMAN is connected to database prod, then you cannot use RMAN commands to perform operations with the /sbyhost/disk2/df1.dbf backup located on the standby host. However, RMAN does consider the tape backup made on standby1 as eligible to be restored.

Note: You can FTP a backup from a standby host to a primary host or vice versa, connect as TARGET to the database on this host, and then CATALOG the backup. After a file is cataloged by the target database, the file is associated with the target database.

See Also:

- *Oracle Data Guard Concepts and Administration* to learn how to perform RMAN backup and recovery in a Data Guard environment
- "Maintenance Commands in a Data Guard Environment" on page 11-2
- Chapter 12, "Managing a Recovery Catalog" to learn how to manage a recovery catalog in a Data Guard environment
- *Oracle Database Backup and Recovery Reference* for CONFIGURE syntax

4

Starting and Interacting with the RMAN Client

This chapter explains how to start the RMAN command-line interface and make database connections. This chapter contains the following topics:

- Starting and Exiting RMAN
- Specifying the Location of RMAN Output
- Setting Globalization Support Environment Variables for RMAN
- Entering RMAN Commands
- Making Database Connections with RMAN
- Using the RMAN Pipe Interface

Starting and Exiting RMAN

The RMAN executable is automatically installed with the database and is typically located in the same directory as the other database executables. For example, the RMAN client on Linux is located in `$ORACLE_HOME/bin`. You have the following basic options for starting RMAN:

- Start the RMAN executable at the operating system command line without specifying any connection options, as in the following example:

  ```
  % rman
  ```

- Start the RMAN executable at the operating system command line while connecting to a **target database** and, possibly, a **recovery catalog**, as in the following examples:

  ```
  % rman TARGET /                       # operating system authentication
  % rman TARGET SYS@prod NOCATALOG      # RMAN prompts for SYS password
  % rman TARGET / CATALOG rco@catdb     # RMAN prompts for rco password
  ```

> **Note:** Most RMAN commands require that RMAN connect to at least a target database to perform useful work. See "Making Database Connections with RMAN" on page 4-7 for more details on connecting RMAN to different types of databases.

To quit RMAN and terminate the program, enter `EXIT` or `QUIT` at the RMAN prompt:

```
RMAN> EXIT
```

See Also: *Oracle Database Backup and Recovery Reference* for RMAN command-line syntax

Specifying the Location of RMAN Output

By default, RMAN writes command output to standard output. To redirect output to a log file, enter the `LOG` parameter on the command line when starting RMAN, as in the following example:

```
% rman LOG /tmp/rman.log
```

In this case, RMAN displays command input but does not display the RMAN output. The easiest way to send RMAN output both to a log file and to standard output is to use the Linux `tee` command or its equivalent. For example, the following technique enables both input and output to be visible in the RMAN command-line interface:

```
% rman | tee rman.log
RMAN>
```

See Also: *Oracle Database Backup and Recovery Reference* to learn about RMAN command-line options

Setting Globalization Support Environment Variables for RMAN

Before invoking RMAN, it may be useful to set the `NLS_DATE_FORMAT` and `NLS_LANG` environment variables. These variables determine the format used for the time parameters in RMAN commands such as `RESTORE`, `RECOVER`, and `REPORT`.

The following example shows typical language and date format settings:

```
NLS_LANG=american
NLS_DATE_FORMAT='Mon DD YYYY HH24:MI:SS'
```

If you are going to use RMAN to connect to an unmounted database and mount the database later while RMAN is still connected, then set the `NLS_LANG` environment variable so that it also specifies the character set used by the database.

A database that is not mounted assumes the default character set, which is `US7ASCII`. If your character set is different from the default, then RMAN returns errors after the database is mounted. For example, if the character set is `WE8DEC`, then you can set the `NLS_LANG` variable as follows:

```
NLS_LANG=american_america.we8dec
```

`NLS_LANG` and `NLS_DATE_FORMAT` must be set for `NLS_DATE_FORMAT` to be used.

See Also:

- *Oracle Database Reference* for more information about the `NLS_LANG` and `NLS_DATE_FORMAT` parameters
- *Oracle Database Globalization Support Guide*

Entering RMAN Commands

You can enter RMAN command either directly from the RMAN prompt or read them in from a text file.

This section contains the following topics:

- Entering RMAN Commands at the RMAN Prompt

- Using Command Files with RMAN
- Entering Comments in RMAN Command Files
- Using Substitution Variables in Command Files
- Checking RMAN Syntax

Entering RMAN Commands at the RMAN Prompt

When the RMAN client is ready for your commands, it displays the command prompt, as in this example:

```
RMAN>
```

Enter commands for RMAN to execute. For example:

```
RMAN> CONNECT TARGET
RMAN> BACKUP DATABASE;
```

Most RMAN commands take a number of parameters and must end with a semicolon. Some commands, such as `STARTUP`, `SHUTDOWN`, and `CONNECT`, can be used with or without a semicolon.

When you enter a line of text that is not a complete command, RMAN prompts for continuation input with a line number. For example:

```
RMAN> BACKUP DATABASE
2> INCLUDE CURRENT
3> CONTROLFILE
4> ;
```

Using Command Files with RMAN

For repetitive tasks, you can create a text file containing RMAN commands, and start the RMAN client with the `@` argument, followed by a filename. For example, create a text file `cmdfile1` in the current directory containing one line of text as shown here:

```
BACKUP DATABASE PLUS ARCHIVELOG;
```

You can run this command file from the command line as shown in this example, and the command contained in it is executed:

```
% rman TARGET / @cmdfile1
```

After the command completes, RMAN exits.

You can also use the `@` command at the RMAN command prompt to execute the contents of a command file during an **RMAN session**. RMAN reads the file and executes the commands in it. For example:

```
RMAN> @cmdfile1
```

After the command file contents have been executed, RMAN displays the following message:

```
RMAN>  **end-of-file**
```

Unlike the case where a command file is executed from the operating system command line, RMAN does not exit.

> **See Also:** *Oracle Database Backup and Recovery Reference* for RMAN command-line syntax

Entering Comments in RMAN Command Files

The comment character in RMAN is a pound sign (#). All text from the pound sign to the end of the line is ignored. For example, the following command file contents backs up the database and archived redo log files and includes comments:

```
# Command file name: mybackup.rman
# The following command backs up the database
BACKUP DATABASE;
# The following command backs up the archived redo logs
BACKUP ARCHIVELOG ALL;
```

The following example shows that you can break a single RMAN command across multiple lines:

```
RMAN> BACKUP # this is a comment
2> SPFILE;
```

Using Substitution Variables in Command Files

When running a command file, you can specify one or more values in a `USING` clause for use in substitution variables in a command file. In this way, you can make your command files dynamic.

As in SQL*Plus, `&1` indicates where to place the first value, `&2` where to place the second value, and so on. The substitution variable syntax is `&`*`integer`* followed by an optional period, for example, `&1.3`. The optional period is part of the variable and replaced with the value, thus enabling the substitution text to be immediately followed by another integer. For example, if you pass the value `mybackup` to a command file containing the variable `&1.3`, then the result of the substitution is `mybackup3`.

The following procedure explains how to create and use a dynamic shell script that calls a command file containing substitution variables.

To create and use a dynamic shell script:

1. Create an RMAN command file that uses substitution variables.

   The following example shows the contents of a command file named `quarterly_backup.cmd`, which is run every quarter. The script uses substitution variables for the name of the tape set, for a string in the `FORMAT` specification, and for the name of the restore point to be created.

   ```
   # quarterly_backup.cmd
   CONNECT TARGET /
   RUN
   {
     ALLOCATE CHANNEL c1
       DEVICE TYPE sbt
       PARMS 'ENV=(OB_MEDIA_FAMILY=&1)';
     BACKUP DATABASE
       TAG &2
       FORMAT '/disk2/bck/&1%U.bck'
       KEEP FOREVER
       RESTORE POINT &3;
   }
   EXIT;
   ```

2. Create a shell script that you can use to run the RMAN command file created in the previous step.

The following example creates a shell script named `runbackup.sh`. The example creates shell variables for the format and restore point name and accepts the values for these variables as command-line arguments to the script.

```
#!/bin/tcsh
# name: runbackup.sh
# usage: use the tag name and number of copies as arguments
set media_family = $argv[1]
set format = $argv[2]
set restore_point = $argv[3]
rman @'/disk1/scripts/whole_db.cmd' USING $media_family $format $restore_point
```

3. Execute the shell script created in the previous step, specifying the desired arguments on the command line.

   The following example runs the `runbackup.sh` shell script and passes it `archival_backup` as the media family name, `bck0906` as the format string, and `FY06Q3` as the restore point name.

   ```
   % runbackup.sh archival_backup bck0906 FY06Q3
   ```

   See Also: *Oracle Database Backup and Recovery Reference* for `@` syntax

Checking RMAN Syntax

You may want to test RMAN commands for syntactic correctness without executing them. Use the command-line argument `CHECKSYNTAX` to start the RMAN client in a mode in which it only parses the commands you enter and returns an `RMAN-00558` error for commands that are not legal RMAN syntax.

See Also: *Oracle Database Backup and Recovery Reference* to learn about the `CHECKSYNTAX` command-line option

Checking RMAN Syntax at the Command Line

You can check the syntax of RMAN commands interactively without actually executing the commands.

To check RMAN syntax at the command line:

1. Start RMAN with the `CHECKSYNTAX` parameter.

   For example, enter the following commands:

   ```
   % rman CHECKSYNTAX
   ```

2. Enter the RMAN commands to be tested.

   The following shows a sample interactive session, with user-entered text in bold.

   ```
   RMAN> run [ backup database; ]

   RMAN-00571: ===========================================================
   RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
   RMAN-00571: ===========================================================
   RMAN-00558: error encountered while parsing input commands
   RMAN-01006: error signalled during parse

    RMAN-02001: unrecognized punctuation symbol "["

   RMAN> run { backup database; }
   ```

```
The command has no syntax errors

RMAN>
```

Checking RMAN Syntax in Command Files

To test commands in a command file, start RMAN with the CHECKSYNTAX parameter and use the @ command to name the command file to be passed.

To test commands in a command file:

1. Use a text editor to create a command file.

   Assume that you create the /tmp/goodcmdfile with the following contents:

   ```
   # command file with legal syntax
   RESTORE DATABASE;
   RECOVER DATABASE;
   ```

   Assume that you create another command file, /tmp/badcmdfile, with the following contents:

   ```
   # command file with bad syntax commands
   RESTORE DATABASE
   RECOVER DATABASE
   ```

2. Run the command file from the RMAN prompt in the following format, where *filename* is the name of the command file:

   ```
   % rman CHECKSYNTAX @filename
   ```

   The following example shows the output when you run /tmp/goodcmdfile with CHECKSYNTAX:

   ```
   RMAN> # command file with legal syntax
   2> restore database;
   3> recover database;
   4>
   The cmdfile has no syntax errors

   Recovery Manager complete.
   ```

   In contrast, the following example shows the output you run /tmp/badcmdfile with CHECKSYNTAX:

   ```
   RMAN> #command file with syntax error
   2> restore database
   3> recover
   RMAN-00571: ===========================================================
   RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS===============
   RMAN-00571: ===========================================================
   RMAN-00558: error encountered while parsing input commands
   RMAN-01005: syntax error: found "recover": expecting one of: "archivelog,
         channel, check, controlfile, clone, database, datafile, device,
         from, force, high, (, preview, ;, skip, spfile, standby, tablespace,
         until, validate"
   RMAN-01007: at line 3 column 1 file: /tmp/badcmdfile
   ```

As explained in "Using Substitution Variables in Command Files" on page 4-4, you make your command files dynamic by including substitution variables. When you check the syntax of a command file that contains substitution variables, RMAN

prompts you to enter values. Example 4–1 illustrates what happens you enter invalid values when checking the syntax of a dynamic command file. The text in bold indicates text entered as the prompt.

Example 4–1 Checking the Syntax of a Command File with Bad Syntax

```
RMAN> CONNECT TARGET *
2> BACKUP TAG
Enter value for 1: mybackup
abc COPIES
Enter value for 2: mybackup
abc
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-00558: error encountered while parsing input commands
RMAN-01009: syntax error: found "identifier": expecting one of: "integer"
RMAN-01008: the bad identifier was: mybackup
RMAN-01007: at line 2 column 25 file: /tmp/whole_db.cmd
```

RMAN indicates a syntax error because the string `mybackup` is not a valid argument for `COPIES`.

Making Database Connections with RMAN

This section explains how to connect the RMAN client to target databases. It contains the following topics:

- About RMAN Database Connections
- Connecting RMAN to an Auxiliary Database
- Making RMAN Database Connections Within Command Files
- Diagnosing RMAN Connection Problems

About RMAN Database Connections

To perform useful work, the RMAN client must connect to a database. The following table describes the types of database connections that you can make with RMAN.

Table 4–1 Overview of RMAN Database Connections

Type of Database Connection	Keyword	Description
target database	`TARGET`	A database to be backed up or restored by RMAN
recovery catalog database	`CATALOG`	A database that provides an optional backup store for the RMAN repository in addition to the control file.
auxiliary instance or **auxiliary database**	`AUXILIARY`	A **physical standby database**, or a database instance created for performing a specific task such as creating a **duplicate database**, transporting tablespaces, or performing **tablespace point-in-time recovery (TSPITR)**. For many tasks that use an auxiliary database, RMAN creates an automatic auxiliary instance for use during the task, connects to it, performs the task, and then destroys it when the task is completed. You do not give any explicit command to connect to automatic auxiliary instances.

Authentication for RMAN Database Connections

RMAN connections to a database are specified and authenticated in the same way as SQL*Plus connections to a database. The only difference is that RMAN connections to a target or auxiliary database require the SYSDBA privilege. The AS SYSDBA keywords are implied for target and auxiliary connections and cannot be explicitly specified.

A SYSDBA privilege is not required when connecting to the recovery catalog. Note that you must grant the RECOVERY_CATALOG_OWNER role to the catalog schema owner.

> **See Also:** *Oracle Database Administrator's Guide* to learn about database connection options when using SQL*Plus

Authentication for RMAN Database Connections Using the Operating System To connect to a database using operating system authentication, you must set the environment variable specifying the Oracle SID. For example, to set the SID to prod in some UNIX shells, you would enter:

```
% ORACLE_SID=prod; export ORACLE_SID
```

A special operating system groups controls SYSDBA connections when using operating system authentication. This group is generically referred to as OSDBA. The group is created and assigned a specific name as part of the database installation process. The specific name varies depending upon your operating system.

If the current operating system user is a member of the OSDBA group, and if the Oracle SID is set, then RMAN can connect to this database with SYSDBA privileges as follows:

```
% rman TARGET /
```

Authentication for RMAN Database Connections Using a Password File If a database uses a **password file**, then RMAN can use a password to connect to this database. Use a password file for either local or remote access. You must use a password file if you are connecting remotely as SYSDBA with a net service name.

> **Caution:** Good security practice requires that passwords should not be entered in plain text on the command line. You should enter passwords in RMAN only when requested by an RMAN prompt. See *Oracle Database Security Guide* to learn about password protection.

You can start RMAN without a password in the connect string, as in this example:

```
% rman TARGET SYS@prod

target database Password: password
connected to target database: PROD1 (DBID=39525561)
```

RMAN prompts for a password and does not echo the characters.

Making RMAN Database Connections from the Operating System Command Line

To connect to a target database from the operating system command line, enter the rman command followed by the connection information. You can begin executing commands after the RMAN prompt is displayed.

In the examples in this chapter, the generic values have the meanings shown in Table 4–2.

Table 4–2 Values in Examples

Value Used in Example	Meaning
SYS	User with SYSDBA privileges
prod	The net service name for the target database
rco	User that owns the recovery catalog schema. This is a user defined in the recovery catalog database that has been granted the RECOVERY_CATALOG_OWNER role.
catdb	The net service name for the recovery catalog database
aux	The net service name for an auxiliary instance

Example 4–2 illustrates a connection to a target database that uses operating system authentication. The NOCATALOG option indicates that a recovery catalog will not be used in the session.

Example 4–2 Connecting to a Target Database from the System Prompt

```
% rman TARGET / NOCATALOG

connected to target database: PROD (DBID=39525561)
using target database control file instead of recovery catalog

RMAN>
```

Example 4–3 illustrates a connection to a target database that uses Oracle Net authentication. RMAN prompts for the password.

Example 4–3 Connecting to a Target Database from the System Prompt

```
% rman TARGET SYS@prod NOCATALOG

target database Password: password
connected to target database: PROD (DBID=39525561)

RMAN>
```

Use the CATALOG keyword to connect to a recovery catalog. Example 4–4 illustrates a connection that uses Oracle Net authentication for the target and recovery catalog databases. In both cases RMAN prompts for a password.

Example 4–4 Connecting to Target and Catalog Databases from the System Prompt

```
% rman TARGET SYS@prod CATALOG rco@catdb

target database Password: password
connected to target database: PROD (DBID=39525561)
recovery catalog database Password: password
connected to recovery catalog database

RMAN>
```

You can also start RMAN without specifying NOCATALOG or CATALOG. If you do not specify NOCATALOG on the command line, and if you do not specify CONNECT CATALOG after RMAN has started, then RMAN defaults to NOCATALOG mode the first time that you run a command that requires the use of the RMAN repository.

> **Note:** After you have executed a command that uses the RMAN repository in NOCATALOG mode, you must exit and restart RMAN to be able to connect to a recovery catalog.

If you connect to the target database on the operating system command line, then you can begin executing commands after the RMAN prompt is displayed.

Making Database Connections from the RMAN Prompt

If you start RMAN without connecting to a target database, then you must issue a CONNECT TARGET command at the RMAN prompt to connect to a target database and begin performing useful work.

To make a database connection from the RMAN prompt:

1. On the operating system command line, start the RMAN client without making a database connection. For example, enter rman as follows:

   ```
   % rman
   RMAN>
   ```

2. At the RMAN prompt, enter one or more CONNECT commands.

   The following example connects to a target database using operating system authentication:

   ```
   RMAN> CONNECT TARGET /
   ```

   The following alternative example connects to a target database and then a recovery catalog. The target connection uses operating system authentication, whereas the catalog database connection uses Oracle Net authentication. RMAN prompts for the password of the recovery catalog user.

   ```
   RMAN> CONNECT TARGET /
   RMAN> CONNECT CATALOG rco@catdb

   recovery catalog database Password: password
   connected to recovery catalog database
   ```

   The following example connects to a target database with database-level credentials. RMAN prompts for the SYS password.

   ```
   % rman
   RMAN> CONNECT TARGET SYS@prod

   target database Password: password
   connected to target database: PROD (DBID=39525561)
   ```

> **See Also:** *Oracle Database Backup and Recovery Reference* to learn about the CONNECT command

Connecting RMAN to an Auxiliary Database

To use the DUPLICATE command, you need to connect to an **auxiliary instance**. To perform RMAN **tablespace point-in-time recovery (TSPITR)**, you may also need to connect to an auxiliary instance.

> **Note:** When you use the `DUPLICATE ... FROM ACTIVE DATABASE` command, a net service name is required. See "Step 1: Create an Oracle Password File for the Auxiliary Instance" on page 23-7 for more details.

The form of an auxiliary connection is identical to a target database connection, except that you use the `AUXILIARY` keyword instead of the `TARGET` keyword. Example 4–5 connects to a target database and auxiliary instance from the RMAN prompt.

Example 4–5 Connecting to the Target and Catalog Databases from the RMAN Prompt

```
% rman
RMAN> CONNECT TARGET /
RMAN> CONNECT AUXILIARY SYS@aux

auxiliary database Password: password
connected to auxiliary database: PROD (DBID=30472568)
```

See Also:

- Chapter 23, "Duplicating a Database" for more details on using the `DUPLICATE` command
- Chapter 20, "Performing RMAN Tablespace Point-in-Time Recovery (TSPITR)" for more details on performing TSPITR

Making RMAN Database Connections Within Command Files

If you create an RMAN command file which uses a `CONNECT` command with database level credentials (user name and password), then anyone with read access to this file can learn the password. There is no secure way to incorporate a `CONNECT` string with a password into a command file.

If you create an RMAN command file which uses a `CONNECT` command, then RMAN does not echo the connect string when you run the command file with the `@` command. This behavior prevents connect strings from appearing in any log files that contain RMAN output. For example, suppose you create a command file `listbkup.rman` as follows:

```
cat > listbkup.rman << EOF
CONNECT TARGET /
LIST BACKUP;
EOF
```

You execute this script by running RMAN with the `@` command line option as follows:

```
% rman @listbkup.rman
```

When the command file executes, RMAN replaces the connection string with an asterisk, as shown in the following output:

```
RMAN> CONNECT TARGET *
2> LIST BACKUP;
3>
connected to target database: RDBMS (DBID=771530996)

using target database control file instead of recovery catalog

List of Backup Sets
```

```
==================
. . .
```

Diagnosing RMAN Connection Problems

When diagnosing errors RMAN encounters in connecting to the target, catalog and auxiliary databases, using SQL*Plus to connect to the databases directly can reveal underlying problems with the connection information or the databases.

Diagnosing Target and Auxiliary Database Connection Problems

RMAN always connects to target and auxiliary databases using the `SYSDBA` privilege. Thus, when using SQL*Plus to diagnose connection problems to the target or auxiliary databases, request a `SYSDBA` connection to reproduce RMAN behavior.

For example, suppose that the following RMAN command encountered connection errors:

```
RMAN> CONNECT TARGET /
```

You could reproduce the preceding connection attempt with the SQL*Plus command as follows:

```
SQL> CONNECT / AS SYSDBA
```

Diagnosing Recovery Catalog Connection Problems

When RMAN connects to the recovery catalog database, it does not use the `SYSDBA` privilege. So, when you are using SQL*Plus to diagnose connection problems to the recovery catalog database, you must enter the database connect string exactly as it was entered into RMAN. Do not specify `AS SYSDBA`.

Using the RMAN Pipe Interface

The RMAN pipe interface is an alternative method for issuing commands to RMAN and receiving the output from those commands. With this interface, RMAN obtains commands and sends output by using the `DBMS_PIPE` PL/SQL package instead of the operating system shell. Using this interface, it is possible to write a portable programmatic interface to RMAN.

The pipe interface is invoked by using the `PIPE` command-line parameter for the RMAN client. RMAN uses two private pipes: one for receiving commands and the other for sending output. The names of the pipes are derived from the value of the `PIPE` parameter. For example, you can invoke RMAN with the following command:

```
% rman PIPE abc TARGET /
```

RMAN opens the two pipes in the target database: `ORA$RMAN_ABC_IN`, which RMAN uses to receive user commands, and `ORA$RMAN_ABC_OUT`, which RMAN uses to send all output back to RMAN. All messages on both the input and output pipes are of type `VARCHAR2`.

Note that RMAN does not permit the pipe interface to be used with public pipes, because they are a potential security problem. With a public pipe, any user who knows the name of the pipe can send commands to RMAN and intercept its output.

If the pipes are not already initialized, then RMAN creates them as private pipes. If you want to put commands on the input pipe before starting RMAN, you must first create the pipe by calling `DBMS_PIPE.CREATE_PIPE`. Whenever a pipe is not

explicitly created as a private pipe, the first access to the pipe automatically creates it as a public pipe, and RMAN returns an error if it is told to use a public pipe.

> **Note:** If multiple RMAN sessions can run against the target database, then you must use unique pipe names for each **RMAN session**. The `DBMS_PIPE.UNIQUE_SESSION_NAME` function is one method that you can use to generate unique pipe names.

Executing Multiple RMAN Commands In Succession Through a Pipe: Example

This scenario assumes that the application controlling RMAN wants to run multiple commands in succession. After each command is sent down the pipe and executed and the output returned, RMAN will pause and wait for the next command.

To execute RMAN commands through a pipe:

1. Start RMAN by connecting to a target database (required) and specifying the `PIPE` option. For example, issue:

   ```
   % rman PIPE abc TARGET /
   ```

   You can also specify the `TIMEOUT` option, which forces RMAN to exit automatically if it does not receive any input from the input pipe in the specified number of seconds. For example, enter:

   ```
   % rman PIPE abc TARGET / TIMEOUT 60
   ```

2. Connect to the target database and put the desired commands on the input pipe by using `DBMS_PIPE.PACK_MESSAGE` and `DBMS_PIPE.SEND_MESSAGE`. In pipe mode, RMAN issues message `RMAN-00572` when it is ready to accept input instead of displaying the standard RMAN prompt.
3. Read the RMAN output from the output pipe by using `DBMS_PIPE.RECEIVE_MESSAGE` and `DBMS_PIPE.UNPACK_MESSAGE`.
4. Repeat steps 2 and 3 to execute further commands with the same RMAN instance that was started in step 1.
5. If you used the `TIMEOUT` option when starting RMAN, then RMAN terminates automatically after not receiving any input for the specified length of time. To force RMAN to terminate immediately, send the `EXIT` command.

Executing RMAN Commands In a Single Job Through a Pipe: Example

This scenario assumes that the application controlling RMAN wants to run one or more commands as a single job. After running the commands that are on the pipe, RMAN will exit.

To execute RMAN commands in a single job through a pipe:

1. After connecting to the target database, create a pipe (if it does not already exist under the name `ORA$RMAN_`*pipe*`_IN`).
2. Put the desired commands on the input pipe. In pipe mode, RMAN issues message `RMAN-00572` when it is ready to accept input instead of displaying the standard RMAN prompt.
3. Start RMAN with the `PIPE` option, and specify `TIMEOUT 0`. For example, enter:

   ```
   % rman PIPE abc TARGET / TIMEOUT 0
   ```

4. RMAN reads the commands that were put on the pipe and executes them by using `DBMS_PIPE.PACK_MESSAGE` and `DBMS_PIPE.SEND_MESSAGE`. When it has exhausted the input pipe, RMAN exits immediately.
5. Read RMAN output from the output pipe by using `DBMS_PIPE.RECEIVE_MESSAGE` and `DBMS_PIPE.UNPACK_MESSAGE`.

See Also: *Oracle Database PL/SQL Packages and Types Reference* for documentation on the `DBMS_PIPE` package

5

Configuring the RMAN Environment

This chapter explains the most basic tasks involved in configuring RMAN. This chapter includes the following topics:

- Configuring the Environment for RMAN Backups
- Configuring RMAN to Make Backups to a Media Manager
- Configuring the Flash Recovery Area
- Configuring the Backup Retention Policy
- Configuring Backup Optimization
- Configuring an Archived Redo Log Deletion Policy
- Configuring Oracle Flashback Database and Restore Points
- Configuring RMAN in a Data Guard Environment

> **See Also:** Chapter 6, "Configuring the RMAN Environment: Advanced Topics" to learn about configuration options not covered in this chapter, including backup compression and encryption

Configuring the Environment for RMAN Backups

For most parameters required for backups, RMAN provides sensible defaults that enable you to perform basic backup and recovery. When implementing an RMAN-based backup strategy, you can use RMAN more effectively if you understand the more common configurations.

To simplify ongoing use of RMAN, you can set a number of persistent configuration settings for each target database. These settings control many aspects of RMAN behavior. For example, you can configure the backup retention policy, default destinations for backups, default backup device type, and so on. You can use the `SHOW` and `CONFIGURE` commands to view and change RMAN configurations.

This section explains what an RMAN configuration is and how you can use the `CONFIGURE` command to change RMAN default behavior for your backup and recovery environment. This section also introduces the major settings available to you and their most common values.

Note: If you plan to back up to tape, then refer to "Configuring RMAN to Make Backups to a Media Manager" on page 5-8 to learn how to set up RMAN for use with a media manager.

This section includes the following topics:

- Showing and Clearing Persistent RMAN Configurations
- Configuring the Default Device for Backups: Disk or SBT
- Configuring the Default Type for Backups: Backup Sets or Copies
- Configuring Channels
- Configuring Control File and Server Parameter File Autobackups

> **See Also:** *Oracle Database Backup and Recovery Reference* for `CONFIGURE` syntax

Showing and Clearing Persistent RMAN Configurations

You can use the `SHOW` command to display the current value of RMAN configured settings for the target database. You can also view whether these commands are currently set to their default values.

To view or change your CONFIGURE command settings:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Run the RMAN `SHOW` command.

   For example, run `SHOW ALL` as shown in Example 5–1 (sample output included). The output includes parameters you have changed and that are set to the default. The configuration is displayed as the series of RMAN commands required to re-create the configuration. You can save the output in a text file and use this command file to re-create the configuration on the same or a different database.

Example 5–1 SHOW ALL Command

```
SHOW ALL;

RMAN configuration parameters for database with db_unique_name PROD1 are:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 3 DAYS;
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE SBT_TAPE TO '%F'; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE DEVICE TYPE 'SBT_TAPE' PARALLELISM 2 BACKUP TYPE TO COMPRESSED BACKUPSET;
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS 'ENV=(OB_DEVICE=tape1)';
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/disk1/oracle/dbs/snapcf_ev.f'; # default
```

   You can also use the `SHOW` command with the name of a particular configuration. For example, you can view the retention policy and default device type as follows:

```
SHOW RETENTION POLICY;
SHOW DEFAULT DEVICE TYPE;
```

3. Optionally, use the `CONFIGURE ... CLEAR` command to return any configuration to its default value, as shown in the following examples:

```
CONFIGURE BACKUP OPTIMIZATION CLEAR;
CONFIGURE RETENTION POLICY CLEAR;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK CLEAR;
```

See Also: *Oracle Database Backup and Recovery Reference* for `SHOW` syntax

Configuring the Default Device for Backups: Disk or SBT

Backups for which no destination device type is specified are directed to the configured default device. RMAN is preconfigured to use disk as the default device type. No additional configuration is necessary.

You may need to change the default device type from disk to tape, or change it back from tape to disk. Table 5–1 shows the commands that configure the default device.

Table 5–1 Commands to Configure the Default Device Type

Command	Explanation
`CONFIGURE DEFAULT DEVICE TYPE TO DISK`	Specifies that backups should go to disk by default. If a recovery area is enabled, then the backup location defaults to the **flash recovery area**. Otherwise, the backup location defaults to an operating system-specific directory on disk. When backing up to disk, the logical block size of the database file must be an even multiple of the physical block size of the destination device. For example, a device of type DISK with a block size of 2 KB can only be used as a destination for backups of database files with logical block sizes of 2 KB, 4 KB, 6 KB, and so on. Most disk drives have physical block sizes of 512 bytes, so this limitation rarely affects backup to disk drives. Nevertheless, you can encounter this limitation when backing up to a writable DVD or a device that has a larger physical block size.
`CONFIGURE DEFAULT DEVICE TYPE TO sbt`	Specifies that backups should go to tape by default. "Configuring RMAN to Make Backups to a Media Manager" on page 5-8 explains how to set up RMAN for use with a media manager. After RMAN is able to communicate with the media manager, you can configure RMAN to make backups to tape and specify **SBT** as the default device type.

Note that you can always override the default device by using the `DEVICE TYPE` clause of the `BACKUP` command, as shown in the following examples:

```
BACKUP DEVICE TYPE sbt DATABASE;
BACKUP DEVICE TYPE DISK DATABASE;
```

To change the configured default device type:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Run the `SHOW ALL` command to show the currently configured default device.
3. Run the `CONFIGURE DEFAULT DEVICE TYPE` command, specifying either `TO DISK` or `TO sbt`.

See Also: *Oracle Database Backup and Recovery Reference* for more details on using the `BACKUP` command with the `DEVICE TYPE` clause

Configuring the Default Type for Backups: Backup Sets or Copies

The BACKUP command can create either backup sets or image copies. For disk, you can configure RMAN to create either backup sets or image copies as its default backup type with the CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO command. The default backup type for disk is an uncompressed **backup set**.

> **Note:** Because RMAN can write an **image copy** only to disk, the backup type for tape can only be a backup set.

RMAN can create backup sets using **binary compression**. You can configure RMAN to use compressed backup sets by default on a device type by specifying the COMPRESSED option in the BACKUP TYPE TO ... BACKUPSET clause. To disable compression, use the CONFIGURE DEVICE TYPE command with arguments specifying your other desired settings, but omit the COMPRESSED keyword.

To configure the default type of backup:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Configure backup sets or image copies as the default backup type.

   The following examples configure the backup type for disk backups to copies and backup sets:

   ```
   CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COPY; # image copies
   CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO BACKUPSET; # uncompressed
   ```

   The following examples configure compression for backup sets:

   ```
   CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COMPRESSED BACKUPSET;
   CONFIGURE DEVICE TYPE sbt BACKUP TYPE TO COMPRESSED BACKUPSET;
   ```

See Also:

- Backup Sets on page 7-3
- Making Compressed Backups on page 8-6

Configuring Channels

As explained in "RMAN Channels" on page 3-3, an RMAN **channel** is a connection to a database server session. RMAN uses channels to perform almost all RMAN tasks.

About Channel Configuration

Use the CONFIGURE CHANNEL command to configure options for disk or **SBT** channels. CONFIGURE CHANNEL takes the same options used to specify one-time options with the ALLOCATE CHANNEL command. You can configure generic channel settings for a device type, that is, a template that is used for any channels created based on configured settings for that device.

> **Note:** This section explains configuration of disk channels. To learn how to configure channels for tape, see "Configuring SBT Channels for Use with a Media Manager" on page 5-12.

Note that if you use CONFIGURE CHANNEL to specify generic channel settings for a device, any previous settings are discarded, even if the settings are not in conflict. For

example, after the second CONFIGURE CHANNEL command, which specifies only the FORMAT for configured disk channels, the MAXPIECESIZE for the disk channel is returned to its default value:

```
CONFIGURE CHANNEL DEVICE TYPE DISK MAXPIECESIZE 2G;
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT /tmp/%U;
```

Configuring Channels for Disk

By default, RMAN allocates one disk channel for all operations. You may wish to specify different options for this channel, for example, a new default location for backups. Example 5–2 configures RMAN to write disk backups to the /disk1 directory and specifies a nondefault format for the relative filename:

Example 5–2 Configuring a Nondefault Backup Location

```
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/disk1/ora_df%t_s%s_s%p';
```

In Example 5–2, RMAN automatically replaces the format specifier %t with a four byte time stamp, %s with the backup set number, and %p with the backup piece number.

> **Note:** By configuring an explicit format for disk channels, RMAN does not create backups by default in the **flash recovery area**. In this case, you lose the disk space management capabilities of the flash recovery area.

You can also specify an ASM disk location, as shown in the following example:

```
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '+dgroup1';
```

> **See Also:** "Backing Up Database Files with RMAN" on page 8-7 to learn how to make backups

Configuring Channel Parallelism for Disk and SBT Devices

The number of channels available for a device type when you run a command determines whether RMAN will read or write in parallel. As a rule, the number of channels used in executing a command should match the number of devices accessed. Thus, for tape backups, allocate one channel for each tape drive. For disk backups, allocate one channel for each physical disk, unless you can optimize the backup for your disk subsystem architecture with multiple channels. Failing to allocate the right number of channels adversely affects RMAN performance during I/O operations.

You can configure **channel parallelism** settings, **binary compression** for backup sets, and other options for an **SBT** device with CONFIGURE DEVICE TYPE sbt. You set the configuration for the device type independently of the channel configuration.

Example 5–3 changes the parallelism for the SBT device (sample output included) so that RMAN can back up to a media manager using two tape drives in parallel. Each configured SBT channel will back up roughly half the total data.

Example 5–3 Configuring Parallelism for an SBT Device

```
RMAN> CONFIGURE DEVICE TYPE sbt PARALLELISM 2;

old RMAN configuration parameters:
CONFIGURE DEVICE TYPE 'SBT_TAPE' BACKUP TYPE TO COMPRESSED BACKUPSET PARALLELISM 1;
new RMAN configuration parameters:
CONFIGURE DEVICE TYPE 'SBT_TAPE' PARALLELISM 2 BACKUP TYPE TO COMPRESSED BACKUPSET;
new RMAN configuration parameters are successfully stored
```

Example 5–4 changes the default backup type for the SBT device to an uncompressed backup set (sample output included).

Example 5–4 Configuring the Backup Type for an SBT Device

```
RMAN> CONFIGURE DEVICE TYPE sbt BACKUP TYPE TO BACKUPSET;

old RMAN configuration parameters:
CONFIGURE DEVICE TYPE 'SBT_TAPE' PARALLELISM 2 BACKUP TYPE TO COMPRESSED BACKUPSET;
new RMAN configuration parameters:
CONFIGURE DEVICE TYPE 'SBT_TAPE' BACKUP TYPE TO BACKUPSET PARALLELISM 2;
new RMAN configuration parameters are successfully stored
```

Note that the `CONFIGURE DEVICE TYPE` commands used in this example to set parallelism and backup type do not affect the values of settings not specified. In Example 5–3, the default backup type of compressed backup set was not changed by changing the parallelism. In Example 5–4, the parallelism was not changed by changing the default backup type.

See Also:

- "Specifying Multiple Formats for Disk Backups" on page 8-4 to learn how to parallelize disk backups
- *Oracle Database Backup and Recovery Reference* for reference material on the `CHANNEL` parameter of the `BACKUP` command
- *Oracle Real Application Clusters Administration and Deployment Guide* for information about parallelization in an Oracle Real Application Clusters (Oracle RAC) configuration

Manually Overriding Configured Channels

If you manually allocate a channel during a job, then RMAN disregards any configured channel settings. For example, assume that the default device type is SBT, and you execute the following command:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE DISK;
  BACKUP TABLESPACE users;
}
```

In this case, RMAN uses only the disk channel that you manually allocated within the `RUN` command, overriding any defaults set by using `CONFIGURE DEVICE TYPE`, `CONFIGURE DEFAULT DEVICE`, or `CONFIGURE CHANNEL` settings.

See Also:

- "RMAN Channels" on page 3-3 to learn about configured and allocated channels
- *Oracle Database Backup and Recovery Reference* for `ALLOCATE` syntax
- *Oracle Database Backup and Recovery Reference* for `CONFIGURE` syntax

Configuring Control File and Server Parameter File Autobackups

As explained in "Control File and Server Parameter File Autobackups" on page 7-12, you can configure RMAN to automatically back up the control file and server parameter file. The autobackup occurs whenever a backup record is added. If the database runs in ARCHIVELOG mode, then an autobackup is also taken whenever the database structure metadata in the control file changes. A **control file autobackup** enables RMAN to recover the database even if the current control file, recovery catalog, and server parameter file are lost.

Because the filename for the autobackup follows a well-known format, RMAN can search for it without access to a repository and then restore the server parameter file. After you have started the instance with the restored server parameter file, RMAN can restore the control file from an autobackup. After you mount the control file, the RMAN repository is available and RMAN can restore the datafiles and find the archived redo logs.

You can enable the autobackup feature by running the following command:

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

You can disable the feature by running the following command:

```
CONFIGURE CONTROLFILE AUTOBACKUP OFF;
```

> **See Also:** *Oracle Database Backup and Recovery Reference* for CONFIGURE syntax

Configuring the Control File Autobackup Format

By default, the format of the autobackup file for all configured devices is the substitution variable %F in the FORMAT clause. This variable format translates into *c-IIIIIIIIII-YYYYMMDD-QQ*, with the placeholders defined as follows:

- *IIIIIIIIII* stands for the DBID.
- *YYYYMMDD* is a time stamp of the day the backup is generated.
- *QQ* is the hex sequence that starts with 00 and has a maximum of FF.

You can change the default format by using the following command, where *deviceSpecifier* is any valid device type, and '*string*' must contain the substitution variable %F (and no other substitution variables) and is a valid handle for the specified device:

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT
  FOR DEVICE TYPE deviceSpecifier TO 'string';
```

For example, you can run the following command to specify a nondefault filename for the control file autobackup. In the filename, ? stands for ORACLE_HOME.

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT
  FOR DEVICE TYPE DISK TO '?/oradata/cf_%F';
```

The following example configures the autobackup to write to an Automatic Storage Management disk group:

```
CONFIGURE CONTROLFILE AUTOBACKUP
  FOR DEVICE TYPE DISK TO '+dgroup1/%F';
```

To clear control file autobackup formats for a device, use the following commands:

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK CLEAR;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE sbt CLEAR;
```

If you have set up a **flash recovery area** for the database, then you can direct control file autobackups to the flash recovery area by clearing the control file autobackup format for disk.

Overriding the Configured Control File Autobackup Format

The `SET CONTROLFILE AUTOBACKUP FORMAT` command, which you can specify either within a `RUN` command or at the RMAN prompt, overrides the configured autobackup format in the current session only. The order of precedence is:

1. `SET CONTROLFILE AUTOBACKUP FORMAT` (within a `RUN` block)
2. `SET CONTROLFILE AUTOBACKUP FORMAT` (at `RMAN` prompt)
3. `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT`

The following example shows how the two forms of `SET CONTROLFILE AUTOBACKUP FORMAT` interact:

```
SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO 'controlfile_%F';
BACKUP AS COPY DATABASE;
RUN
{
  SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/tmp/%F.bck';
  BACKUP AS BACKUPSET
    DEVICE TYPE DISK
    DATABASE;
}
```

The first `SET CONTROLFILE AUTOBACKUP FORMAT` controls the name of the control file autobackup until the RMAN client exits, overriding any configured control file autobackup format. The `SET CONTROFILE AUTOBACKUP FORMAT` in the `RUN` block overrides the `SET CONTROLFILE AUTOBACKUP FORMAT` outside the `RUN` block for the duration of the `RUN` block.

Configuring RMAN to Make Backups to a Media Manager

On most platforms, to back up to and restore from sequential media such as tape you must integrate a **media manager** with your Oracle database. You can use **Oracle Secure Backup**, which supports both database and file system backups to tape, as your media manager. You should refer to *Oracle Secure Backup Administrator's Guide* to learn how to set up RMAN for use specifically with Oracle Secure Backup.

If you do not use Oracle Secure Backup, then you can use a third-party media manager. This section describes the generic steps for configuring RMAN for use with a third-party media manager. The actual steps depend on the media management product that you install and the platform on which you are running the database. If you choose to use RMAN with a media manager other than Oracle Secure Backup, then you must obtain all product-specific information from the vendor.

Read the following sections in order when configuring the media manager:

1. Prerequisites for Using a Media Manager with RMAN
2. Determining the Location of the Media Management Library
3. Configuring Media Management Software for RMAN Backups
4. Testing Whether the Media Manager Library Is Integrated Correctly
5. Configuring SBT Channels for Use with a Media Manager

See Also: "Media Management" on page 3-6 for an overview of media management software and its implications for RMAN

Prerequisites for Using a Media Manager with RMAN

Before you can begin using RMAN with a third-party media manager, you must install it and make sure that RMAN can communicate with it. Refer to the media management vendor's software documentation for instructions.

In general, you should begin by installing and configuring the media management software on the target host or production network. Ensure that you can make non-RMAN backups of operating system files on the target database host. This step makes later troubleshooting much easier by confirming that the basic integration of the media manager with the target host has been successful. Refer to your media management documentation to learn how to back up files to the media manager without using RMAN.

Then, obtain and install the third-party media management module for integration with the database server. This module contains the **media management library** that the Oracle database loads and uses when accessing the media manager. It is generally a third-party product which must be purchased separately. Contact your media management vendor for details.

Determining the Location of the Media Management Library

Before attempting to use RMAN with a media manager, determine the location of the media management library. When allocating or configuring a **channel** for RMAN to use to communicate with a media manager, you must specify the `SBT_LIBRARY` parameter in an `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` command. The `SBT_LIBRARY` parameter specifies the path to the library.

The following example shows the channel syntax, where *pathname* is the absolute filename of the library:

```
CONFIGURE CHANNEL DEVICE TYPE sbt
  PARMS 'SBT_LIBRARY=pathname';
```

When RMAN allocates channels to communicate with a media manager, it attempts to load the library indicated by the `SBT_LIBRARY` parameter.

If you do not provide a value for the `SBT_LIBRARY` parameter in an allocated or preconfigured channel, then RMAN looks in a platform-specific default location. On Linux and UNIX, the default library filename is `$ORACLE_HOME/lib/libobk.so`, with the extension name varying according to platform: `.so`, `.sl` on HP-UX, `.a` on AIX, and so on. On Windows the default library location is `%ORACLE_HOME%\bin\orasbt.dll`.

Note: The default media management library file is *not* part of the standard database installation. It is only present if you install third-party media management software.

See Also: Your operating system-specific database documentation and the documentation supplied by your media vendor for instructions on how to achieve media manager integration on your platform

Configuring Media Management Software for RMAN Backups

After installing the media management software, perform whatever configuration that your vendor requires so that the software can accept RMAN backups. Depending on the type of media management software that you installed, you may have to define media pools, configure users and classes, and so on.

Consult your media management vendor documentation for the appropriate RMAN settings. The `PARMS` parameter sends instructions to the media manager. If `PARMS` values are needed for the `ALLOCATE CHANNEL` or the `CONFIGURE CHANNEL` command, or if a `FORMAT` string is recommended for the `BACKUP` or `CONFIGURE` command, then refer to the vendor documentation for this information.

Example 5–5 shows a `PARMS` setting for Oracle Secure Backup. This `PARMS` settings instructs the media manager to back up to a family of tapes called `datafile_mf`. The `PARMS` settings are always vendor-specific.

Example 5–5 PARMS Setting for Oracle Secure Backup

```
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE'
  PARMS 'ENV=(OB_MEDIA_FAMILY=datafile_mf)';
```

See Also:

- *Oracle Database Backup and Recovery Reference* for `ALLOCATE CHANNEL` syntax
- *Oracle Database Backup and Recovery Reference* for channel control options
- *Oracle Secure Backup Reference* for RMAN-specific parameter settings for Oracle Secure Backup

Testing Whether the Media Manager Library Is Integrated Correctly

After you have confirmed that the database server can load the media management library, test to make sure that RMAN can back up to the media manager.

Testing ALLOCATE CHANNEL on the Media Manager

The following steps use the `ALLOCATE CHANNEL` command to perform a basic test that RMAN is able to communicate with the media manager.

To test channel allocation on the media manager:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Run the `ALLOCATE CHANNEL` command with the `PARMS` required by your media management software.

   The following `RUN` command shows sample vendor-specific `PARMS` settings:

   ```
   RUN
   {
     ALLOCATE CHANNEL c1 DEVICE TYPE sbt
       PARMS 'SBT_LIBRARY=/mydir/lib/libobk.so,
       ENV=(OB_DEVICE=drive1,OB_MEDIA_FAMILY=datafile_mf)';
   }
   ```

3. Examine the RMAN output.

If you do not receive an error message, then the database successfully loaded the media management library. If you receive the ORA-27211 error, the media management library could not be loaded:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03009: failure of allocate command on c1 channel at 11/30/2007 13:57:18
ORA-19554: error allocating device, device type: SBT_TAPE, device name:
ORA-27211: Failed to load Media Management Library
Additional information: 25
```

In this case, you must check the media management installation to make sure that the library is correctly installed, and re-check the value for the SBT_LIBRARY parameter as described in "Determining the Location of the Media Management Library" on page 5-9.

If the database is unable to locate a media management library in the location specified by the SBT_LIBRARY parameter or the default location, then RMAN issues an ORA-27211 error and exits.

Whenever channel allocation fails, the database writes a trace file to the trace subdirectory in the **Automatic Diagnostic Repository (ADR)** home directory. The following shows sample output:

```
SKGFQ OSD: Error in function sbtinit on line 2278
SKGFQ OSD: Look for SBT Trace messages in file /oracle/rdbms/log/sbtio.log
SBT Initialize failed for /oracle/lib/libobk.so
```

> **See Also:** *Oracle Database Administrator's Guide* to learn how to use the Automatic Diagnostic Repository to monitor database operations

Testing Backup and Restore Operations on the Media Manager

After testing a channel allocation on the media manager, create and restore a test backup. For example, you could use the command in Example 5–6 (substituting the channel settings required by your media management vendor) to test whether a backup can be created on the media manager. If your database does not use a server parameter file, then back up the current control file instead.

Example 5–6 Backing Up the Server Parameter File to Tape

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
    PARMS 'SBT_LIBRARY=/mydir/lib/libobk.so,
    ENV=(OB_DEVICE=drive1,OB_MEDIA_FAMILY=datafile_mf)';
  BACKUP SPFILE;
  # If your database does not use a server parameter file, use:
  # BACKUP CURRENT CONTROLFILE;
}
```

If the backup succeeds, then attempt to restore the server parameter file as an initialization parameter file. Example 5–7 restores the backup created in Example 5–6 to a temporary directory.

Example 5–7 Restoring the Server Parameter File from Tape

```
RUN
{
```

```
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
    PARMS 'SBT_LIBRARY=/mydir/lib/libobk.so,
    ENV=(OB_DEVICE=drive1,OB_MEDIA_FAMILY=datafile_mf)';
  RESTORE SPFILE TO PFILE '/tmp/test_restore.f';
  # If your database does not use a server parameter file, use:
  # RESTORE CURRENT CONTROLFILE;
}
```

If the backup and restore operations succeed, then you are ready to use the media manager with RMAN. Possible failures include the following cases:

- The backup hangs.

  A hanging backup usually indicates that the media manager is waiting to mount a tape. Check if there are any media manager jobs in tape mount request mode and fix the problem. Ensure that the steps in "Configuring RMAN to Make Backups to a Media Manager" on page 5-8 are correctly done.

- The backup fails with `ORA-27211: Failed to load Media Management Library`.

  This error indicates that the media management software is not correctly configured. Ensure that the steps in "Configuring RMAN to Make Backups to a Media Manager" on page 5-8 are correctly done. Also, ensure that you have the correct `PARMS` and `FORMAT` strings required by your media management software.

  See Also: "Testing the Media Management API" on page 22-11 and Chapter 22, "Troubleshooting RMAN Operations"

Configuring SBT Channels for Use with a Media Manager

This section describes how to configure channels for use with a media manager. For an overview of configured channels and how they are used, refer to the section "Configuring Advanced Channel Options" on page 6-1.

About Media Manager Backup Piece Names

A **backup piece** name is determined by the `FORMAT` string specified in the `BACKUP` command, `CONFIGURE CHANNEL` command, or `ALLOCATE CHANNEL` command. The media manager considers the backup piece name as the name of the backup file, so every backup piece must have a unique name in the **media management catalog**.

You can use the substitution variables in a `FORMAT` parameter to generate unique backup piece names. For example, `%d` specifies the name of the database, whereas `%t` specifies the backup timestamp. For most purposes, you can use `%U`, in which case RMAN automatically generates a unique filename. The backup piece name `12i1nk47_1_1` is an example. If you do not specify the `FORMAT` parameter, then RMAN automatically generates a unique filename with the `%U` substitution variable.

Your media manager may impose restrictions on file names and sizes. In this case, you may need more fine-grained control over the naming of backup pieces so that they obey media manager restrictions. For example, some media managers only support a 14-character backup piece name, and some require special `FORMAT` strings. Note that the unique names generated by the `%U` substitution variable do not exceed 14 characters.

Note that some media managers may have limits on the maximum size of files that they can back up or restore. You must ensure that RMAN does not produce backup sets larger than those limits. To limit backup piece sizes, use the parameter

MAXPIECESIZE, which you can set in the CONFIGURE CHANNEL and ALLOCATE CHANNEL commands.

See Also:

- *Oracle Database Backup and Recovery Reference* and "Number and Size of Backup Pieces" on page 7-5 to learn how to set MAXPIECESIZE
- *Oracle Database Backup and Recovery Reference* for the complete list of variables allowable in format strings with the BACKUP command
- Your media management documentation to determine the string character limit for the media manager

Configuring Automatic SBT Channels

The easiest technique for backing up to a media manager is to configure automatic **SBT** channels. As explained in "Configuring the Default Device for Backups: Disk or SBT" on page 5-3, you can use a tape device as your default backup destination.

To configure channels for use with a media manager:

1. Configure a generic SBT channel.

   In the configuration enter all parameters that you tested in the section "Testing Backup and Restore Operations on the Media Manager" on page 5-11. The following example configures vendor-specific channel parameters and sets the default device:

   ```
   CONFIGURE CHANNEL DEVICE TYPE sbt
     PARMS 'ENV=(OB_RESOURCE_WAIT_TIME=1minute,OB_DEVICE=tape1)';
   ```

2. Configure the default device type to SBT, as shown in the following command:

   ```
   CONFIGURE DEFAULT DEVICE TYPE TO sbt;
   ```

   If you use more than one tape device, then you must specify the channel parallelism as described in "Configuring Channel Parallelism for Disk and SBT Devices" on page 5-5. The following configuration enables you to back up to two tape drives in parallel:

   ```
   CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
   ```

   Optionally, check your channel configuration by running the following command:

   ```
   SHOW CHANNEL FOR DEVICE TYPE sbt;
   ```

3. Make a test backup to tape.

   The following command backs up the server parameter file to tape:

   ```
   BACKUP SPFILE;
   ```

4. List your backups to make sure that the test backup went to the media manager:

   ```
   LIST BACKUP OF SPFILE;
   ```

Configuring the Flash Recovery Area

As explained in "Flash Recovery Area" on page 3-7, the **flash recovery area** feature enables you to set up a disk area where the database can create and manage a variety

of files related to backup and recovery. Use of the flash recovery area is strongly recommended. Consider configuring a flash recovery area as one of the first steps in implementing a backup strategy.

This section outlines the functions of the flash recovery area, identifies the files stored there, explains the rules for file management, and introduces the most important configuration options. This section contains the following topics:

- Overview of the Flash Recovery Area
- Enabling the Flash Recovery Area
- Configuring Locations for Control Files and Redo Logs
- Configuring RMAN File Creation in the Flash Recovery Area

See Also: "Maintaining the Flash Recovery Area" on page 11-6

Overview of the Flash Recovery Area

The flash recovery area can contain control files, online redo logs, archived redo logs, flashback logs, and RMAN backups. Files in the recovery area are **permanent** or **transient**. Permanent files are active files used by the database instance. All files that are not permanent are transient. In general, Oracle Database eventually deletes transient files after they become obsolete under the **backup retention policy** or have been backed up to tape.

Table 5–2 describes the files in the recovery area, the classification of each file as permanent or temporary, and how database availability is affected.

Table 5–2 Files in the Flash Recovery Area

Files	Type	Database Behavior When Flash Recovery Area Is Inaccessible
Multiplexed copies of the current control file	Permanent	The instance fails if the database cannot write to a multiplexed copy of the control file stored in the flash recovery area. Failure occurs even if accessible multiplexed copies are located outside the recovery area. **See Also:** "Configuring Control File Locations" on page 5-19 to learn how to configure control files in the recovery area
Online redo log files	Permanent	Instance availability is not affected if a mirrored copy of the online redo log exists in an accessible location outside the flash recovery area. Otherwise, the instance fails. **See Also:** "Configuring Online Redo Log Locations" on page 5-19 to learn how to configure online redo logs in the recovery area
Archived redo log files	Transient	Instance availability is not affected if the log is archived to an accessible location outside the flash recovery area. Otherwise, the database eventually halts because it cannot archive the online redo logs. **See Also:** "Configuring Archived Redo Log Locations" on page 5-20 to learn how to configure archived redo logs in the recovery area
Foreign archived redo log files	Transient	Instance availability is not affected. **Note:** Foreign archived redo logs are received by a logical standby database for a LogMiner session. Unlike a normal archived log, a **foreign archived redo log** has a different DBID. For this reason, it cannot be backed up or restored on a logical standby database.

Table 5–2 (Cont.) Files in the Flash Recovery Area

Files	Type	Database Behavior When Flash Recovery Area Is Inaccessible
Image copies of datafiles and control files	Transient	Instance availability is not affected.
Backup pieces	Transient	Instance availability is not affected.
Flashback logs	Transient	Instance availability is not affected if guaranteed restore points are not defined. In this case, the database automatically disables Flashback Database, writes a message to the alert log, and continues with database processing. If guaranteed restore points are configured, the instance fails because of interdependencies on the flashback logs. The **Oracle Flashback Database** feature, which provides an convenient alternative to **database point-in-time recovery (DBPITR)**, generates **flashback logs**. These logs are transient files and must be stored in the flash recovery area. Unlike other transient files, flashback logs cannot be backed up to other media. If the flash recovery area has insufficient space to store flashback logs and meet other backup retention requirements, then the recovery area may delete flashback logs. **See Also:** "Enabling Flashback Database" on page 5-33 to learn how to enable flashback logging

Note that if you are on a Windows platform, then you can use the **Volume Shadow Copy Service (VSS)** in conjunction with the **Oracle VSS writer**. In this case, the flash recovery area automates management of files that are already backed up in a VSS snapshot and deletes them as needed.

See Also:

- Chapter 16, "Performing Flashback and Database Point-in-Time Recovery"
- *Oracle Database Platform Guide for Microsoft Windows* to learn about making backups in a VSS environment

Oracle Managed Files and Automatic Storage Management

The flash recovery area can be used in conjunction with **Oracle Managed Files (OMF)** and **Automatic Storage Management (ASM)**. The flash recovery area is built on top of OMF, so the flash recovery area can be stored anywhere Oracle-managed files can. You can also use the recovery area with ASM.

Even if you choose not to set up the flash recovery area in ASM storage, you can still use Oracle Managed Files to manage your backup files in an ASM disk group. You will lose one of the major benefits of the flash recovery area: the automatic deletion of files no longer needed to meet your recoverability goals as space is needed for more recent backups. Nevertheless, the other automatic features of OMF will still function.

When storing backups, using OMF on top of ASM without using a flash recovery area is supported but discouraged. It is awkward to directly manipulate files under ASM.

How Oracle Manages Disk Space in the Flash Recovery Area

Space in the flash recovery area is balanced among backups and archived logs that must be kept according to the retention policy, and other files which may be subject to deletion. Oracle Database does not delete eligible files from the flash recovery area until the space must be reclaimed for some other purpose. Thus, files recently moved to tape are often still available on disk for use in recovery. The recovery area can thus

serve as a cache for tape. When the flash recovery area is full, Oracle Database automatically deletes eligible files to reclaim space in the recovery area as needed.

> **See Also:** "Deletion Rules for the Flash Recovery Area" on page 11-6 and "Responding to a Full Flash Recovery Area" on page 11-8

Enabling the Flash Recovery Area

To enable the recovery area, you must set the initialization parameters in Table 5–3 that are listed as required. Note that in an Oracle Real Application Clusters (Oracle RAC) database, all instances must have the same values for these initialization parameters. The location must be on a cluster file system, ASM, or a shared directory.

Table 5–3 Initialization Parameters for the Flash Recovery Area

Initialization Parameter	Required	Description
DB_RECOVERY_FILE_DEST_SIZE	Yes	Specifies the **disk quota**, which is maximum storage in bytes of data to be used by the recovery area for this database. You must set this parameter *before* DB_RECOVERY_FILE_DEST. The DB_RECOVERY_FILE_DEST_SIZE setting does not include the following kinds of disk overhead: ■ Block 0 or the operating system block header of each Oracle file is not included. Allow an extra 10% for this data when computing the actual disk usage required for the flash recovery area. ■ DB_RECOVERY_FILE_DEST_SIZE does not indicate the real size occupied on disk when the underlying file system is mirrored, compressed, or affected by overhead not known to Oracle Database. For example, if the recovery area is on a two-way mirrored ASM disk group, each file of *x* bytes occupies 2*x* bytes on the ASM disk group. In this case, set DB_RECOVERY_FILE_DEST_SIZE to no more than half the size of the disks for the ASM disk group. Likewise, when using a three-way mirrored ASM disk group, DB_RECOVERY_FILE_DEST_SIZE must be no more than one third the size of the disks in the disk group, and so on.
DB_RECOVERY_FILE_DEST	Yes	Specifies the recovery area location, which can be a file system directory or ASM disk group, but not raw disk. The location must be large enough for the disk quota.
DB_FLASHBACK_RETENTION_TARGET	No	Specifies the upper limit (in minutes) on how far back in time the database may be flashed back. This parameter is required only for Flashback Database. This parameter indirectly determines how much flashback log data is kept in the recovery area. The size of flashback logs generated by the database can vary considerably depending on the database workload. If more blocks are affected by database updates during a given interval, then more disk space is used by the flashback log data generated for that interval.

> **See Also:** *Oracle Database SQL Language Reference* for ALTER SYSTEM syntax, and *Oracle Database Administrator's Guide* for details on setting and changing database initialization parameters

Considerations When Setting the Size of the Flash Recovery Area

The larger the flash recovery area is, the more useful it becomes. Ideally, the flash recovery area should be large enough to contain the files listed in Table 5–2 on page 5-14. The recovery area should be able to contain a copy of all datafiles in the database and the incremental backups used by your chosen backup strategy.

If providing this much space is impractical, then it is best to create an area large enough to keep a backup of the most important tablespaces and all the archived logs not yet on tape. At an absolute minimum, the flash recovery area must be large enough to contain the archived redo logs not yet on tape. If the recovery area has insufficient space to store flashback logs and meet other backup retention requirements, then the recovery area may delete flashback logs to make room.

Formulas for estimating a useful flash recovery area size depend on whether:

- Your database has a small or large number of data blocks that change frequently
- You store backups only on disk, or on disk and tape
- You use a redundancy-based **backup retention policy**, or a recovery window-based retention policy
- You plan to use Flashback Database or a **guaranteed restore point** as alternatives to **point-in-time recovery** in response to logical errors

If you plan to enable flashback logging, then note that the volume of flashback log generation is approximately the same order of magnitude as redo log generation. For example, if you intend to set `DB_FLASHBACK_RETENTION_TARGET` to 24 hours, and if the database generates 20 GB of redo in a day, then a rule of thumb is to allow 20 GB to 30 GB disk space for the flashback logs. The same rule applies to guaranteed restore points when flashback logging is enabled. For example, if the database generates 20 GB redo every day, and if the guaranteed restore point will be kept for a day, then plan to allocate 20 to 30 GB.

Suppose that you want to determine the size of a flash recovery when the backup retention policy is set to `REDUNDANCY 1` and you intend to follow the Oracle Suggested Strategy of using an **incrementally updated backup**. In this example, you use the following formula to estimate the disk quota, where *n* is the interval in days between incremental updates and *y* is the delay in applying the foreign archived redo logs on a logical standby database:

```
Disk Quota =
Size of a copy of database +
Size of an incremental backup +
Size of (n+1) days of archived redo logs +
Size of (y+1) days of foreign archived redo logs (for logical standby) +
Size of control file +
Size of an online redo log member * number of log groups +
Size of flashback logs (based on DB_FLASHBACK_RETENTION_TARGET value)
```

Considerations When Setting the Location of the Flash Recovery Area

Place the flash recovery area on a separate disk from the **database area**, where the database maintains active database files such as datafiles, control files, and online redo logs. Keeping the flash recovery area on the same disk as the database area exposes you to loss of both your live database files and backups if a media failure occurs.

Oracle recommends that `DB_RECOVERY_FILE_DEST` be set to a different value from `DB_CREATE_FILE_DEST` or any of the `DB_CREATE_ONLINE_LOG_DEST_`*n* initialization parameters. The database writes a warning to the alert log if `DB_RECOVERY_FILE_DEST` is the same as these parameters.

Multiple databases can have the same value for DB_RECOVERY_FILE_DEST, but one of the following must be true:

- No two databases for which the DB_UNIQUE_NAME initialization parameters are specified have the same value for DB_UNIQUE_NAME.
- For those databases where no DB_UNIQUE_NAME is provided, no two databases have the same value for DB_NAME.

When databases share a single recovery area in this way, the location should be large enough to hold the files for all databases. Add the values for DB_RECOVERY_FILE_DEST_SIZE for the databases, then allow for overhead such as mirroring or compression.

Setting the Flash Recovery Area Location and Initial Size

Table 5–3 lists the initialization parameters that you must set to enable the flash recovery area. This section explains how to specify a location for the recovery area and set its initial size.

To determine the optimum size for the flash recovery area:

1. If you plan to use flashback logging or guaranteed restore points, then query V$ARCHIVED_LOG to determine how much redo the database generates in the time to which you intend to set DB_FLASHBACK_RETENTION_TARGET.
2. Set the recovery area size.

   If you plan to use flashback logging or guaranteed restore points, then ensure that the size value obtained from step 1 is incorporated into the setting. Set the DB_RECOVERY_FILE_DEST_SIZE initialization parameter by any of the following means:

   - Shut down the database and set the DB_RECOVERY_FILE_DEST_SIZE parameter in the initialization parameter file of the database, as shown in the following example:

     ```
     DB_RECOVERY_FILE_DEST_SIZE = 10G
     ```

   - Specify them with the SQL statement ALTER SYSTEM SET when the database is open, as shown in the following examples:

     ```
     ALTER SYSTEM SET
       DB_RECOVERY_FILE_DEST_SIZE = 10G
       SCOPE=BOTH SID='*';
     ```

   - Use the Database Configuration Assistant to set the size
3. Set the recovery area location.

   Set the initialization parameters by any of the following means:

   - Set DB_RECOVERY_FILE_DEST in the initialization parameter file of the database, as shown in the following example:

     ```
     DB_RECOVERY_FILE_DEST = '/u01/oradata/rcv_area'
     ```

   - Specify DB_RECOVERY_FILE_DEST with the SQL statement ALTER SYSTEM SET when the database is open, as shown in the following example:

     ```
     ALTER SYSTEM SET
       DB_RECOVERY_FILE_DEST = '+disk1'
       SCOPE=BOTH SID='*';
     ```

- Use the Database Configuration Assistant to set the location

If you do *not* plan to use flashback logging, then open the database (if it is closed) and do not complete the rest of the steps in this procedure.

4. If flashback logging is enabled, then run the database under a normal workload for the time period specified by DB_FLASHBACK_RETENTION_TARGET.

   In this way, the database can generate a representative sample of flashback logs.

5. Query the V$FLASHBACK_DATABASE_LOG view as follows:

```
SELECT ESTIMATED_FLASHBACK_SIZE
FROM   V$FLASHBACK_DATABASE_LOG;
```

   The result is an estimate of the disk space needed to meet the current flashback retention target, based on the database workload since Flashback Database was enabled.

6. If necessary, adjust the flashback log space requirement based on the actual size of flashback logs generated during the time period specified by DB_FLASHBACK_RETENTION_TARGET.

See also: "Managing Space For Flashback Logs in the Flash Recovery Area" on page 11-7

Configuring Locations for Control Files and Redo Logs

As explained in "Overview of the Flash Recovery Area" on page 5-14, the only permanent files are multiplexed copies of the current control file and online redo logs. This section explains how to set locations for these files as well as the archived logs.

Configuring Online Redo Log Locations

The initialization parameters that determine where online redo log files are created are DB_CREATE_ONLINE_LOG_DEST_*n*, DB_RECOVERY_FILE_DEST, and DB_CREATE_FILE_DEST. Details of the effect of combinations of these parameters on online redo log creation can be found in *Oracle Database SQL Language Reference* in the description of the LOGFILE clause of the CREATE DATABASE statement.

The following SQL statements can create online redo logs in the flash recovery area:

- CREATE DATABASE
- ALTER DATABASE ADD LOGFILE
- ALTER DATABASE ADD STANDBY LOGFILE
- ALTER DATABASE OPEN RESETLOGS

The default size of an online log created in the flash recovery area is 100 MB. The log member filenames are automatically generated by the database.

Configuring Control File Locations

The initialization parameters CONTROL_FILES, DB_CREATE_ONLINE_LOG_DEST_*n*, DB_RECOVERY_FILE_DEST, and DB_CREATE_FILE_DEST all interact to determine the location where the database control files are created. For a full description of how these parameters interact, see the "Semantics" section of the description of CREATE CONTROLFILE in *Oracle Database SQL Language Reference.*

If the database creates an Oracle managed control file, and if the database uses a server parameter file, then the database sets the CONTROL_FILES initialization parameter in

the server parameter file. If the database uses a client-side initialization parameter file, then you must set the CONTROL_FILES initialization parameter manually in the initialization parameter file.

Configuring Archived Redo Log Locations

It is recommended that you the use flash recovery area as an archiving location because the archived logs are automatically managed by the database. The generated filenames for the archived logs in the flash recovery area are for Oracle-managed files and are not determined by LOG_ARCHIVE_FORMAT. Whatever archiving scheme you choose, it is always advisable to create multiple copies of archived redo logs.

You have the following basic options for archiving redo logs, listed from most to least recommended:

1. Enable archiving to the flash recovery area *only* and use disk mirroring to create the redundancy needed to protect the archived redo logs.

   If DB_RECOVERY_FILE_DEST is specified and no LOG_ARCHIVE_DEST_*n* is specified, then LOG_ARCHIVE_DEST_10 is implicitly set to the recovery area. You can override this behavior by setting LOG_ARCHIVE_DEST_10 to an empty string.

2. Enable archiving to the flash recovery area and set other LOG_ARCHIVE_DEST_*n* initialization parameter to locations outside the flash recovery area.

   If a flash recovery area is configured, then you can add the flash recovery area as an archiving destination by setting any LOG_ARCHIVE_DEST_*n* parameter to LOCATION=USE_DB_RECOVERY_FILE_DEST.

3. Set LOG_ARCHIVE_DEST_*n* initialization parameters to archive *only* to non-flash recovery area locations.

If you use the flash recovery area, then you cannot use the LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST initialization parameters. If you do, then will not be able to start the instance. Instead, set the LOG_ARCHIVE_DEST_*n* parameters. After your database is using LOG_ARCHIVE_DEST_*n*, you can configure a recovery area.

Note also that if you enable archiving but do not set any value for LOG_ARCHIVE_DEST, LOG_ARCHIVE_DEST_*n*, or DB_RECOVERY_FILE_DEST, then the redo logs are archived to a default location that is platform-specific. For example, on Solaris the default is ?/dbs.

> **See Also:** *Oracle Database Reference* for details on the semantics of the LOG_ARCHIVE_DEST_*n* parameters

Configuring RMAN File Creation in the Flash Recovery Area

This section describes RMAN commands or implicit actions (such as control file autobackups) that can create files in the flash recovery area, and how to control whether a command creates files there or in another destination. The commands are:

- BACKUP

  If you do not specify the FORMAT clause for disk backups, then RMAN creates backup pieces and image copies in the flash recovery area, with names in Oracle Managed Files name format. If a flash recovery area is enabled, and if you *do* specify FORMAT on BACKUP or a channel, then RMAN creates the backup in a platform-specific location rather than in the recovery area.

- Control File Autobackup

RMAN can create control file autobackups in the flash recovery area. Use the RMAN command `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK CLEAR` to clear any configured format option for the control file autobackup location on disk. RMAN creates control file autobackups in in the flash recovery area when no other destination is configured.

- `RESTORE ARCHIVELOG`

  Explicitly or implicitly set one of the `LOG_ARCHIVE_DEST_`*n* parameters to `LOCATION=USE_DB_RECOVERY_FILE_DEST`. If you do not specify `SET ARCHIVELOG DESTINATION` to override this behavior, then RMAN restores archived redo log files to the flash recovery area.

- `RECOVER DATABASE` or `RECOVER TABLESPACE`, `RECOVER ... BLOCK`, and `FLASHBACK DATABASE`

  These commands restore archived redo log files from backup for use during media recovery, as required by the command. RMAN restores any redo log files needed during these operations to the flash recovery area and deletes them after they are applied during media recovery.

  To direct the restored archived logs to the flash recovery area, set one of the `LOG_ARCHIVE_DEST_`*n* parameters to `LOCATION = USE_DB_RECOVERY_FILE_DEST`. Make sure you are not using `SET ARCHIVELOG DESTINATION` to direct restored logs to some other destination.

Configuring the Backup Retention Policy

As explained in "Backup Retention Policies" on page 7-17, the **backup retention policy** specifies which backups must be retained to meet your data recovery requirements. This policy can be based on a **recovery window** or **redundancy**. Use the `CONFIGURE RETENTION POLICY` command to specify the retention policy.

> **See Also:** *Oracle Database Backup and Recovery Reference* for `CONFIGURE` syntax

Configuring a Redundancy-Based Retention Policy

The `REDUNDANCY` parameter of the `CONFIGURE RETENTION POLICY` command specifies how many full or level 0 backups of each datafile and control file that RMAN should keep. In other words, if the number of full or level 0 backups for a specific datafile or control file exceeds the `REDUNDANCY` setting, then RMAN considers the extra backups as obsolete. The default retention policy is `REDUNDANCY 1`.

As you produce more backups, RMAN keeps track of which ones to retain and which are obsolete. RMAN retains all archived logs and incremental backups that are needed to recover the nonobsolete backups.

Assume that you make a full backup of datafile 7 on Monday, Tuesday, Wednesday, and Thursday. You now have four full backups of this datafile. If `REDUNDANCY` is `2`, then the Monday and Tuesday backups are obsolete. If you make another backup on Friday, then the Wednesday backup of datafile 7 becomes obsolete.

Assume a different case in which `REDUNDANCY` is `1`. You run a level 0 database backup at noon Monday, a level 1 cumulative backup at noon on Tuesday and Wednesday, and a level 0 backup at noon on Thursday. Immediately after each daily backup you run a `DELETE OBSOLETE`. The Wednesday `DELETE` command does not remove the Tuesday level 1 backup because this backup is not redundant: the Tuesday level 1 backup could be used to recover the Monday level 0 backup to a time between noon on Tuesday and

noon on Wednesday. However, the DELETE command on Thursday removes the previous level 0 and level 1 backups.

Run the CONFIGURE RETENTION POLICY command at the RMAN prompt, as in the following example:

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 3;
```

> **See Also:** "Deleting Obsolete RMAN Backups Based on Retention Policies" on page 11-23

Configuring a Recovery Window-Based Retention Policy

The RECOVERY WINDOW parameter of the CONFIGURE command specifies the number of days between the current time and the earliest point of recoverability. RMAN does not consider any full or **level 0 incremental backup** as obsolete if it falls within the recovery window. Additionally, RMAN retains all archived logs and level 1 incremental backups that are needed to recover to a random point within the window.

Run the CONFIGURE RETENTION POLICY command at the RMAN prompt. This example ensures that you can recover the database to any point within the last week:

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
```

RMAN does not automatically delete backups rendered obsolete by the recovery window. Instead, RMAN shows them as OBSOLETE in the REPORT OBSOLETE output and in the OBSOLETE column of V$BACKUP_FILES. RMAN deletes obsolete files if you run the DELETE OBSOLETE command.

> **See Also:** "Deleting Obsolete RMAN Backups Based on Retention Policies" on page 11-23

Disabling the Retention Policy

When you disable the retention policy, RMAN does not consider any backup as obsolete. To disable the retention policy, run this command:

```
CONFIGURE RETENTION POLICY TO NONE;
```

Configuring the retention policy to NONE is not the same as clearing it. Clearing it returns it to its default setting of REDUNDANCY 1, whereas NONE disables it.

If you disable the retention policy and run REPORT OBSOLETE or DELETE OBSOLETE without passing a retention policy option to the command, then RMAN issues an error because no retention policy exists to determine which backups are obsolete.

Note: If you are using a flash recovery area, then you should not run your database with the retention policy disabled. If files are never considered obsolete, then a file can only be deleted from the flash recovery area if it has been backed up to some other disk location or to a tertiary storage device such as tape. It is likely that all of the space in your recovery area will be used, which interferes with the normal operation of your database. See "How Oracle Manages Disk Space in the Flash Recovery Area" on page 5-15.

Configuring Backup Optimization

Run the CONFIGURE command to enable and disable backup optimization. Backup optimization skips the backup of files in certain circumstances if the identical file or an identical version of the file has already been backed up.

Overview of Backup Optimization

If you enable **backup optimization**, then the BACKUP command skips backing up files when the identical file has already been backed up to the specified device type. Table 5–4 describes criteria that RMAN uses to determine whether a file is identical to a file that it already backed up.

Table 5–4 Criteria to Determine an Identical File

Type of File	Criteria to Determine an Identical File
Datafile	The datafile must have the same **DBID**, **checkpoint** SCN, creation SCN, and RESETLOGS SCN and time as a datafile already in a backup. The datafile must be offline-normal, read-only, or closed normally.
Archived log	Same DBID, thread, sequence number, and RESETLOGS SCN and time.
Backup set	Same DBID, backup set record ID and stamp.

If RMAN determines that a file is identical and it has already been backed up, then it is a candidate to be skipped. RMAN must do further checking to determine whether to skip the file, however, because both the retention policy and the backup duplexing feature are factors in the algorithm that determines whether RMAN has sufficient backups on the specified device type.

RMAN uses backup optimization when the following conditions are true:

- The CONFIGURE BACKUP OPTIMIZATION ON command has been run to enable backup optimization.
- You run BACKUP DATABASE, BACKUP ARCHIVELOG with ALL or LIKE options, or BACKUP BACKUPSET ALL, BACKUP RECOVERY AREA, BACKUP RECOVERY FILES, or BACKUP DATAFILECOPY.
- Only one type of channel is allocated, that is, you do not mix disk and **SBT** channels in the same backup command.

> **Note:** In **backup undo optimization**, RMAN excludes undo that is not needed for recovery of a backup, that is, for transactions that have already been committed. You can enable and disable backup optimization, but backup undo optimization is built-in behavior.

For example, assume that you have configured backup optimization. These commands back up to tape the database, all archived logs, and all backup sets:

```
BACKUP DEVICE TYPE sbt DATABASE PLUS ARCHIVELOG;
BACKUP DEVICE TYPE sbt BACKUPSET ALL;
```

If none of the backed-up files has changed since the last backup, then RMAN does not back up the files again. RMAN also does not signal an error if it skips all files specified in the command because the files have already been backed up.

You can override optimization at any time by specifying the FORCE option on the BACKUP command. For example, you can run:

```
BACKUP DATABASE FORCE;
BACKUP ARCHIVELOG ALL FORCE;
```

> **See Also:** The `CONFIGURE` entry in *Oracle Database Backup and Recovery Reference* for a complete description of the backup optimization rules

Effect of Retention Policies on Backup Optimization for SBT Backups

Backup optimization is not always applied when backing up to **SBT** devices. The exceptions to normal backup optimization behavior for recovery window-based and redundancy-based retention policies are described in the following sections.

> **Note:** Use caution when enabling backup optimization if you use a media manager with its own internal expiration policy. Run the `CROSSCHECK` command periodically to synchronize the RMAN repository with the media manager. Otherwise, RMAN may skip backups due to optimization without recognizing that the media manager has discarded backups stored on tape.

Backup Optimization for SBT Backups with Recovery Window Retention Policy

Suppose that backup optimization is enabled, and a recovery window **backup retention policy** is in effect. In this case, when performing SBT backups RMAN always backs up datafiles whose most recent backup is older than the **recovery window**. For example, assume the following scenario:

- Today is February 21.
- The recovery window is 7 days.
- The most recent backup of tablespace `tools` to tape is January 3.
- Tablespace `tools` is read-only.

On February 21, when you issue a command to back up tablespace `tools` to tape, RMAN backs it up even though it did not change after the January 3 backup (because it is read-only). RMAN makes the backup because no backup of the tablespace exists within the 7-day recovery window.

This behavior enables the media manager to expire old tapes. Otherwise, the media manager would be forced to keep the January 3 backup of tablespace `tools` indefinitely. By making a more recent backup of tablespace `tools` on February 21, RMAN enables the media manager to expire the tape containing the January 3 backup.

Backup Optimization for SBT Backups With Redundancy Retention Policy

Assume that you configure a retention policy for redundancy. In this case, RMAN only skips backups of offline or read-only datafiles to SBT when there are *r* + 1 backups of the files, where *r* is set in `CONFIGURE RETENTION POLICY TO REDUNDANCY` *r*.

For example, assume that you enable backup optimization and set the following retention policy:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
```

With these settings, RMAN only skips backups when three identical files are already backed up. Also assume that you have never backed up the `users` tablespace, which

is read/write, and that you perform the actions described in Table 5–5 over the course of the week.

Table 5–5 Effect of Redundancy Setting on Backup Optimization

Day	Action	Result	Redundant Backup
Monday	Take `users` offline normal.		
Tuesday	`BACKUP DATABASE`	The `users` tablespace is backed up.	
Wednesday	`BACKUP DATABASE`	The `users` tablespace is backed up.	
Thursday	`BACKUP DATABASE`	The `users` tablespace is backed up.	Tuesday backup
Friday	`BACKUP DATABASE`	The `users` tablespace is *not* backed up.	Tuesday backup
Saturday	`BACKUP DATABASE`	The `users` tablespace is *not* backed up.	Tuesday backup
Sunday	`DELETE OBSOLETE`	The Tuesday backup is deleted.	
Monday	`BACKUP DATABASE`	The `users` tablespace is backed up.	Wednesday backup

The backups on Tuesday, Wednesday, and Thursday back up the offline `users` tablespace to satisfy the condition that three backups must exist (one more than redundancy setting). The Friday and Saturday backups do not back up the `users` tablespace because of backup optimization. Note that the Tuesday backup of `users` is obsolete beginning on Thursday.

On Sunday, you delete all obsolete backups, which removes the Tuesday backup of `users`. The Tuesday backup is obsolete because of the retention policy setting. The whole database backup on Monday then backs up the `users` tablespace to satisfy the condition that three backups must exist (one more than redundancy setting). In this way, you can recycle your tapes over time.

See Also: "Configuring Backup Optimization" on page 5-23

Configuring Backup Optimization

By default, backup optimization is configured to `OFF`. You can use the `SHOW BACKUP OPTIMIZATION` command to view the current settings of backup optimization.

To configure backup optimization:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Run the `SHOW BACKUP OPTIMIZATION` command to determine whether optimization is currently enabled.

   For example, enter the following command:

   ```
   SHOW BACKUP OPTIMIZATION;
   ```

   Sample output for `SHOW BACKUP OPTIMIZATION` follows:

   ```
   RMAN configuration parameters for database with db_unique_name PROD1 are:
   CONFIGURE BACKUP OPTIMIZATION ON;
   ```

3. Enable backup optimization by running the following command:

   ```
   CONFIGURE BACKUP OPTIMIZATION ON;
   ```

See Also: "Using Backup Optimization to Skip Files" on page 9-3 for examples of how to optimize RMAN backups

Configuring an Archived Redo Log Deletion Policy

You can use RMAN to create a persistent configuration that governs when archived redo logs are eligible for deletion from disk.

About Archived Redo Log Deletion Policies

You can use the `CONFIGURE ARCHIVELOG DELETION POLICY` command to specify when archived redo logs are eligible for deletion. This deletion policy applies to all archiving destinations, including the **flash recovery area**.

Archived redo logs can be deleted automatically by the database or as a result of user-initiated RMAN commands. Only logs in the flash recovery area can be deleted automatically by the database. For archived redo log files in the flash recovery area, the database retains them as long as possible and automatically deletes eligible logs when additional disk space is required. You can manually delete eligible logs from any location, whether inside or outside the flash recovery area, when you issue `BACKUP ... DELETE INPUT` or `DELETE ARCHIVELOG`.

When the Archived Redo Log Deletion Policy Is Disabled

The **archived redo log deletion policy** is configured to `NONE` by default. In this case, RMAN considers archived redo log files in the recovery area as eligible for deletion if they meet *both* of the following conditions:

- The archived redo logs, whether in the flash recovery area or outside of it, have been transferred to the required remote destinations specified by `LOG_ARCHIVE_DEST_`*n*.
- The archived redo logs have been backed up at least once to disk or SBT *or* the logs are obsolete according to the **backup retention policy**.

  The backup retention policy considers logs obsolete *only if* the logs are not needed by a **guaranteed restore point** *and* the logs are not needed by **Oracle Flashback Database**. Archived redo logs are needed by Flashback Database if the logs were created later than `SYSDATE-'DB_FLASHBACK_RETENTION_TARGET'`.

  See Also:

  - The `CONFIGURE ARCHIVELOG DELETION POLICY` entry in *Oracle Database Backup and Recovery Reference* for detailed information about policy options
  - *Oracle Data Guard Concepts and Administration* to learn how to configure an archived log deletion policy in a Data Guard environment

When the Archived Redo Log Deletion Policy Is Enabled

You can use the `CONFIGURE ARCHIVELOG DELETION POLICY BACKED UP` *`integer`* `TIMES TO DEVICE TYPE` command to enable an archived log deletion policy. This configuration specifies that archived logs are eligible for deletion only when the specified number of archived log backups exist on the specified device type.

If the deletion policy is configured with the `BACKED UP` *`integer`* `TIMES` clause, then a `BACKUP ARCHIVELOG` command copies the logs unless *`integer`* backups already exist on the specified device type. If *`integer`* backups of the logs exist, then the `BACKUP ARCHIVELOG` command skips the logs. In this way, the archived log deletion policy functions as a default `NOT BACKED UP` *`integer`* `TIMES` clause on the `BACKUP`

ARCHIVELOG command. Note that you can override the deletion policy by specifying the FORCE option on the BACKUP command.

The archived log deletion policy also has options specific to a Data Guard environment. For example, if you specify the APPLIED ON STANDBY clause, then RMAN can delete logs after they have been applied at all mandatory remote destinations. If you specify SHIPPED TO STANDBY, for example, then RMAN can delete logs when they have been transferred to all mandatory standby destinations.

See Also:

- The CONFIGURE ARCHIVELOG DELETION POLICY entry in *Oracle Database Backup and Recovery Reference* for detailed information about policy options
- *Oracle Data Guard Concepts and Administration* to learn how to configure an archived log deletion policy in a Data Guard environment

Enabling an Archived Redo Log Deletion Policy

This section explains how to configure an archived redo log deletion policy. By default the policy is set to NONE.

To enable an archived redo log deletion policy:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Run the CONFIGURE ARCHIVELOG DELETION POLICY command with the desired options.

   The following example specifies that archived redo logs are eligible to be deleted from the flash recovery area and all local archiving destinations when logs have been backed up at least twice tape:

```
CONFIGURE ARCHIVELOG DELETION POLICY
  TO BACKED UP 2 TIMES TO SBT;
```

See Also:

- "Deleting Archived Redo Logs After Backups" on page 8-13
- *Oracle Data Guard Concepts and Administration* to learn how to manage archived redo logs in a Data Guard environment
- *Oracle Database Backup and Recovery Reference* for a complete explanation of the CONFIGURE ARCHIVELOG DELETION POLICY command and the conditions under which archived logs are eligible for deletion

Configuring Oracle Flashback Database and Restore Points

This section describes how to plan and configure the environment for Oracle Flashback Database. This section also explains how to create restore points. This section contains the following topics:

- About Restore Points and Flashback Database
- Prerequisites for Flashback Database and Guaranteed Restore Points
- Enabling Flashback Database
- Setting the Flash Recovery Area Location and Initial Size

- Creating Normal and Guaranteed Restore Points
- Configuring the Environment for Optimal Flashback Database Performance

About Restore Points and Flashback Database

Oracle Flashback Database and restore points are related data protection features. These features provide more efficient alternatives to point-in-time recovery for reversing unwanted database changes. Flashback Database enables you to rewind an entire database backward in time, reversing the effects of database changes within a time window. The effects are similar to **database point-in-time recovery (DBPITR)**.

Restore points provide capabilities related to Flashback Database and other media recovery operations. In particular, a guaranteed restore point created at an SCN ensures that you can use Flashback Database to rewind the database to this SCN. You can use restore points and Flashback Database independently or together.

Flashback Database

Oracle Flashback Database is accessible through the RMAN command `FLASHBACK DATABASE` or through the SQL statement `FLASHBACK DATABASE`. You can use either command to quickly recover the database from logical data corruptions or user errors.

Flashback Database is similar to conventional point-in-time recovery in its effects, enabling you to return a database to its state at a time in the recent past. Flashback Database is much faster than point-in-time recovery, however, because it does not require restoring datafiles from backup and requires applying fewer changes from the archived redo logs.

You can use Flashback Database to reverse most unwanted changes to a database as long as the datafiles are intact. You can even return a database to its state in a previous **incarnation**, that is, undo the effects of an `ALTER DATABASE OPEN RESETLOGS` statement. "Rewinding a Database with Flashback Database" on page 16-11 explains how to use the `FLASHBACK DATABASE` command to reverse database changes.

Flashback Database uses its own logging mechanism, creating **flashback logs** and storing them in the **flash recovery area**. You can only use Flashback Database if flashback logs are available. Thus, to take advantage of this feature you must set up your database in advance to create flashback logs.

To enable Flashback Database, you configure a flash recovery area and set a **flashback retention target**. This retention target specifies how far back you can rewind a database with Flashback Database. From the target time onwards, the database regularly copies images of every changed block in the datafiles to the flashback logs.

When you use Flashback Database to rewind a database to a past target time, the command determines which blocks changed after the target time and restores them from the flashback logs. The database restores the version of each block that is most immediately prior to the target time. The database then uses redo logs to reapply changes that were made after these blocks were written to the flashback logs.

Redo logs on disk or tape must be available for the entire time period spanned by the flashback logs. For example, if the flashback retention target is one week, then you must ensure that online and archived redo logs that contain all changes for the past week are accessible. Note that in practice, redo logs are typically needed much longer than the flashback retention target to support point-in-time recovery.

About the Flashback Database Window The range of SCNs for which there is currently enough flashback log data to support the `FLASHBACK DATABASE` command is called

the **flashback database window**. The flashback database window cannot extend further back than the earliest SCN in the available flashback logs.

> **Note:** Some database operations, such as dropping a tablespace or shrinking a datafile, cannot be reversed with Flashback Database. In these cases the flashback database window begins at the time immediately following that operation.

You cannot back up flashback logs to locations outside the flash recovery area. To increase the likelihood that enough flashback logs are retained to meet the flashback database window, you can increase the space in your flash recovery area (see "Setting the Flash Recovery Area Location and Initial Size" on page 5-18).

If the flash recovery area is not large enough to hold the flashback logs and files such as archived redo logs and other backups needed for the retention policy, then the database may delete flashback logs from the earliest SCNs to make room for other files. Consequently, the flashback database window can be shorter than the flashback retention target, depending on the size of the flash recovery area, other backups that must be retained, and how much flashback logging data is needed. The flashback retention target is a target, not a guarantee that Flashback Database will be available.

If you cannot use `FLASHBACK DATABASE` because the flashback database window is not long enough, then you can use database point-in-time recovery (DBPITR) in most cases to achieve a similar result. Guaranteed restore points are the only way to ensure that you can use Flashback Database to return to a specific point in time or guarantee the size of the flashback window.

See Also:

- "Rewinding a Database with Flashback Database" on page 16-11 to learn about Flashback Database
- "Performing Database Point-in-Time Recovery" on page 16-14 to learn about DBPITR
- "Guaranteed Restore Points" on page 5-30 to learn about guaranteed restore points and Flashback Database

Normal Restore Points

A **normal restore point** assigns a restore point name to an SCN or specific point in time. Thus, a restore point functions as a bookmark or alias for this SCN. Before performing any operation that you may have to reverse, you can create a normal restore point. The control file stores the name of the restore point and the SCN.

If you need to use flashback features or point-in-time recovery, then you can use the name of the restore point instead of a time or SCN. The following commands support this use of restore points:

- The `RECOVER DATABASE` and `FLASHBACK DATABASE` commands in RMAN
- The `FLASHBACK TABLE` statement in SQL

A normal restore point eliminates the need to manually record an SCN in advance or determine the correct SCN after the fact by using features such as Flashback Query.

Normal restore points are lightweight. The control file can maintain a record of thousands of normal restore points with no significant impact on database performance. Normal restore points eventually age out of the control file if not manually deleted, so they require no ongoing maintenance.

> **See Also:** *Oracle Database Advanced Application Developer's Guide* to learn how to use Flashback Query

Guaranteed Restore Points

Like a normal restore point, a **guaranteed restore point** serves as an alias for an SCN in recovery operations. A principal difference is that guaranteed restore points never age out of the control file and must be explicitly dropped. In general, you can use a guaranteed restore point as an alias for an SCN with any command that works with a normal restore point. Except as noted, the information about where and how to use normal restore points applies to guaranteed restore points as well.

A guaranteed restore point ensures that you can use Flashback Database to rewind a database to its state at the restore point SCN, even if the generation of **flashback logs** is disabled. If flashback logging is enabled, then a guaranteed restore point enforces the retention of flashback logs required for Flashback Database to any SCN after the earliest guaranteed restore point. Thus, if flashback logging is enabled, you can rewind the database to any SCN in the continuum rather than to a single SCN only.

> **Caution:** If flashback logging is disabled, then you *cannot* use `FLASHBACK DATABASE` to rewind the database to SCNs between the guaranteed restore points and the current time. In this case, your only option to return the database to an intermediate SCN is DBPITR.

If the recovery area has enough disk space to store the needed logs, then you can use a guaranteed restore point to rewind a whole database to a known good state days or weeks ago. As with Flashback Database, even the effects of `NOLOGGING` operations like direct load inserts can be reversed with guaranteed restore points.

> **Note:** Limitations that apply to Flashback Database also apply to guaranteed restore points. For example, shrinking a datafile or dropping a tablespace can prevent flashing back the affected datafiles to the guaranteed restore point. See the `FLASHBACK DATABASE` entry in *Oracle Database Backup and Recovery Reference* for a complete list of command prerequisites and usage notes.

Guaranteed Restore Points and Storage Snapshots In practice, guaranteed restore points provide a useful alternative to storage snapshots. Storage snapshots are often used to protect a database before risky operations such as large-scale database updates or application patches or upgrades. Rather than creating a snapshot or **duplicate database** to test the operation, you can create a guaranteed restore point on a primary or **physical standby database**. You can then perform the risky operation with the certainty that the required flashback logs are retained.

Logging for Flashback Database and Guaranteed Restore Points

Logging for Flashback Database and guaranteed restore points involves capturing images of datafile blocks before changes are applied. The `FLASHBACK DATABASE` command can use these images to return the datafiles to their previous state.

The chief differences between normal flashback logging and logging for guaranteed restore points are related to when blocks are logged and whether the logs can be deleted in response to space pressure in the flash recovery area. These differences affect space usage for logs and database performance.

Your recoverability goals partially determine whether to enable logging for flashback database, use guaranteed restore points, or both. The implications in performance and in space usage for these features, separately and when used together, should also factor into your decision.

Guaranteed Restore Points and Flash Recovery Area Space Usage The following rules govern creation, retention, overwriting and deletion of flashback logs in the flash recovery area:

- If the flash recovery area has enough space, then a flashback log is created whenever necessary to satisfy the flashback retention target.
- If a flashback log is old enough that it is no longer needed to satisfy the flashback retention target, then a flashback log is reused.
- If the database needs to create a new flashback log and the flash recovery area is full or there is no disk space, then the oldest flashback log is reused instead.

> **Note:** Reusing the oldest flashback log shortens the flashback database window. If enough flashback logs are reused due to a lack of disk space, then the flashback retention target may not be satisfied.

- If the flash recovery area is full, then an archived redo log that is not needed to satisfy the backup retention policy may be automatically deleted by the flash recovery area to make space for other files. In this case, any flashback logs that would require the use of that redo log file for the use of `FLASHBACK DATABASE` are also deleted.
- No file in the flash recovery area is eligible for deletion if it is required to satisfy a guaranteed restore point. Thus, retention of flashback logs and other files required to satisfy the guaranteed restore point, in addition to files required to satisfy the backup retention policy, can cause the flash recovery area to fill completely. Consult "Responding to a Full Flash Recovery Area" if your flash recovery area becomes full.

When you create a guaranteed restore point, with or without enabling full flashback database logging, you must monitor the space available in your flash recovery area. "Managing Space For Flashback Logs in the Flash Recovery Area" on page 11-7 explains how to monitor flash recovery area disk space usage.

> **Caution:** If no files are eligible for deletion from the flash recovery area because of the requirements imposed by your retention policy and the guaranteed restore point, then the database behaves as if it has encountered a disk full condition. In many circumstances, this causes your database to halt. See "Responding to a Full Flash Recovery Area" on page 11-8.

Logging for Guaranteed Restore Points With Flashback Logging Disabled Assume that you create a guaranteed restore point when logging for Flashback Database is disabled. In this case, the first time a datafile block is modified after the time of the guaranteed restore point, the database stores an image of the block before the modification in the flashback logs. Thus, the flashback logs preserve the contents of every changed data block at the time that the guaranteed restore point was created. Later modifications to the same block do not cause the contents to be logged again unless another guaranteed restore point was created after the block was last modified.

This method of logging has the following important consequences:

- FLASHBACK DATABASE can re-create the datafile contents at the time of a guaranteed restore point by means of the block images.
- For workloads that repeatedly modify the same data, disk space usage can be less than normal flashback logging. Less space is needed because each changed block is only logged once. Applications with low volume inserts may benefit from this disk space saving while this advantage is less likely for applications with high volume inserts or large batch inserts. The performance overhead of logging for a guaranteed restore point without flashback database logging enabled can also be lower.

Assume that your primary goal is the ability to return your database to the time at which the guaranteed restore point was created. In this case, it is usually more efficient to turn off flashback logging and use only guaranteed restore points. For example, suppose that you are performing an application upgrade on a database host over a weekend. You could create a guaranteed restore point at the start of the upgrade. If the upgrade fails, then reverse the changes with FLASHBACK DATABASE.

Logging for Flashback Database With Guaranteed Restore Points Defined If you enable Flashback Database and define one or more guaranteed restore points, then the database performs normal flashback logging. In this case, the recovery area retains the flashback logs required to flash back to any arbitrary time between the present and the earliest currently defined guaranteed restore point. Flashback logs are not deleted in response to space pressure if they are required to satisfy the guarantee.

Flashback logging causes some performance overhead. Depending upon the pattern of activity on your database, it can also cause significant space pressure in the flash recovery area. Thus, you should monitor space used in the flash recovery area.

Prerequisites for Flashback Database and Guaranteed Restore Points

The prerequisites for enabling Flashback Database are the following:

- Your database must be running in ARCHIVELOG mode, because archived logs are used in the Flashback Database operation.
- You must have a flash recovery area enabled, because flashback logs can only be stored in the flash recovery area.
- For Real Application Clusters databases, the flash recovery area must be stored in a clustered file system or in ASM.

To support the use of guaranteed restore points, the database must satisfy the following prerequisites:

- The COMPATIBLE initialization parameter must be set to 10.2 or greater.
- The database must be running in ARCHIVELOG mode.

  To rewind your database to a guaranteed restore point, the FLASHBACK DATABASE command requires the use of archived redo logs starting from around the time of the restore point.
- A flash recovery area must be configured, as described in "Configuring the Flash Recovery Area" on page 5-13.

  Guaranteed restore points use a mechanism similar to flashback logging. As with flashback logging, Oracle Database must store the required logs in the flash recovery area.

- If Flashback Database is not enabled, then the database must be mounted, not open, when creating the first guaranteed restore point (or if all previously created guaranteed restore points have been dropped).

> **Note:** No special requirements exist for using normal restore points.

Enabling Flashback Database

To enable logging for Flashback Database, set the `DB_FLASHBACK_RETENTION_TARGET` initialization parameter and issue the `ALTER DATABASE FLASHBACK ON` statement.

To enable flashback logging:

1. Start SQL*Plus and ensure that the database is mounted, but not open. For example:

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT;
   ```

2. Optionally, set the `DB_FLASHBACK_RETENTION_TARGET` to the length of the desired flashback window in minutes:

   ```
   ALTER SYSTEM SET DB_FLASHBACK_RETENTION_TARGET=4320; # 3 days
   ```

   By default `DB_FLASHBACK_RETENTION_TARGET` is set to one day (1440 minutes).

3. Enable the Flashback Database feature for the whole database:

   ```
   ALTER DATABASE FLASHBACK ON;
   ```

4. Optionally, disable flashback logging for specific tablespaces.

   By default, flashback logs are generated for all permanent tablespaces. If you wish, you can reduce overhead by disabling flashback logging for specific tablespaces as in the following example:

   ```
   ALTER TABLESPACE tbs_3 FLASHBACK OFF;
   ```

   You can re-enable flashback logging for a tablespace later with this command:

   ```
   ALTER TABLESPACE tbs_3 FLASHBACK ON;
   ```

   Note that if you disable Flashback Database for a tablespace, then you must take its datafiles offline before running `FLASHBACK DATABASE`.

You can disable flashback logging for the entire database with this command:

```
ALTER DATABASE FLASHBACK OFF;
```

Enabling Flashback Database on a **physical standby database** enables you to flash back a standby database. Flashback Database of standby databases has a number of applications in the Data Guard environment. See *Oracle Data Guard Concepts and Administration* for details.

Creating Normal and Guaranteed Restore Points

To create normal or guaranteed restore points, use the `CREATE RESTORE POINT` SQL statement, providing a name for the restore point and specifying whether it is to be a guaranteed restore point or a normal one (the default).

To create a restore point:

1. Connect SQL*Plus to a target database.
2. Ensure that the database is open or mounted. If the database is mounted, then it must have been shut down cleanly (unless it is a physical standby database).
3. Run the `CREATE RESTORE POINT` statement.

   The following example shows how to create a normal restore point in SQL*Plus:

   ```
   SQL> CREATE RESTORE POINT before_upgrade;
   ```

   This example shows how to create a guaranteed restore point:

   ```
   SQL> CREATE RESTORE POINT before_upgrade GUARANTEE FLASHBACK DATABASE;
   ```

See Also:

- *Oracle Database SQL Language Reference* for reference information about the SQL `CREATE RESTORE POINT` statement
- "Listing Restore Points" on page 10-9 to learn how to list restore point
- "Dropping Restore Points" on page 11-9 to learn how to delete restore points

Configuring the Environment for Optimal Flashback Database Performance

Maintaining flashback logs imposes comparatively limited overhead on an Oracle database instance. Changed blocks are written from memory to the flashback logs at relatively infrequent, regular intervals, to limit processing and I/O overhead.

To achieve good performance for large production databases with Flashback Database enabled, Oracle recommends the following:

- Use a fast file system for your flash recovery area, preferably without operating system file caching.

  Files that the database creates in the flash recovery area, including flashback logs, are typically large. Operating system file caching is typically not effective for these files, and may actually add CPU overhead for reading from and writing to these files. Thus, it is recommended to use a file system that avoids operating system file caching, such as ASM.

- Configure enough disk spindles for the file system that will hold the flash recovery area.

  For large production databases, multiple disk spindles may be needed to support the required disk throughput for the database to write the flashback logs effectively.

- If the storage system used to hold the flash recovery area does not have nonvolatile RAM, then try to configure the file system on striped storage volumes.

  Use a relatively small stripe size such as 128 KB. This technique enables each write to the flashback logs to be spread across multiple spindles, improving performance.

- For large databases, set the initialization parameter `LOG_BUFFER` to at least 8 MB.

  This setting ensures that the database allocates maximum memory (typically 16MB) for writing flashback database logs.

The overhead of logging for Flashback Database depends on the mixture of reads and writes in the database workload. The more write-intensive the workload, the higher the overhead caused by turning on logging for Flashback Database. Queries do not change data and thus do not contribute to logging activity for Flashback Database.

Configuring RMAN in a Data Guard Environment

If you use RMAN in a Data Guard environment, then you can use the `CONFIGURE` command to register and configure settings for the physical databases in this environment. RMAN uses the `DB_UNIQUE_NAME` initialization parameter to distinguish one database from another. Thus, it is critical that you maintain the uniqueness of the `DB_UNIQUE_NAME` in the Data Guard environment.

RMAN must be connected to a recovery catalog when you create or alter a configuration for a database in the Data Guard environment. If you use the `SET DBID` command to set the DBID in the **RMAN session**, then you can configure a standby database even when RMAN is not connected as `TARGET` to a database in the Data Guard environment. You can even create a configuration for a standby database that has not yet been created.

You can use the following forms of the `CONFIGURE` command:

- `CONFIGURE DB_UNIQUE_NAME` defines a connection to a **physical standby database** and implicitly registers the new database.

  New standby databases are also automatically registered when RMAN connects as `TARGET` to a standby database for the first time.

- `CONFIGURE FOR DB_UNIQUE_NAME` configures settings for a database in the Data Guard environment.

  For example, you can configure channels, default devices, and so on for a specified database or for all databases in the environment. You can use `SHOW ALL FOR DB_UNIQUE_NAME` to show the configuration for a specific standby database or `SHOW ALL FOR DB_UNIQUE_NAME ALL` to show configurations for all known databases.

A Data Guard environment involves many considerations that are only relevant for Data Guard. For example, you can configure an **archived redo log deletion policy** based on whether archived logs are transferred to or applied on a standby database.

See Also:

- *Oracle Data Guard Concepts and Administration* to learn how to configure the RMAN environment for use with a standby database
- *Oracle Database Backup and Recovery Reference* for a complete explanation of the `CONFIGURE ARCHIVELOG DELETION POLICY` command and the conditions under which archived logs are eligible for deletion

6

Configuring the RMAN Environment: Advanced Topics

This chapter describes how to perform setup and configuration tasks. This chapter contains the following topics:

- Configuring Advanced Channel Options
- Configuring Advanced Backup Options
- Configuring Auxiliary Instance Datafile Names
- Configuring the Snapshot Control File Location
- Configuring RMAN for Use with a Shared Server
- Enabling Lost Write Detection

Configuring Advanced Channel Options

While "Configuring Channels" on page 5-4 explains the basics for configuring channels, this section explains more advanced **channel** topics. This section contains the following topics:

- About Channel Control Options
- Configuring Specific Channel Parameters

> **See Also:** "RMAN Channels" on page 3-3 for a conceptual overview of configured and allocated channels, and *Oracle Database Backup and Recovery Reference* for `CONFIGURE` syntax

About Channel Control Options

Whether you allocate channels manually or use **automatic channel allocation**, you can use channel commands and options to control behavior. Table 6–1 summarizes the ways in which you can control channel behavior. Unless noted, all channel parameters are supported in both `CONFIGURE CHANNEL` and `ALLOCATE CHANNEL` commands.

Table 6–1 Channel Control Options

Type of Channel Control	Commands
Limit I/O bandwidth consumption	You can use the RATE channel parameter to act as a throttling mechanism for backups.
Limit backup sets and pieces	You can use the MAXPIECESIZE channel parameter to set limits on the size of backup pieces. You can also use the MAXSETSIZE parameter on the BACKUP and CONFIGURE commands to set a limit for the size of backup sets.
Vendor-specific instructions	You can use the PARMS channel parameter to specify vendor-specific information for a **media manager**. You can also use the SEND command to send vendor-specific commands to a media manager.
Channel parallelism for backup and restore operations	You can use CONFIGURE DEVICE TYPE ... PARALLELISM for persistent channel parallelism or multiple ALLOCATE CHANNEL commands for job-level parallelism.
Connection settings for database instances	You can specify which instance performs an operation with the CONNECT channel parameter.

See Also: *Oracle Database Backup and Recovery Reference* for ALLOCATE CHANNEL syntax, and *Oracle Database Backup and Recovery Reference* for CONFIGURE syntax

Configuring Specific Channel Parameters

In addition to configuring parameters that apply to all channels of a particular type, you can also configure parameters that apply to one specific channel. Run the CONFIGURE CHANNEL *n* command (where *n* is a positive integer less than 255) to configure a specific channel.

When manually numbering channels, you must specify one or more channel options (for example, MAXPIECESIZE or FORMAT) for each channel. When you use that specific numbered channel in a backup, the configured settings for that channel will be used instead of the configured generic channel settings.

Configure specific channels by number when it is necessary to control the parameters set for each channel separately. This technique could be necessary in the following situations:

- When running an Oracle Real Application Clusters (Oracle RAC) database in which individual nodes do not have access to the full set of backups. Each channel must be configured with a node-specific connect string so that all backups are accessible by at least one channel.
- When using a media manager that requires different PARMS settings on each channel.

See Also: *Oracle Real Application Clusters Administration and Deployment Guide* to learn about RMAN backups in an Oracle RAC environment

Configuring Specific Channels: Examples

In this example, you want to send disk backups to two different disks. Configure disk channels as follows:

```
CONFIGURE DEFAULT DEVICE TYPE TO disk;        # backup goes to disk
CONFIGURE DEVICE TYPE disk PARALLELISM 2;     # two channels used in in parallel
```

```
CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT '/disk1/%U' # 1st channel to disk1
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT '/disk2/%U' # 2nd channel to disk2
BACKUP DATABASE; # backup - first channel goes to disk1 and second to disk2
```

Assume a different case in which you have two tape drives and want each tape drive to use tapes from a different tape media family. Configure your default output device and default tape channels as shown in the following example to parallelize the database backup.

Example 6–1 Configuring Channel Parallelism for Tape Devices

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;    # backup goes to sbt
CONFIGURE DEVICE TYPE sbt PARALLELISM 2; # two sbt channels allocated by default
# Configure channel 1 to pool named first_pool
CONFIGURE CHANNEL 1 DEVICE TYPE sbt
  PARMS 'ENV=(OB_MEDIA_FAMILY=first_pool)';
# configure channel 2 to pool named second_pool
CONFIGURE CHANNEL 2 DEVICE TYPE sbt
  PARMS 'ENV=(OB_MEDIA_FAMILY=second_pool)';
BACKUP DATABASE; # first stream goes to 'first_pool' and second to 'second_pool'
```

Note that in Example 6–1, the backup data is divided between the two tape devices. Each configured channel backs up roughly half the total data.

Relationship Between CONFIGURE CHANNEL and Parallelism Setting

The `PARALLELISM` setting is not constrained by the number of specifically configured channels. For example, if you back up to 20 different tape devices, then you can configure 20 different SBT channels, each with a manually assigned number (from 1 to 20) and each with a different set of channel options. In such a situation, you can set `PARALLELISM` to any value up to the number of devices, in this instance 20.

RMAN always numbers parallel channels starting with `1` and ending with the `PARALLELISM` setting. For example, if the default device is SBT and parallelism is set to 3, then RMAN names the channels as follows:

```
ORA_SBT_TAPE_1
ORA_SBT_TAPE_2
ORA_SBT_TAPE_3
```

RMAN always uses the name `ORA_SBT_TAPE_`*n* even if you configure `DEVICE TYPE sbt` (not the synonymous `sbt_tape`). RMAN always allocates the number of channels specified in `PARALLELISM`, using specifically configured channels if you have configured them and generic channels if you have not. Note that if you configure specific channels with numbers higher than the parallelism setting, RMAN will not use these channels.

See Also: "RMAN Channels" on page 3-3 to learn about channels

Configuring Advanced Backup Options

"Configuring the Environment for RMAN Backups" on page 5-1 explains the basics for configuring RMAN to make backups. This section explained more advanced configuration options. This section contains the following topics:

- Configuring the Maximum Size of Backup Sets
- Configuring the Maximum Size of Backup Pieces
- Configuring Backup Duplexing

- Configuring Tablespaces for Exclusion from Whole Database Backups
- Configuring the Backup Compression Algorithm
- Configuring Backup Encryption

Configuring the Maximum Size of Backup Sets

In tape backups, it is possible for a **multiplexed backup set** to span multiple tapes, which means that blocks from each datafile in the backup set are written to multiple tapes. If one tape of a multivolume backup set fails, then you lose the data on all the tapes rather than just one. If a backup is not a **multisection backup**, then a backup set always includes a whole datafile rather than a partial datafile. You can use `MAXSETSIZE` to specify that each backup set should fit on one tape rather than spanning multiple tapes.

The `CONFIGURE MAXSETSIZE` command limits the size of backup sets created on a channel. This `CONFIGURE` setting applies to any channel, whether manually allocated or configured, when the `BACKUP` command is used to create backup sets. The default value is given in bytes and is rounded down to the lowest kilobyte value.

The value set by the `CONFIGURE MAXSETSIZE` command is a default for the given channel. You can override the configured `MAXSETSIZE` value by specifying a `MAXSETSIZE` option for an individual `BACKUP` command.

Assume that you issue the following commands at the RMAN prompt:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS 'ENV=(OB_MEDIA_FAMILY=first_pool)';
CONFIGURE MAXSETSIZE TO 7500K;
BACKUP TABLESPACE users;
BACKUP TABLESPACE tools MAXSETSIZE 5G;
```

The results will be as follows:

- The backup of the `users` tablespace uses the configured **SBT** channel and the configured default `MAXSETSIZE` setting of `7500K`.
- The backup of the `tools` tablespace uses the `MAXSETSIZE` setting of `5G` used in the `BACKUP` command.

See Also:

- "Limiting the Size of Backup Sets with BACKUP ... MAXSETSIZE" on page 9-2
- *Oracle Database Backup and Recovery Reference* for `BACKUP` syntax

Configuring the Maximum Size of Backup Pieces

Backup piece size is an issue in situations where it exceeds the maximum file size permitted by the file system or media management software. You can use the `MAXPIECESIZE` parameter of the `CONFIGURE CHANNEL` or `ALLOCATE CHANNEL` command to limit the size of backup pieces.

For example, to always limit the backup piece size to 2 GB or less, you can configure the automatic `DISK` channel as follows and then run `BACKUP DATABASE`:

```
CONFIGURE CHANNEL DEVICE TYPE DISK MAXPIECESIZE 2G;
BACKUP DATABASE;
```

Note: In version 2.0 of the media management API, media management vendors can specify the maximum size of a backup piece that can be written to their media manager. RMAN respects this limit regardless of the settings you configure for `MAXPIECESIZE`.

See Also: *Oracle Database Backup and Recovery Reference* to learn about the `CONFIGURE CHANNEL ... MAXPIECESIZE` command

Configuring Backup Duplexing

You can use the `CONFIGURE ... BACKUP COPIES` command to specify how many copies of each backup piece should be created on the specified device type for the specified type of file. This type of backup is known as a **duplexed backup set**. The `CONFIGURE` settings for duplexing only affect backups of datafiles, control files, and archived logs into backup sets, and do not affect image copies.

Note: A **control file autobackup** is *never* duplexed.

RMAN can duplex backups to either disk or tape, but cannot duplex backups to tape and disk simultaneously. When backing up to tape, ensure that the number of copies does not exceed the number of available tape devices. The following examples show possible duplexing configurations:

```
# Makes 2 disk copies of each datafile and control file backup set
# (autobackups excluded)
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 2;
# Makes 3 copies of every archived redo log backup to tape
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE sbt TO 3;
```

To return a `BACKUP COPIES` configuration to its default value, run the same `CONFIGURE` command with the `CLEAR` option, as in the following example:

```
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE sbt CLEAR;
```

By default, `CONFIGURE ... BACKUP COPIES` is set to `1` for each device type.

Note: If you do not want to create a persistent copies configuration, then you can specify copies with the `BACKUP COPIES` and the `SET BACKUP COPIES` commands.

See Also:

- "Multiple Copies of RMAN Backups" on page 7-10 for an overview of duplexed backups
- "Duplexing Backup Sets" on page 9-6 to learn how to create duplexed backups
- *Oracle Database Backup and Recovery Reference* for `BACKUP` syntax
- *Oracle Database Backup and Recovery Reference* for `CONFIGURE` syntax
- *Oracle Database Backup and Recovery Reference* for `SET` syntax

Configuring Tablespaces for Exclusion from Whole Database Backups

In some cases you may want to exclude specified tablespace part of the regular backup schedule, as in these cases:

- A tablespace is easy to rebuild, so it is more cost-effective to rebuild it than back it up every day.
- A tablespace contains temporary or test data that you do not need to back up.
- A tablespace does not change often and therefore should be backed up on a different schedule from other backups.

You can run `CONFIGURE EXCLUDE FOR TABLESPACE` to exclude the specified tablespace from the `BACKUP DATABASE` command. The exclusion condition applies to any datafiles that you add to this tablespace in the future.

For example, you can exclude testing tablespaces `cwmlite` and `example` from whole database backups as follows:

```
CONFIGURE EXCLUDE FOR TABLESPACE cwmlite;
CONFIGURE EXCLUDE FOR TABLESPACE example;
```

If you run the following command, then RMAN backs up all tablespaces in the database except `cwmlite` and `example`:

```
BACKUP DATABASE;
```

You can still back up the configured tablespaces by explicitly specifying them in a `BACKUP` command or by specifying the `NOEXCLUDE` option on a `BACKUP DATABASE` command. For example, you can enter one of the following commands:

```
# backs up the whole database, including cwmlite and example
BACKUP DATABASE NOEXCLUDE;
BACKUP TABLESPACE cwmlite, example;  # backs up only cwmlite and example
```

You can disable the exclusion feature for `cwmlite` and `example` as follows:

```
CONFIGURE EXCLUDE FOR TABLESPACE cwmlite CLEAR;
CONFIGURE EXCLUDE FOR TABLESPACE example CLEAR;
```

RMAN includes these tablespaces in future whole database backups.

See Also:

- *Oracle Database Backup and Recovery Reference* for `BACKUP` syntax
- *Oracle Database Backup and Recovery Reference* for `CONFIGURE` syntax

Configuring the Backup Compression Algorithm

RMAN supports **binary compression** of backup sets. The supported algorithms are `BZIP2` (default) and `ZLIB`. The `BZIP2` algorithm is optimized for maximum compression, whereas the `ZLIB` algorithm is optimized for CPU efficiency. `BZIP2` consumes more CPU resource than `ZLIB`, but will usually produce more compact backups. The `COMPATIBLE` initialization parameter must be set to 11.0.0 or higher for `ZLIB` compression, which requires the Oracle Advanced Compression option.

You can configure the compression algorithm with the following syntax, where *alg_name* is a placeholder specifying either `BZIP2` or `ZLIB`.

Example 6–2 Configuring the Backup Compression Algorithm

```
CONFIGURE COMPRESSION ALGORITHM TO 'alg_name';
```

Configuring Backup Encryption

For improved security, you can configure **backup encryption** for RMAN backup sets. Encrypted backups cannot be read if they are obtained by unauthorized users. This feature requires the Enterprise Edition of the database.

About Backup Encryption

The V$RMAN_ENCRYPTION_ALGORITHMS view contains a list of encryption algorithms supported by RMAN. If no encryption algorithm is specified, then the default encryption algorithm is 128-bit Advanced Encryption Standard (AES). Note that RMAN encryption requires the COMPATIBLE initialization parameter at a target database to be at least 10.2.0.

RMAN offers the following encryptions modes:

- Transparent Encryption of Backups

  This is the default mode and uses the Oracle wallet. A wallet is a password-protected container used to store authentication and signing credentials, including private keys, certificates, and trusted certificates needed by SSL.

- Password Encryption of Backups

  This mode uses only password protection. You must provide a password when creating and restoring encrypted backups.

- Dual Mode Encryption of Backups

  This mode requires either the wallet or a password.

> **Note:** Wallet-based encryption is more secure than password-based encryption because no passwords are involved. You should use password-based encryption only when absolutely necessary because your backups need to be transportable.

Encrypted backups are decrypted automatically during restore and recovery, as long as the required decryption keys are available. Each backup set gets a separate key. The key is stored in encrypted form in the backup piece. The backup is decrypted with keys obtained by means of a user-supplied password or the Oracle wallet.

To create encrypted backups on disk with RMAN, the database must use the Advanced Security Option. The **Oracle Secure Backup SBT** is the only supported interface for making encrypted RMAN backups directly to tape. RMAN issues an ORA-19916 error if you attempt to create encrypted RMAN backups using an SBT library other than Oracle Secure Backup. Note that the Advanced Security Option is *not* required when making encrypted backups using the Oracle Secure Backup SBT.

When you use the BACKUP BACKUPSET command with encrypted backup sets, the backup sets are backed up in encrypted form. Because BACKUP BACKUPSET copies an already-encrypted backup set to disk or tape, no decryption key is needed during BACKUP BACKUPSET. The data is never decrypted during any part of the operation. The BACKUP BACKUPSET command can neither encrypt nor decrypt backup sets.

> **See Also:** *Oracle Database Advanced Security Administrator's Guide* for details about configuring the Oracle wallet

Transparent Encryption of Backups Transparent encryption can create and restore encrypted backups with no DBA intervention, as long as the required Oracle key management infrastructure is available. Transparent encryption is best suited for day-to-day backup operations, where backups are restored to the same database from which they were created. Transparent encryption is the default for RMAN encryption.

When using transparent encryption, you must first configure an Oracle wallet for each database, as described in *Oracle Database Advanced Security Administrator's Guide*. Transparent backup encryption supports both the encrypted and autologin forms of the Oracle wallet. When using the Oracle wallet, the wallet must be opened before you can perform backup encryption. When using the autologin wallet, encrypted backup operations can be done at any time, because the autologin wallet is always open.

> **Caution:** If you use an autologin wallet, then do not back it up along with your encrypted backup data, because users can read the encrypted backups if they obtain both the backups and the autologin wallet. It is safe to back up the Oracle wallet because that form of the wallet cannot be used without the wallet password.

After the Oracle wallet is configured, encrypted backups can be created and restored with no further DBA intervention. Note that if some columns in the database are encrypted with Transparent Data Encryption, and if those columns are backed up using backup encryption, then those columns will be encrypted a second time during the backup. When the backup sets are decrypted during a restore, the encrypted columns are returned to their original encrypted form.

Because the Oracle key management infrastructure archives all previous master keys in the Oracle wallet, changing or resetting the current database master key will not affect your ability to restore encrypted backups performed with an older master key. You can reset the database master key at any time. RMAN will always be able to restore all encrypted backups that were ever created by this database.

> **Caution:** If you lose your Oracle wallet, then you will be unable to restore any transparently-encrypted backups.

Password Encryption of Backups Password encryption requires that the DBA provide a password when creating and restoring encrypted backups. Restoring a password-encrypted backup requires the same password used to create the backup.

Password encryption is useful for backups that will be restored at remote locations, but which must remain secure in transit. Password encryption cannot be persistently configured. You do not need to configure an Oracle wallet if password encryption will be used exclusively.

> **Caution:** If you forget or lose the password that you used to encrypt a password-encrypted backup, then you will be unable to restore the backup.

To use password encryption, use the `SET ENCRYPTION ON IDENTIFIED BY` *`password`* `ONLY` command in your RMAN scripts.

Dual Mode Encryption of Backups Dual-mode encrypted backups can be restored either transparently or by specifying a password. Dual-mode encrypted backups are useful

when you create backups that are normally restored onsite using the Oracle wallet, but which occasionally need to be restored offsite, where the Oracle wallet is not available.

When restoring a dual-mode encrypted backup, you can use either the Oracle wallet or a password for decryption.

> **Caution:** If you forget or lose the password that you used to encrypt a dual-mode encrypted backup and you also lose your Oracle wallet, then you will be unable to restore the backup.

To create dual-mode encrypted backup sets, specify the `SET ENCRYPTION ON IDENTIFIED BY` *`password`* command in your RMAN scripts.

Configuring RMAN Backup Encryption Modes

You can use the `CONFIGURE` command to persistently configure transparent encryption of backups. You can use the command to specify the following:

- Whether to use transparent encryptions for backups of all database files
- Whether to use transparent encryptions for backups of specific tablespaces
- Which algorithm to use for encrypting backups

You can also use the `SET ENCRYPTION` command to perform the following actions:

- Override the encryption settings specified by the `CONFIGURE ENCRYPTION` command. For example, you can use `SET ENCRYPTION OFF` to create an unencrypted backup, even though a database is configured for encrypted backups.
- Set a password for backup encryption, persisting until the RMAN client exits. Because of the sensitive nature of passwords, RMAN does not permit configuration of passwords that persist across RMAN sessions.

Note that no persistent configuration controls whether archived redo log backups are encrypted. Backup sets containing archived redo log files are encrypted if any of the following are true:

- `SET ENCRYPTION ON` is in effect at the time that the archive log backup is being created.
- Encryption is configured for backups of the whole database or at least one tablespace.

This behavior ensures that the redo associated with any encrypted backup of a datafile is also encrypted.

To configure the environment so that all RMAN backups are encrypted:

1. Set up the Oracle wallet as explained in *Oracle Database Advanced Security Administrator's Guide*.
2. Issue the following RMAN command:

   ```
   CONFIGURE ENCRYPTION FOR DATABASE ON;
   ```

   At this stage, all RMAN backup sets created by this database will use transparent encryption by default.

You can explicitly override the persistent encryption configuration for an **RMAN session** with the following command:

```
SET ENCRYPTION ON;
```

The encryption setting remains in effect until you issue the SET ENCRYPTION OFF command during an RMAN session, or change the persistent setting again with the following command:

```
CONFIGURE ENCRYPTION FOR DATABASE OFF;
```

Configuring the Backup Encryption Algorithm

You can use the CONFIGURE command to persistently configure the default algorithm to use for encryption when writing backup sets. Possible values are listed in V$RMAN_ENCRYPTION_ALGORITHMS. The default algorithm is AES 128-bit.

To configure the default backup encryption algorithm:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Ensure that the target database is mounted or open.
3. Execute the CONFIGURE ENCRYPTION ALGORITHM command, specifying a valid value from V$RMAN_ENCRYPTION_ALGORITHMS.ALGORITHM_NAME.

   The following example configures the algorithm to AES 256-bit encryption:

   ```
   CONFIGURE ENCRYPTION ALGORITHM TO 'AES256';
   ```

Configuring Auxiliary Instance Datafile Names

Assume that you are performing **tablespace point-in-time recovery (TSPITR)** or performing data transfer with RMAN. In this case, you may want to set the names of datafiles in the **auxiliary instance** before starting the TSPITR or database duplication. The command is as follows, where *datafileSpec* identifies some datafile by its original name or datafile number, and *filename* is the new path for the specified file:

```
CONFIGURE AUXNAME FOR datafileSpec TO 'filename';
```

For example, you might configure a new auxiliary name for datafile 2 as follows:

```
CONFIGURE AUXNAME FOR DATAFILE 2 TO '/newdisk/datafiles/df2.df';
```

As with other settings, the CONFIGURE command setting persists across RMAN sessions until cleared with CONFIGURE . . . CLEAR, as shown in the following example:

```
CONFIGURE AUXNAME FOR DATAFILE 2 CLEAR;
```

If you are performing TSPITR or running the DUPLICATE command, then by using CONFIGURE AUXNAME you can preconfigure the filenames for use on the auxiliary database without manually specifying the auxiliary filenames during the procedure.

When renaming files with the DUPLICATE command, CONFIGURE AUXNAME is an alternative to SET NEWNAME command. The difference is that after you set the AUXNAME the first time, you do not need to reset the filename when you issue another DUPLICATE command; the AUXNAME setting remains in effect until you issue CONFIGURE AUXNAME . . . CLEAR. In contrast, you must reissue the SET NEWNAME command every time you rename files.

See Chapter 20, "Performing RMAN Tablespace Point-in-Time Recovery (TSPITR)," for more details on using CONFIGURE AUXNAME in connection with TSPITR, and Chapter 23, "Duplicating a Database," for more on using CONFIGURE AUXNAME in performing database duplication.

Configuring the Snapshot Control File Location

When RMAN needs to resynchronize the recovery catalog with a read-consistent version of the control file, it creates a temporary **snapshot control file**. RMAN needs a snapshot control file when resynchronizing with the recovery catalog or when making a backup of the current control file.

The default location for the snapshot control file is platform-specific and depends on the Oracle home of each target database. For example, the default filename on some Linux platforms is `$ORACLE_HOME/dbs/snapcf_@.f`. If a **flash recovery area** is configured for a target database, then the default location for the snapshot control file is *not* the flash recovery area.

Viewing the Configured Location of the Snapshot Control File

You can see the current snapshot location by running the `SHOW` command. This example shows a snapshot location that is determined by the default rule:

```
RMAN> SHOW SNAPSHOT CONTROLFILE NAME;
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/dbs/snapcf_trgt.f'; # default
```

This example shows a snapshot control file that has a nondefault filename:

```
RMAN> SHOW SNAPSHOT CONTROLFILE NAME;
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/oradata/trgt/snap_trgt.ctl';
```

Setting the Location of the Snapshot Control File

Use the `CONFIGURE SNAPSHOT CONTROLFILE NAME TO '`*`filename`*`'` command to change the name of the snapshot control file. Subsequent snapshot control files that RMAN creates use the specified filename.

For example, start RMAN and then enter:

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/oradata/trgt/snap_trgt.ctl';
```

You can also set the snapshot control file name to a **raw device**.

To reset the snapshot control file location to the default, run the `CONFIGURE SNAPSHOT CONTROLFILE LOCATION CLEAR` command.

See Also:

- "Resynchronizing the Recovery Catalog" on page 12-22
- *Oracle Real Application Clusters Administration and Deployment Guide* for handling snapshot control files in Oracle RAC configurations

Configuring RMAN for Use with a Shared Server

RMAN cannot connect to a target database through a shared server dispatcher. RMAN requires a dedicated server process. If your target database is configured for shared server, then you must modify your Oracle Net configuration to provide dedicated server processes for RMAN connections.

To ensure that RMAN does not connect to a dispatcher when a target database is configured for a shared server, the net service name used by RMAN must include `(SERVER=DEDICATED)` in the `CONNECT_DATA` attribute of the connect string.

Oracle Net configuration varies greatly from system to system. The following procedure illustrates only one method. This scenario assumes that the following service name in the `tnsnames.ora` connects to a target database using the shared server architecture, where `inst1` is a value of the `SERVICE_NAMES` initialization parameter:

```
inst1_shs =
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=inst1_host)(port=1521))
    (CONNECT_DATA=(SERVICE_NAME=inst1)(SERVER=shared))
  )
```

To use RMAN with a shared server:

1. Create a net service name in the `tnsnames.ora` file that connects to the nonshared SID. For example, enter:

   ```
   inst1_ded =
     (DESCRIPTION=
       (ADDRESS=(PROTOCOL=tcp)(HOST=inst1_host)(port=1521))
       (CONNECT_DATA=(SERVICE_NAME=inst1)(SERVER=dedicated))
     )
   ```

2. Start SQL*Plus and then connect using both the shared server and dedicated server service names to confirm the mode of each session.

   For example, connect with `SYSDBA` privileges to `inst1_ded` and then execute the following `SELECT` statement (sample output included):

   ```
   SQL> SELECT SERVER
     2  FROM   V$SESSION
     3  WHERE  SID = (SELECT DISTINCT SID FROM V$MYSTAT);

   SERVER
   ---------
   DEDICATED
   1 row selected.
   ```

   To connect to a shared server session, you could connect with `SYSDBA` privileges to `inst1_shs` and then execute the following `SELECT` statement (sample output included):

   ```
   SQL> SELECT SERVER
     2  FROM   V$SESSION
     3  WHERE  SID = (SELECT DISTINCT SID FROM V$MYSTAT);

   SERVER
   ---------
   SHARED
   1 row selected.
   ```

3. Start RMAN and connect to the target database using the dedicated service name. Optionally, connect to a recovery catalog. For example, enter:

   ```
   % rman
   RMAN> CONNECT TARGET SYS@inst1_ded

   target database Password: password
   connected to target database: INST1 (DBID=39525561)

   RMAN> CONNECT CATALOG rman@catdb
   ```

See Also: Your platform-specific Oracle documentation and your *Oracle Database Net Services Reference* for a complete description of Oracle Net connect string syntax

Enabling Lost Write Detection

A data block **lost write** occurs when an I/O subsystem acknowledges the completion of the block write, while in fact the write did not occur in the persistent storage. On a subsequent block read, the I/O subsystem returns the stale version of the data block, which might be used to update other blocks of the database, thereby corrupting it.

You can set the `DB_LOST_WRITE_PROTECT` initialization parameter to `TYPICAL` or `FULL` so that a database records buffer cache block reads in the redo log. The default setting is `NONE`. When the parameter is set to `TYPICAL`, the instance logs buffer cache reads for read/write tablespaces in the redo log, but not read-only tablespaces. When set to `FULL`, the instance also records reads for read-only tablespaces. The performance overhead for `TYPICAL` mode is roughly 5-10%. For Oracle RAC the overhead for `FULL` mode can increase to 20%.

Lost write detection is most effective when used with Data Guard. In this case, you set `DB_LOST_WRITE_PROTECT` in both primary and standby databases. When a standby database applies redo during managed recovery, it reads the corresponding blocks and compares the SCNs with the SCNs in the redo log. If the block SCN on the primary database is lower than on the standby database, then it detects a lost write on the primary database and throws an external error (`ORA-752`). If the SCN is higher, it detects a lost write on the standby database and throws an internal error (`ORA-600 [3020]`). In either case, the standby database writes the reason for the failure in the alert log and trace file.

To repair a lost write on a primary database, you must initiate failover to the standby database. To repair a lost write on a standby database, you must re-create the entire standby database or restore a backup of only the affected files.

Enabling lost write detection is also useful when not using Data Guard. In this case, you can encounter a lost write in two ways: during normal database operation or during media recovery. In the first case, there is no deterministic way to detect the error. Indirect symptoms such as inconsistent tables cannot be unambiguously traced to the lost write. If you retained a backup made *before* the suspected lost write, however, then you can restore this backup to an alternative location and recover it. To diagnose the problem, recover the database or tablespace to the SCN of the stale block read, which will generate the lost write error (`ORA-752`).

In the case of a lost write error encountered during media recovery, the only response is to open the database with the `RESETLOGS` option. The database is in consistent state, but all data after the `RESETLOGS` SCN is lost. Note that if you recover a backup made after database creation, you have no guarantee that other stale blocks have not already corrupted the database. This possibility exists because the restored backup may have been made after an earlier lost write. To guarantee that no lost writes have corrupted the database, you must perform media recovery from database creation, which is not a practical strategy for most database environments.

See Also:

- *Oracle Data Guard Concepts and Administration* to learn how to use a standby database for lost write detection and repair
- *Oracle Database Reference* to learn about the `DB_LOST_WRITE_PROTECT` initialization parameter

Part III

Backing Up and Archiving Data

The chapters in this part describe how to use the RMAN utility to perform advanced backup and recovery operations, and explain RMAN performance tuning and troubleshooting.

This part contains these chapters:

- Chapter 7, "RMAN Backup Concepts"
- Chapter 8, "Backing Up the Database"
- Chapter 9, "Backing Up the Database: Advanced Topics"

7

RMAN Backup Concepts

This chapter describes the general concepts that you need to understand to make any type of RMAN backup. This chapter contains the following topics:

- Consistent and Inconsistent RMAN Backups
- Online Backups and Backup Mode
- Backup Sets
- Image Copies
- Multiple Copies of RMAN Backups
- Control File and Server Parameter File Autobackups
- Incremental Backups
- Backup Retention Policies

Consistent and Inconsistent RMAN Backups

The RMAN command for making backups is `BACKUP`. The RMAN `BACKUP` command supports backing up the following types of files:

- Datafiles and control files
- Server parameter file
- Archived redo logs
- RMAN backups

Although the database depends on other types of files, such as network configuration files, password files, and the contents of the Oracle home, you cannot back up these files with RMAN. Likewise, some features of Oracle, such as external tables, may depend upon files other than the datafiles, control files, and redo log. RMAN cannot back up these files. Use some non-RMAN backup solution for any files not in the preceding list.

When you execute the `BACKUP` command in RMAN, the output is always either one or more backup sets or one or more image copies. A **backup set** is an RMAN-specific proprietary format, whereas an **image copy** is a bit-for-bit copy of a file. By default, RMAN creates backup sets.

Consistent Backups

You can use the `BACKUP` command to make consistent and inconsistent backups of the database. A **consistent backup** occurs when the database is in a consistent state. A

database is in a consistent state after being shut down with the SHUTDOWN NORMAL, SHUTDOWN IMMEDIATE, or SHUTDOWN TRANSACTIONAL commands. A consistent shutdown guarantees that all redo has been applied to the datafiles. If you mount the database and make a backup at this point, then you can restore the database backup later and open it without performing media recovery.

Inconsistent Backups

Any database backup that is not consistent is an **inconsistent backup**. A backup made when the database is open is inconsistent, as is a backup made after an instance failure or SHUTDOWN ABORT command. When a database is restored from an inconsistent backup, Oracle must perform media recovery before the database can be opened, applying any pending changes from the redo logs.

> **Note:** RMAN does not permit you to make inconsistent backups when the database is in NOARCHIVELOG mode. If you employ user-managed backup techniques for a NOARCHIVELOG database, then you must not make inconsistent backups of this database.

As long as the database runs in ARCHIVELOG mode, and you back up the archived redo logs and datafiles, inconsistent backups can be the foundation for a sound backup and recovery strategy. Inconsistent backups offer superior availability because you do not have to shut down the database to make backups that fully protect the database.

Online Backups and Backup Mode

When performing a **user-managed backup** of an online tablespace or database, an operating system utility can back up a datafile at the same time that database writer is updating the file. It is possible for the utility to read a block in a half-updated state, so that the block that is copied to the backup media is updated in its first half, while the second half contains older data. This type of **logical corruption** is known as a **fractured block**, that is, a block that is not consistent with respect to an SCN. If this backup needs to be restored later, and if the block requires recovery, then recovery will fail because the block is not usable.

When performing a user-managed online backup, you must place your datafiles into **backup mode** with the ALTER DATABASE or ALTER TABLESPACE statement with the BEGIN BACKUP clause. When a tablespace is in backup mode, the database writes the before image for an entire block to the redo stream before modifying a block. The database also records changes to the block in the online redo log. Backup mode also freezes the **datafile checkpoint** until the file is removed from backup mode. Oracle Database performs this safeguard because it cannot guarantee that a third-party backup tool will copy the file header before copying the data blocks.

Unlike user-managed tools, RMAN does not require extra logging or backup mode because it knows the format of data blocks. RMAN is guaranteed not to back up fractured blocks. During an RMAN backup, a database server session reads each data block and checks whether it is fractured by comparing the block header and footer. If a block is fractured, then the session rereads the block. If the same fracture is found, then the block is considered permanently corrupt. Also, RMAN does not need to freeze the datafile header checkpoint because it knows the order in which the blocks will be read, which enables it to capture a known good checkpoint for the file.

See Also: Chapter , "Making User-Managed Backups of Online Tablespaces and Datafiles" on page 27-5 to learn how to back up online tablespaces when not using RMAN

Backup Sets

When you execute the BACKUP command in RMAN, you create one or more backup sets or image copies. By default, RMAN creates backup sets regardless of whether the destination is disk or a media manager.

This section contains the following topics:

- Backup Sets and Backup Pieces
- Compression for Backup Sets
- Encryption for Backup Sets
- Filenames for Backup Pieces
- Number and Size of Backup Pieces
- Number and Size of Backup Sets
- Multiplexed Backup Sets
- Proxy Copies

Backup Sets and Backup Pieces

RMAN can store backup data in a logical structure called a **backup set**, which is the smallest unit of an RMAN backup. A backup set contains the data from one or more datafiles, archived redo logs, or control files or server parameter file. Backup sets, which are only created and accessed through RMAN, are the only form in which RMAN can write backups to media managers such as tape drives and tape libraries.

A backup set contains one or more binary files in an RMAN-specific format. This file is known as a **backup piece**. A backup set can contain multiple datafiles. For example, you can back up ten datafiles into a single backup set consisting of a single backup piece. In this case, RMAN creates one backup piece as output. The backup set contains only this backup piece.

If you specify the SECTION SIZE parameter on the BACKUP command, then RMAN produces a **multisection backup**. This is a backup of a single large file, produced by multiple channels in parallel, each of which produces one backup piece. Each backup piece contains one **file section** of the file being backed up.

For non-multisection backups, RMAN only records backup sets in the repository that complete successfully. There is no such thing as a partial backup set. This differs from an unsuccessful multisection backup, where it is possible for RMAN metadata to contain a record for a partial backup set. In the latter case, you must use the DELETE command to delete the partial backup set.

Note: RMAN never considers partial backups as candidates for restore and recovery.

See Also: Chapter 8, "Backing Up the Database" to learn how to back up the database

Compression for Backup Sets

When backing up datafiles to backup sets, RMAN can use **unused block compression** to skip datafile blocks. RMAN always skips blocks that have never been used. Under certain conditions, which are listed in the `BACKUP AS BACKUPSET` entry in *Oracle Database Backup and Recovery Reference*, RMAN also skips blocks that are not currently used. Thus, datafile backup sets are typically smaller than datafile copies and take less time to write. Unused block compression is fundamental to how RMAN writes datafiles into backup pieces and cannot be disabled.

RMAN also supports **binary compression** of backup sets. The supported algorithms are `BZIP2` (default) and `ZLIB`. The `BZIP2` algorithm is optimized for maximum compression, whereas the `ZLIB` algorithm is optimized for CPU efficiency. `BZIP2` consumes more CPU resource than `ZLIB`, but will usually produce more compact backups. The `COMPATIBLE` initialization parameter must be set to 11.0.0 or higher for `ZLIB` compression, which requires the Oracle Advanced Compression option.

RMAN compresses the backup set contents before writing them to disk. No extra uncompression steps are required during recovery when you use RMAN compression.

In **backup undo optimization**, RMAN excludes undo not needed for recovery of a backup, that is, for transactions have already been committed. You can enable and disable backup optimization, but backup undo optimization is built-in behavior.

See Also:

- "Configuring the Backup Compression Algorithm" on page 6-6
- "Overview of Backup Optimization" on page 5-23
- *Oracle Database Backup and Recovery Reference* to learn about `BACKUP AS BACKUPSET`

Encryption for Backup Sets

RMAN supports **backup encryption** for backup sets. You can use wallet-based transparent encryption, password-based encryption, or both. You can use the `CONFIGURE ENCRYPTION` command to configure persistent transparent encryption. Use the `SET ENCRYPTION`, command at the **RMAN session** level to specify password-based encryption.

Note: Wallet-based encryption is more secure than password-based encryption because no passwords are involved. You should use password-based encryption only when absolutely necessary because your backups need to be transportable.

To create encrypted backups on disk with RMAN, the database must use the Advanced Security Option. The **Oracle Secure Backup SBT** is the only supported interface for making encrypted RMAN backups directly to tape. Note that the Advanced Security Option is not required when making encrypted backups to the Oracle Secure Backup SBT.

See Also:

- "Configuring Backup Encryption" on page 6-7
- "Encrypting RMAN Backups" on page 9-10

Filenames for Backup Pieces

You can either let RMAN determine a unique name for backup pieces or use the FORMAT clause to specify a name. If you do not specify the FORMAT parameter, then RMAN automatically generates a unique filename with the %U substitution variable in the default backup location. An example of an SBT backup piece name generated by %U is 12i1nk47_1_1. An example of a backup piece on disk is as follows:

```
/d1/orcva/TEST/backupset/2007_12_12/o1_mf_nnndf_TAG20071212T162825_2qyl99jm_.bkp
```

The FORMAT clause supports substitution variables other than %U for generating unique filenames. For example, you can use %d to generate the name of the database, %I for the DBID, %t for the timestamp, and so on.

You can specify up to four FORMAT parameters. If you specify multiple FORMAT parameters, then RMAN uses the multiple FORMAT parameters only when you specify multiple copies. You can create multiple copies by using the BACKUP ... COPIES, SET BACKUP COPIES, or CONFIGURE ... BACKUP COPIES commands.

Note: If you use a media manager, then check your vendor documentation for restrictions on FORMAT such as the size of the name, the naming conventions, and so on.

See Also:

- "Specifying a Format for RMAN Backups" on page 8-3
- *Oracle Database Backup and Recovery Reference* for descriptions of the FORMAT clause and the substitution variables

Number and Size of Backup Pieces

By default a backup set contains one backup piece. To restrict the size of each backup piece, specify the MAXPIECESIZE option of the CONFIGURE CHANNEL or ALLOCATE CHANNEL commands. This option limits backup piece size to the specified number of bytes. If the total size of the backup set is greater than the specified backup piece size, then RMAN creates multiple physical pieces to hold the backup set contents.

You can use this option for media managers that cannot manage a backup piece that spans more than one tape. For example, if a tape can hold 10 GB, but the backup set being created must hold 80 GB of data, then you must instruct RMAN to create backup pieces of 10 GB, small enough to fit on the tapes used with the media manager. In this case, the backup set media will consist of eight tapes. Media managers supporting SBT 2.0 can return a value to RMAN indicating the largest supported backup piece size, which RMAN will use in planning backup activities.

Note that if you specify the SECTION SIZE parameter on the BACKUP command, then RMAN can create a **multisection backup**. In this case, a single backup set can contain multiple backup pieces, each containing a **file section**. The purpose of multisection backups is to enable multiple channels to back up a large file in parallel.

See Also:

- "Configuring the Maximum Size of Backup Pieces" on page 6-4
- *Oracle Database Backup and Recovery Reference* for `ALLOCATE CHANNEL` syntax
- *Oracle Database Backup and Recovery Reference* for `CONFIGURE` syntax

Number and Size of Backup Sets

You use the *backupSpec* clause of the `BACKUP` command to specify the objects to be backed up. Each *backupSpec* clause produces at least one backup set.

The total number and size of backup sets depends for the most part on an internal RMAN algorithm. However, you can influence RMAN behavior with the `MAXSETSIZE` parameter in the `CONFIGURE` or `BACKUP` command. By limiting the size of the backup set, the parameter indirectly limits the number of files in the set and can possibly force RMAN to create additional backup sets. Also, you can specify `BACKUP ... FILESPERSET` to specify the maximum number of files in each backup set.

See Also:

- "About Backup Set Size" on page 9-1
- Chapter 21, "Tuning RMAN Performance" to learn about RMAN buffer management
- *Oracle Database Backup and Recovery Reference* to learn the syntax for the *backupSpec* clause

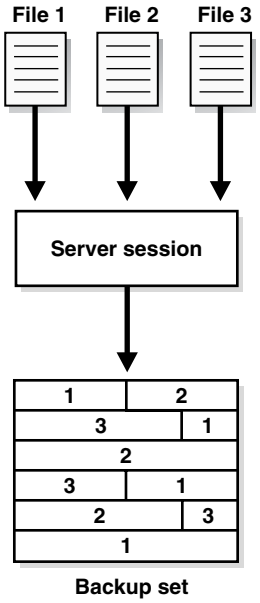
Multiplexed Backup Sets

When creating backup sets, RMAN can simultaneously read multiple files from disk and then write their blocks into the same backup set. For example, RMAN can read from two datafiles simultaneously, and then combine the blocks from these datafiles into a single backup piece. The combination of blocks from multiple files is called backup **multiplexing**. Image copies, by contrast, are never multiplexed.

Note: If RMAN creates a **multisection backup** of a datafile, then the datafile will not be multiplexed with any other datafile or **file section**.

As Figure 7–1 illustrates, RMAN can back up three datafiles into a backup set that contains only one backup piece. This backup piece contains the intermingled data blocks of the three input datafiles.

Figure 7–1 Datafile Multiplexing



RMAN multiplexing is determined by several factors. For example, the `FILESPERSET` parameter of the `BACKUP` command determines how many datafiles to put in each backup set. The `MAXOPENFILES` parameter of `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` defines how many datafiles RMAN can read from simultaneously. The basic multiplexing algorithm is as follows:

- Number of files in each backup set

  This number is the minimum of the `FILESPERSET` setting and the number of files read by each channel. The `FILESPERSET` default is 64.

- The **level of multiplexing**

  This is the number of input files simultaneously read and then written into the same backup piece. The level of multiplexing is the minimum of `MAXOPENFILES` and the number of files in each backup set. The `MAXOPENFILES` default is 8.

Suppose that you back up 12 datafiles with one channel. The number of files in each backup set is 4. The level of multiplexing is the lesser of this number and 8. Thus, the channel simultaneously writes blocks from 4 datafiles into each backup piece.

Now suppose that you back up 50 datafiles with one channel. The number of files in each backup set is 50. The level of multiplexing is the lesser of this number and 8. Thus, the channel simultaneously writes blocks from 8 datafiles into each backup piece.

RMAN multiplexing of backup sets is different from **media manager multiplexing**. One type of media manager multiplexing occurs when the media manager writes the concurrent output from multiple RMAN channels to a single sequential device. Another type occurs when a backup mixes database files and non-database files on the same tape.

Caution: Oracle recommends that you never use media management multiplexing for RMAN backups.

See Also:

- "Allocation of Input Disk Buffers" on page 21-3 to learn how multiplexing affects allocation of disk buffers during backups
- *Oracle Database Backup and Recovery Reference* for BACKUP syntax

Proxy Copies

During a **proxy copy**, RMAN turns over control of the data transfer to a media manager that supports this feature. Proxy copy can only be used with media managers that support it and cannot be used with channels of type DISK. The PROXY option of the BACKUP command specifies that a backup should be a proxy copy.

For each file that you attempt to back up with the BACKUP PROXY command, RMAN queries the media manager to determine whether it can perform a proxy copy. If the media manager cannot proxy copy the file, then RMAN backs the file up as if the PROXY option had not been used. (Use the PROXY ONLY option to force RMAN to fail if a proxy copy cannot be performed.)

Note that control files are never backed up with proxy copy. If the PROXY option is specified on an operation backing up a control file, then it is silently ignored for the purposes of backing up the control file.

See Also:

- *Oracle Database Reference* for more information about V$PROXY_DATAFILE and V$PROXY_ARCHIVEDLOG
- *Oracle Database Backup and Recovery Reference* for the BACKUP command and the PROXY option

Image Copies

An **image copy** is an exact copy of a single datafile, archived redo log file, or control file. Image copies are not stored in an RMAN-specific format. They are identical to the results of copying a file with operating system commands. RMAN can use image copies during RMAN restore and recover operations, and you can also use image copies with non-RMAN restore and recovery techniques.

RMAN-Created Image Copies

To create image copies and have them recorded in the RMAN repository, you run the RMAN BACKUP AS COPY command. Alternatively, you can configure the default backup type for disk as image copies. A database server session is used to create the copy. The server session also performs actions such as validating the blocks in the file and recording the image copy in the RMAN repository.

As with backup pieces, FORMAT variables are also used to specify the names of image copies. The default format %U, which was explained in "Filenames for Backup Pieces" on page 7-5, is defined differently for image copies. The following example shows name for datafile 2 generated by %U:

```
/d1/oracle/work/orcva/RDBMS/datafile/o1_mf_sysaux_2qylngm3_.dbf
```

When creating image copies, you can also name the output copies with the DB_FILE_NAME_CONVERT parameter of the BACKUP command. This parameter works identically to the DB_FILE_NAME_CONVERT initialization parameter. Pairs of filename prefixes are provided to change the names of the output files. If a file is not converted

by any of the pairs, then RMAN uses the FORMAT specification: if no FORMAT is specified, then RMAN uses the default format %U.

Example 7–1 copies the datafiles whose filename is prefixed with /maindisk/oradata/users so that they are prefixed with /backups/users_ts.

Example 7–1 Specifying Filenames with DB_FILE_NAME_CONVERT

```
BACKUP AS COPY
  TABLESPACE users
  DB_FILE_NAME_CONVERT ('/maindisk/oradata/users',
                        '/backups/users_ts');
```

If you run a RESTORE command, then by default RMAN restores a datafile or control file to its original location by copying an image copy backup to that location. Image copies are chosen over backup sets because of the extra overhead of reading through an entire backup set in search of files to be restored.

If you need to restore and recover a current datafile, and if you have an image copy available on disk, then you do not need to have RMAN copy the image copy back to its old location. Instead, you can use the image copy in place as a replacement for the datafile to be restored. "Performing Complete Recovery After Switching to a Copy" on page 17-16 explains how to perform this task.

See Also:

- "Configuring the Default Type for Backups: Backup Sets or Copies" on page 5-4 to learn how to make either backup sets or image copies the default type of RMAN backups
- "Specifying Backup Set or Copy for an RMAN Backup to Disk" on page 8-3
- *Oracle Database Backup and Recovery Reference* to learn about the meaning of %U for image copies

User-Managed Image Copies

RMAN can use image copies created by mechanisms outside of RMAN, such as native operating system file copy commands or third-party utilities that leave image copies of files on disk. This type of copy is known as a **user-managed backup** or **operating system backup**.

You can use the CATALOG command to inspect an existing image copy and enter its metadata into the RMAN repository. However, the CATALOG command does not do the following:

- Read all blocks in the datafile copy to insure there are no corruptions
- Guarantee that the image copy was correctly made in **backup mode**

After you catalog these files, you can use them with the RESTORE or SWITCH commands just as you can for RMAN-generated image copies.

Some sites store their datafiles on mirrored disk volumes, which permit the creation of image copies by **breaking a mirror**. After you have broken the mirror, you can notify RMAN of the existence of a new user-managed copy, thus making it eligible for a backup. You must notify RMAN when the copy is no longer available by using the CHANGE . . . UNCATALOG command.

See Also:

- Chapter 27, "Making User-Managed Database Backups"
- "Adding Backup Records to the RMAN Repository" on page 11-16 to learn how to catalog datafile and archived log image copies
- "Making Split Mirror Backups with RMAN" on page 9-8
- *Oracle Database Backup and Recovery Reference* for `CHANGE` syntax

Multiple Copies of RMAN Backups

In RMAN, you can make multiple, identical copies of backups in the following ways:

- Duplex backups with the `BACKUP ... COPIES` command, in which case RMAN creates more than one copy of each backup set
- Back up your files as backup sets or image copies, and then back up the backup sets or image copies with the RMAN `BACKUP BACKUPSET` or `BACKUP COPY OF` commands

Duplexed Backup Sets

When backing up datafiles, archived redo log files, server parameter files, and control files into backup pieces, RMAN can create a **duplexed backup set**, producing up to four identical copies of each backup piece in the backup set on different backup destinations with one `BACKUP` command. Duplexing is not supported for backup operations that produce image copies.

You can use the `COPIES` parameter in the `CONFIGURE`, `SET`, or `BACKUP` commands to specify duplexing of backup sets when using the `BACKUP` command. RMAN can duplex backups to either disk or tape, but cannot duplex backups to tape and disk simultaneously. When backing up to tape, ensure that the number of copies does not exceed the number of available tape devices.

The `FORMAT` parameter of the `BACKUP` command specifies the destinations for duplexed backups. The following example creates 3 copies of the backup of datafile `7`:

```
BACKUP DEVICE TYPE DISK COPIES 3 DATAFILE 7
  FORMAT '/disk1/%U','?/oradata/%U','?/%U';
```

RMAN places the first copy of each backup piece in `/disk1`, the second in `?/oradata`, and the third in the Oracle home. Note that RMAN does not produce three backup sets, each with a different unique backup set key. Rather, RMAN produces one backup set with a unique key, and generates three identical copies of each backup piece in the set.

See Also:

- "Configuring Backup Duplexing" on page 6-5
- "Duplexing Backup Sets" on page 9-6
- *Oracle Database Backup and Recovery Reference* for `CONFIGURE` syntax
- *Oracle Database Backup and Recovery Reference* for `SET` syntax

Backups of Backups

You can use the BACKUP command to back up existing backup sets and image copies.

Backups of Backup Sets

The RMAN BACKUP BACKUPSET command backs up backup sets that were created on disk. The command is a useful way to spread backups among multiple media.

If RMAN discovers that one copy of a backup set is corrupted or missing, then it searches for other copies of the same backup set. This behavior is similar to the behavior of RMAN when backing up archived redo logs that exist in multiple archiving destinations.

Example 7–2 shows how you might run the BACKUP command weekly as part of the production backup schedule. In this way, you ensure that all your backups exist on both disk and tape.

Example 7–2 Backing Up Backup Sets to Tape

```
BACKUP DEVICE TYPE DISK AS BACKUPSET
  DATABASE PLUS ARCHIVELOG;
BACKUP
  DEVICE TYPE sbt
  BACKUPSET ALL; # copies backup sets on disk to tape
```

> **Note:** Backups to sbt that use automatic channels require that you first run the CONFIGURE DEVICE TYPE sbt command.

You can also use BACKUP BACKUPSET to manage backup space allocation. Example 7–3 backs up backup sets that were created more than a week ago from disk to tape, and then deletes them from disk.

Example 7–3 Managing Space Allocation

```
BACKUP
  DEVICE TYPE sbt
  BACKUPSET COMPLETED BEFORE 'SYSDATE-7'
  DELETE INPUT;
```

Note that DELETE INPUT here is equivalent to DELETE ALL INPUT: RMAN deletes all existing copies of the backup set. If you duplexed a backup to four locations, then RMAN deletes all four copies of the pieces in the backup set.

> **See Also:** "Backing Up RMAN Backups" on page 8-26

Backups of Image Copies

You can use the BACKUP COPY OF command to back up existing image copies of database files either as backup sets or as image copies. When using this command, an image copy of every datafile specified in the command must already exist. If there are multiple copies of a datafile, then the latest one is used. If you specify a tablespace or the whole database, then RMAN issues an error if there are datafiles in the database or tablespace for which there are no image copy backups.

Control File and Server Parameter File Autobackups

Having recent backups of your control file and server parameter file is extremely valuable in many recovery situations. To increase the likelihood that you will have such backups, the database supports control file and server parameter file autobackups. The autobackup occurs independently of any backup of the current control file explicitly requested as part of your `BACKUP` command.

With a **control file autobackup**, RMAN can recover the database even if the current control file, recovery catalog, and server parameter file are inaccessible. Because the path used to store the autobackup follows a well-known format, RMAN can search for and restore the server parameter file from that autobackup. After you have started the instance with the restored server parameter file, RMAN can restore the control file from the autobackup. After you mount the control file, use the RMAN repository in the mounted control file to restore the datafiles.

When RMAN Performs Control File Autobackups

If `CONFIGURE CONTROLFILE AUTOBACKUP` is `ON`, then RMAN automatically backs up the control file and the current server parameter file (if used to start up the database) at the end of a successful `BACKUP` command. If the database runs in `ARCHIVELOG` mode, RMAN makes control file autobackups when a structural change to the database affects the contents of the control file.

How RMAN Performs Control File Autobackups

The first channel allocated during the backup job creates the autobackup and places it into its own backup set. For autobackups after database structural changes, the server process associated with the structural change makes the backup.

If a server parameter file is in use by the database, then RMAN backs it up in the same backup set as the control file autobackup. After the autobackup completes, the database writes a message containing the complete path of the backup piece and the device type to the alert log located in the **Automatic Diagnostic Repository (ADR)**.

Note: Control file autobackups are never duplexed.

The control file autobackup filename has a default format of `%F` for all device types, so that RMAN can determine the file location and restore it without a repository. You can specify a different format with the `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT` command, but all autobackup formats must include the `%F` variable. If you do not use the default format, then during disaster recovery you must specify the format that was used to generate the autobackups. Otherwise, RMAN cannot restore the autobackup.

See Also:

- "Configuring Control File and Server Parameter File Autobackups" on page 5-7
- *Oracle Database Backup and Recovery Reference* for `CONFIGURE` syntax
- *Oracle Database Backup and Recovery Reference* for `BACKUP` syntax
- *Oracle Database Backup and Recovery Reference* to learn about the substitution variable `%F`

Incremental Backups

By default, RMAN makes full backups. A **full backup** of a datafile includes every allocated block in the file being backed up. A full backup of a datafile can be an image copy, in which case every data block is backed up. It can also be stored in a backup set, in which case datafile blocks not in use may be skipped.

A full backup is the default type of RMAN backup. A full backup has no effect on subsequent incremental backups and is not considered a part of an **incremental backup** strategy. Image copies are always full backups because they include every data block in a datafile. A backup set is by default a full backup because it can potentially include every data block in a datafile, although **unused block compression** means that blocks never used are excluded and, in some cases, currently unused blocks are excluded (see "Compression for Backup Sets" on page 7-4).

In contrast to a full backup, an incremental backup copies only those data blocks that have changed since a previous backup. You can use RMAN to create incremental backups of datafiles, tablespaces, or the whole database. A full backup cannot be part of an incremental backup strategy; that is, it cannot be the parent for a subsequent incremental backup.

Multilevel Incremental Backups

RMAN can create **multilevel incremental backups**. Each incremental level is denoted by a value of 0 or 1. A **level 0 incremental backup**, which is the base for subsequent incremental backups, copies all blocks containing data. You can create a level 0 database backup as backup sets or image copies.

The only difference between a level 0 incremental backup and a full backup is that a full backup is never included in an incremental strategy. Thus, a full backup is the parent of incremental backups whose level is greater than 0.

A level 1 incremental backup can be either of the following types:

- A **differential incremental backup**, which backs up all blocks changed after the most recent incremental backup at level 1 or 0
- A **cumulative incremental backup**, which backs up all blocks changed after the most recent incremental backup at level 0

Incremental backups are differential by default.

> **Note:** Cumulative backups are preferable to differential backups when recovery time is more important than disk space, because fewer incremental backups need to be applied during recovery.

The size of the backup file depends solely upon the number of blocks modified, the incremental backup level, and the type of incremental (differential or cumulative).

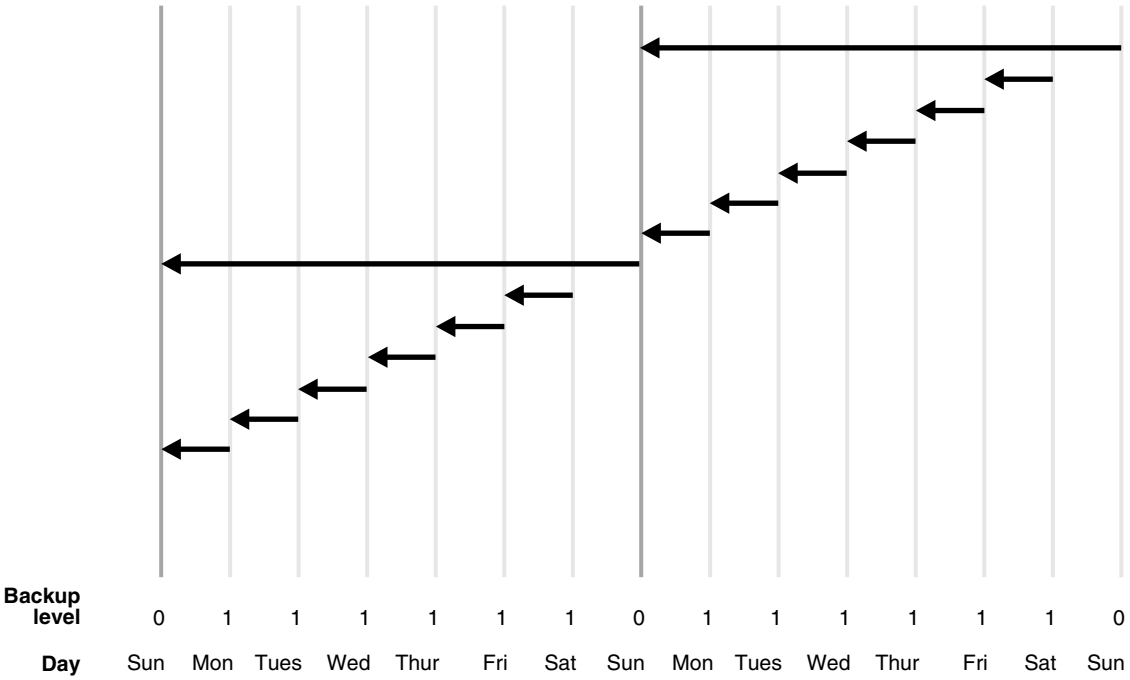
Differential Incremental Backups

In a differential level 1 backup, RMAN backs up all blocks that have changed since the most recent incremental backup at level 1 (cumulative or differential) or level 0. For example, in a differential level 1 backup, RMAN determines which level 1 backup occurred most recently and backs up all blocks modified after that backup. If no level 1 is available, then RMAN copies all blocks changed since the base level 0 backup.

If no level 0 backup is available in either the current or parent **incarnation**, then the behavior varies with the compatibility mode setting. If compatibility is >=10.0.0,

RMAN copies all blocks that have been changed since the file was created. Otherwise, RMAN behaves as it did in previous releases, by generating a level 0 backup.

Figure 7–2 Differential Incremental Backups



In the example shown in Figure 7–2, the following occurs each week:

- Sunday

  An incremental level 0 backup backs up *all* blocks that have ever been in use in this database.

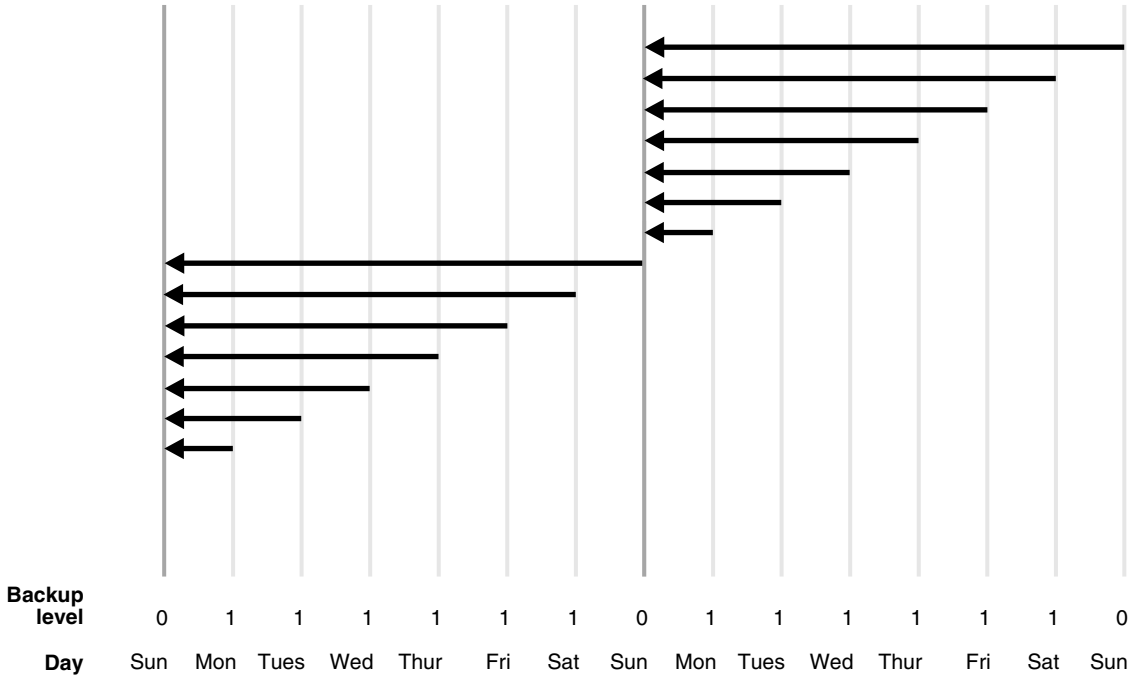
- Monday - Saturday

  On each day from Monday through Saturday, a differential incremental level 1 backup backs up all blocks that have changed since the most recent incremental backup at level 1 or 0. The Monday backup copies blocks changed since Sunday level 0 backup, the Tuesday backup copies blocks changed since the Monday level 1 backup, and so forth.

Cumulative Incremental Backups

In a cumulative level 1 backup, RMAN backs up all blocks used since the most recent level 0 incremental backup in either the current or parent incarnation. Cumulative incremental backups reduce the work needed for a restore by ensuring that you only need one incremental backup from any particular level. Cumulative backups require more space and time than differential backups because they duplicate the work done by previous backups at the same level.

Figure 7–3 Cumulative Incremental Backups



In the example shown in Figure 7–3, the following occurs each week:

- Sunday

  An incremental level 0 backup backs up *all* blocks that have ever been in use in this database.

- Monday - Saturday

  A cumulative incremental level 1 backup copies all blocks changed since the most recent level 0 backup. Because the most recent level 0 backup was created on Sunday, the level 1 backup on each day Monday through Saturday backs up all blocks changed since the Sunday backup.

  See Also: "Making and Updating Incremental Backups" on page 8-14

Block Change Tracking

The **block change tracking** feature for incremental backups improves incremental backup performance by recording changed blocks in each datafile in a **block change tracking file**. This file is a small binary file stored in the **database area**. RMAN tracks changed blocks as redo is generated.

If you enable block change tracking, then RMAN uses the change tracking file to identify changed blocks for an incremental backup, thus avoiding the need to scan every block in the datafile. RMAN only uses block change tracking when the incremental level is greater than 0 because a **level 0 incremental backup** includes all blocks.

See Also: "Using Block Change Tracking to Improve Incremental Backup Performance" on page 8-20

Incremental Backup Algorithm

The following concepts are essential for understanding the algorithm that RMAN uses to make incremental backups:

- Checkpoint SCN

  Every datafile has a **datafile checkpoint** SCN, which you can view in `V$DATAFILE.CHECKPOINT_CHANGE#`. All changes with an SCN lower than this SCN are guaranteed to be in the file. When a level 0 incremental backup is restored, the restored datafile contains the checkpoint SCN that it had when the level 0 was created. When a level 1 incremental is applied to a file, the checkpoint SCN of the file is advanced to the checkpoint SCN that the file had when the incremental level 1 was created.

- Incremental start SCN

  This SCN applies only to level 1 incremental backups. All blocks whose SCN is greater than or equal to the incremental start SCN are included in the backup. Blocks whose SCN is lower than the incremental start SCN are not included in the backup. The incremental start SCN is most often the checkpoint SCN of the parent of the level 1 backup.

- Block SCN

  Every data block in a datafile records the SCN at which the most recent change was made to the block.

When RMAN makes a level 1 incremental backup of a file, RMAN reads the file, examines the SCN of every block, and backs up blocks whose SCN is greater than or equal to the incremental start SCN for this backup. If the backup is differential, then the incremental start SCN is the checkpoint SCN of the most recent level 1 backup. If the backup is cumulative, then the incremental start SCN is the checkpoint SCN of the most recent level 0 backup.

When block change tracking is enabled, RMAN uses bitmaps to avoid reading blocks that have not changed during the range from incremental start SCN to checkpoint SCN. Note that RMAN still examines every block that is read and uses the SCN in the block to decide which blocks to include in the backup.

One consequence of the incremental backup algorithm is that RMAN applies all blocks containing changed data during recovery, even if the change is to an object created with the `NOLOGGING` option. Thus, if you restore a backup made before `NOLOGGING` changes were made, then incremental backups are the only way to recover them.

> **See Also:** *Oracle Database Concepts* for more information about `NOLOGGING` mode

Recovery with Incremental Backups

During **media recovery**, RMAN examines the restored files to determine whether it can recover them with an incremental backup. If it has a choice, then RMAN always chooses incremental backups over archived redo logs because applying changes at a block level is faster than applying redo.

RMAN does not need to restore a base incremental backup of a datafile to apply incremental backups to the datafile during recovery. For example, you can restore datafile image copies and recover them with incremental backups.

> **See Also:** "Selection of Incremental Backups and Archived Redo Logs" on page 13-5

Backup Retention Policies

You can use the `CONFIGURE RETENTION POLICY` command to create a persistent and automatic **backup retention policy**. When a backup retention policy is in effect, RMAN considers a backup of datafiles or control files as an **obsolete backup**, that is, no longer needed for recovery, according to criteria specified in the `CONFIGURE` command. You can use the `REPORT OBSOLETE` command to view obsolete files and the `DELETE OBSOLETE` command to delete them.

As you produce backups over time, older backups become obsolete as they are no longer needed to satisfy the retention policy. RMAN can identify the obsolete files for you, but it does not automatically delete them. You must use the `DELETE OBSOLETE` command to delete files that are no longer needed to satisfy the retention policy.

If a flash recovery area is configured, then the database automatically deletes files that are either obsolete or already backed up to tape when more recovery area space is needed for new files. The disk quota rules are distinct from the retention policy rules, but the database will never delete files in violation of the retention policy to satisfy the disk quota. Refer to "Responding to a Full Flash Recovery Area" on page 11-8.

A backup is obsolete when `REPORT OBSOLETE` or `DELETE OBSOLETE` determines, based on the user-defined retention policy, that it is not needed for recovery. A backup is considered an **expired backup** only when RMAN performs a crosscheck and cannot find the file. In short, *obsolete* means a file is not needed, whereas *expired* means it is not found.

A backup retention policy applies only to full or level 0 datafile and control file backups. For datafile copies and proxy copies, if RMAN determines that the copy or proxy copy is not needed, then the copy or proxy copy can be deleted. For datafile backup sets, RMAN cannot delete the backup set until all datafile backups within the backup set are obsolete.

The retention policy does not directly affect archived redo logs and incremental level 1 backups. Rather, these files become obsolete when no full backups exist that need them. Besides affecting full or level 0 datafile and control file backups, the retention policy affects archived redo logs and level 1 incremental backups. First, RMAN decides which datafile and control file backups are obsolete. Then, RMAN considers as obsolete all archived logs and incremental level 1 backups that are not needed to recover the oldest datafile or control file backup that must be retained.

> **Note:** RMAN cannot implement an automatic retention policy if backups are deleted by non-RMAN techniques, for example, through the media manager's tape retention policy. The media manager should never expire a tape until all RMAN backups on that tape have been removed from the media manager's catalog.

There are two mutually exclusive options for implementing a retention policy: **redundancy** and **recovery window**.

Recovery Window

A recovery window is a period of time that begins with the current time and extends backward in time to the **point of recoverability**. The point of recoverability is the earliest time for a hypothetical point-in-time recovery, that is, the earliest point to which you can recover following a media failure. For example, if you implement a recovery window of one week, then RMAN retains full backups and required

incremental backups and archived logs so that the database can be recovered up to 7 days in the past. You implement this retention policy as follows:

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
```

This command ensures that for each datafile one backup that is older than the point of recoverability must be retained. For example, if the recovery window is `7`, then there must always exist one backup of each datafile that satisfies the following condition:

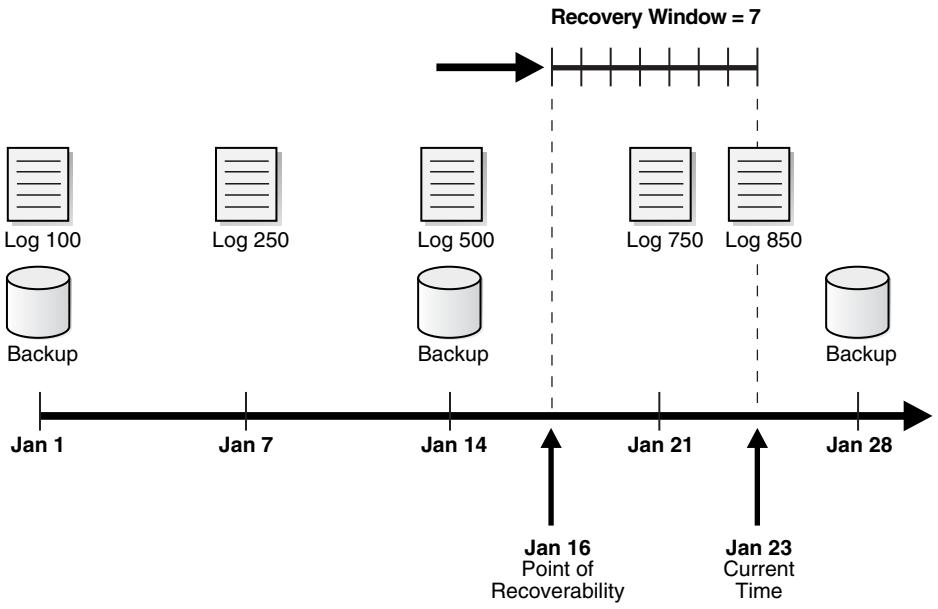
```
SYSDATE - BACKUP CHECKPOINT TIME >= 7
```

All backups older than the most recent backup that satisfied this condition are obsolete.

Assume the following retention policy illustrated in Figure 7–4. The retention policy has the following aspects:

- The recovery window is 7 days.
- Database backups are scheduled every two weeks on these days:
  - January 1
  - January 15
  - January 29
  - February 12
- The database runs in `ARCHIVELOG` mode, and archived logs are saved on disk only as long as needed for the retention policy.

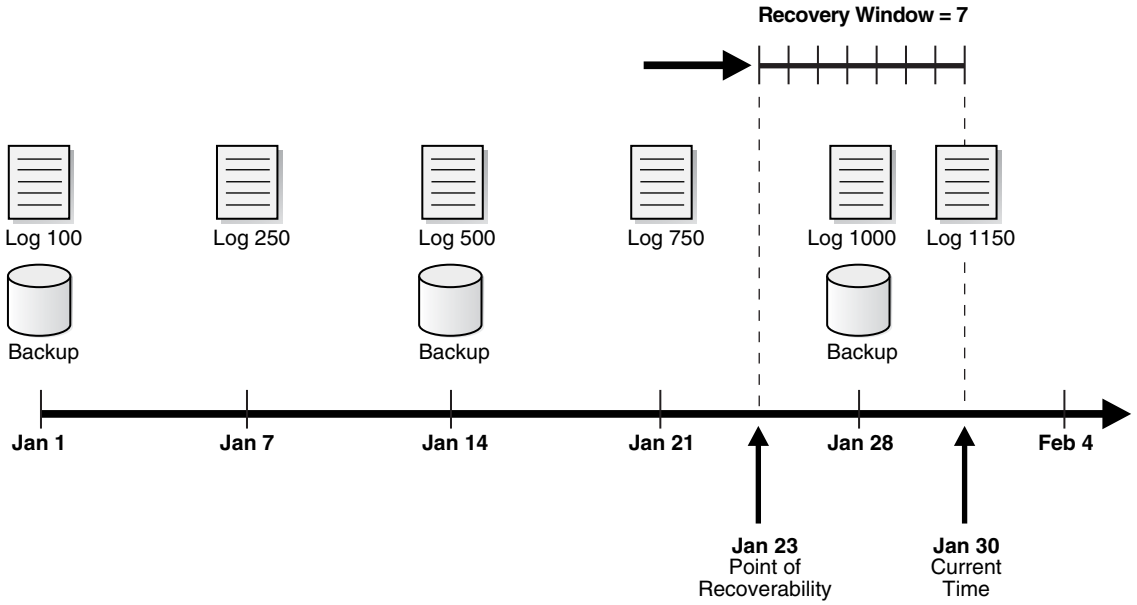
Figure 7–4 Recovery Window, Part 1



As illustrated in Figure 7–4, the current time is January 23 and the point of recoverability is January 16. Hence, the January 14 backup is needed for recovery, and so are the archived logs from log sequence 500 through 850. The logs before 500 and the January 1 backup are obsolete because they are not needed for recovery to a point within the window.

Assume the same scenario a week later, as depicted in Figure 7–5.

Figure 7–5 Recovery Window, Part 2



In this scenario, the current time is January 30 and the point of recoverability is January 23. Note how the January 14 backup is *not* obsolete even though a more recent backup (January 28) exists in the recovery window. This situation occurs because restoring the January 28 backup does not enable you to recover to the earliest time in the window, January 23. To ensure recoverability to any point in the window, you must save the January 14 backup and all archived logs from sequence 500 to 1150.

> **See Also:** "Configuring a Recovery Window-Based Retention Policy" on page 5-22

Backup Redundancy

In some cases using a recovery window can complicate disk space planning because the number of backups that must be retained is not constant and depends on the backup schedule. In contrast, a redundancy-based retention policy specifies how many backups of each datafile must be retained. For example, you can configure a redundancy of 2 as follows:

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
```

The default retention policy is configured to `REDUNDANCY 1`.

> **See Also:** "Configuring a Redundancy-Based Retention Policy" on page 5-21

Batch Deletes of Obsolete Backups

You can run the `REPORT OBSOLETE` command to determine which backups are currently obsolete according to the retention policy. A companion command, `DELETE OBSOLETE`, deletes all files which are obsolete according to the retention policy. You can run `DELETE OBSOLETE` periodically to minimize space wasted by storing obsolete backups. For example, you can run `DELETE OBSOLETE` in a weekly script.

You can also override the configured retention policy the configured retention policy by specifying the `REDUNDANCY` or `RECOVERY WINDOW` options on the `REPORT` or

DELETE commands. Note that using DELETE OBSOLETE with a recovery window shorter than the configured recovery window effectively reduces the window of recoverability. For example, if the configured window is 14 days, but you execute DELETE OBSOLETE RECOVERY WINDOW OF 7 DAYS, then you will no longer be able to recover to a time between 7 and 14 days ago.

See Also:

- Chapter 10, "Reporting on RMAN Operations"to generate reports and delete backups
- *Oracle Database Backup and Recovery Reference* for DELETE syntax
- *Oracle Database Backup and Recovery Reference* for REPORT syntax

Backup Retention Policy and Flash Recovery Area Deletion Rules

The RMAN status OBSOLETE is always determined in reference to a retention policy. For example, if a database backup is considered OBSOLETE in the RMAN repository, it is because it is either not needed for recovery to a point within the recovery window, or it is redundant.

If you configure a **flash recovery area**, then the database uses an internal algorithm to select files in the flash recovery area that are no longer needed to meet the configured retention policy. These backups have status OBSOLETE and are eligible for deletion to satisfy the **disk quota** rules. The retention policy is never violated when determining which files to delete from the flash recovery area to satisfy the disk quota rules.

There is one important difference between the flash recovery area rules for OBSOLETE status and the disk quota rules for deletion eligibility. Assume that archived logs 1000 through 2000, which are on disk, are needed for the current recovery window and so are not obsolete. If you back up these logs to tape, then the retention policy considers the disk logs as required, that is, not obsolete. Nevertheless, the flash recovery area disk quota algorithm considers the logs on disk as eligible for deletion because they have been backed up to tape. The logs on disk do not have OBSOLETE status in the repository, but they are eligible for deletion by the flash recovery area.

8

Backing Up the Database

This chapter explains how to perform the most basic backup tasks and implement backup strategies using RMAN. This chapter includes the following topics:

- Overview of RMAN Backups
- Specifying Backup Output Options
- Backing Up Database Files with RMAN
- Backing Up Archived Redo Logs with RMAN
- Making and Updating Incremental Backups
- Making Database Backups for Long-Term Storage
- Backing Up RMAN Backups

See Also:

- Chapter 9, "Backing Up the Database: Advanced Topics" to learn about more advanced topics such as backup optimization, duplexing, backup encryption, and restartable backups
- *Oracle Data Guard Concepts and Administration* to learn how to perform RMAN backup and recovery in a Data Guard environment

Overview of RMAN Backups

This section provides an overview of RMAN backups.

Purpose of RMAN Backups

The primary purpose of RMAN backups is to protect your data. If a **media failure** or disaster occurs, then you can restore your backups and recover lost changes.

You can also make backups to preserve data for long-time archival, as explained in "Making Database Backups for Long-Term Storage" on page 8-23, and to transfer data, as explained in the chapters included in Part VII, "Transferring Data with RMAN".

Basic Concepts of RMAN Backups

As explained in Chapter 7, "RMAN Backup Concepts", you can back up all or part of your database with the `BACKUP` command from within the RMAN client. Many of the techniques described in this chapter are also used in the Oracle-suggested backup strategy provided by Enterprise Manager and described in *Oracle Database 2 Day DBA*.

In many cases, after your database has been configured in accordance with your backup strategy, you can back up the database by entering the following command at the RMAN prompt:

```
RMAN> BACKUP DATABASE;
```

RMAN uses the configured settings, the records of previous backups, and the control file record of the database structure to determine an efficient set of steps for the backup. RMAN then carries out these steps.

As explained in "RMAN File Management in a Data Guard Environment" on page 3-8, you can run RMAN backups at any database in a Data Guard environment. Any backup of any database in the environment is usable for recovery of any other database as long as the backup is accessible. You can offload all backups of database files, including control file backups, to a **physical standby database** and thereby avoid consuming resources on the primary database.

See Also:

- *Oracle Database Backup and Recovery Reference* to learn about the BACKUP command
- *Oracle Data Guard Concepts and Administration* to learn how to back up a standby database with RMAN

Specifying Backup Output Options

If you specify only the minimum required options for an RMAN command such as BACKUP DATABASE, then RMAN determines the destination device, locations for backup output, and a backup tag automatically based on your configured environment and built-in RMAN defaults.

You can also provide arguments to BACKUP to override these defaults. The most typical options are described in the following sections:

- Specifying the Device Type for an RMAN Backup
- Specifying Backup Set or Copy for an RMAN Backup to Disk
- Specifying a Format for RMAN Backups
- Specifying Tags for an RMAN Backup
- Making Compressed Backups

See Also: Chapter 9, "Backing Up the Database: Advanced Topics" to learn about advanced backup options such as duplexing and restarting backups

Specifying the Device Type for an RMAN Backup

The BACKUP command takes a DEVICE TYPE clause that specifies whether to back up to disk or tape device. The following example illustrates a backup to disk.

Example 8–1 Specifying Device Type DISK

```
BACKUP DATABASE
  DEVICE TYPE DISK;
```

When you run BACKUP without a DEVICE TYPE clause, RMAN stores the backup on the configured default device (disk or **SBT**). You set the default device with the

CONFIGURE DEFAULT DEVICE TYPE command described in "Configuring the Default Device for Backups: Disk or SBT" on page 5-3.

Specifying Backup Set or Copy for an RMAN Backup to Disk

RMAN can create backups on disk as image copies or as backup sets. "Configuring the Default Type for Backups: Backup Sets or Copies" on page 5-4 explains how to configure the default disk device. You can override this default with the AS COPY or AS BACKUPSET clauses. To back up to disk as image copies, use BACKUP AS COPY.

Example 8–2 Making Image Copies

```
BACKUP AS COPY
  DEVICE TYPE DISK
  DATABASE;
```

To back up your data into backup sets, use the AS BACKUPSET clause. You can allow backup sets to be created on the configured default device, or direct them specifically to disk or tape, as shown in the following example.

Example 8–3 Making Backup Sets

```
BACKUP AS BACKUPSET
  DATABASE;

BACKUP AS BACKUPSET
  DEVICE TYPE DISK
  DATABASE;

BACKUP AS BACKUPSET
  DEVICE TYPE SBT
  DATABASE;
```

Specifying a Format for RMAN Backups

RMAN provides a range of options to name the files generated by the BACKUP command. RMAN uses the following set of rules to determine the format of the output files, which are listed in order of precedence:

1. If a FORMAT parameter is specified on the BACKUP command, then this setting controls the generated filename.

   For example, you can direct the output to a specific location, as shown in the following command:

   ```
   BACKUP DATABASE
     FORMAT "/disk1/backup_%U";  # specifies a location on the file system
   ```

   In this case, backups are stored with generated unique filenames with the prefix /disk1/backup_. Note that the %U substitution variable, used to generate a unique string at this point in the filename, is required.

   You can also use the FORMAT parameter to name an ASM disk group as backup destination, as shown in the following example:

   ```
   BACKUP DATABASE
     FORMAT '+dgroup1';  # specifies an ASM disk group
   ```

   No %U is required in this case because **Automatic Storage Management (ASM)** generates unique file names as needed. However, you can specify %U if desired.

> **Note:** If you specify FORMAT when a flash recovery area is enabled, then RMAN obeys the FORMAT setting. If no location is specified in the FORMAT clause, then RMAN creates the backup in a platform-specific location.

2. If a FORMAT setting is configured for the specific channel used for the backup, then this setting controls the generated filename.
3. If a FORMAT setting is configured for the device type used for the backup, then this setting controls the generated filename.
4. If a **flash recovery area** is enabled during a disk backup, and if FORMAT is not specified, then RMAN creates the backup with an automatically generated name in the flash recovery area.
5. If none of the other conditions in this list apply, then the default location and filename format of the backup are platform-specific.

> **See Also:** *Oracle Database Backup and Recovery Reference* to learn about the FORMAT clause, and the installation guides in the Oracle Database documentation library to learn about the default file locations for your platform

Specifying Multiple Formats for Disk Backups

Typically, you do not need to specify a format when backing up to tape because the default %U variable generates a unique filename for tape backups. When backing up to disk, however, you can specify a format to spread the backup across several drives for improved performance. In this case, allocate one DISK channel for each disk drive and specify the format string on the ALLOCATE CHANNEL command so that the filenames are on different disks. For example, issue the following command:

```
RUN
{
  ALLOCATE CHANNEL disk1 DEVICE TYPE DISK FORMAT '/disk1/%d_backups/%U';
  ALLOCATE CHANNEL disk2 DEVICE TYPE DISK FORMAT '/disk2/%d_backups/%U';
  ALLOCATE CHANNEL disk3 DEVICE TYPE DISK FORMAT '/disk3/%d_backups/%U';
  BACKUP AS COPY DATABASE;
}
```

You can distribute backups in this manner by default in the future, by configuring channels as follows:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT '/disk1/%d_backups/%U';
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT '/disk2/%d_backups/%U';
CONFIGURE CHANNEL 3 DEVICE TYPE DISK FORMAT '/disk3/%d_backups/%U';
BACKUP AS COPY DATABASE;
```

Specifying Tags for an RMAN Backup

RMAN attaches a character string called a **tag** to every backup it creates, as a way of identifying the backup. You can either accept the default tag or specify your own with the TAG parameter of the BACKUP command.

About Backup Tags

User-specified tags are a useful way to indicate the purpose or usage of different classes of backups or copies. You can tag backup sets, proxy copies, datafile copies, or control file copies. For example, you can tag datafile copies that you intend to use in a `SWITCH` command as `for_switch_only` and file copies that should be used only for a `RESTORE` command as `for_restore_only`.

Tags do not need to be unique, so multiple backup sets or image copies can have the same tag, for example, `weekly_backup`. Assume that you specify that a datafile should be restored from backups that have a specific tag. If more than one backup of the requested file has the desired tag, then RMAN restores the most recent backup that has the desired tag, within any constraints on the `RESTORE` command.

In practice, tags are often used to distinguish a series of backups created as part of a single strategy, such as an incremental backup strategy. For example, you might create a weekly incremental backups with a tag such as `BACKUP TAG weekly_incremental`. Many forms of the `BACKUP` command let you associate a tag with a backup, and many `RESTORE` and `RECOVER` commands let you specify a tag to restrict which backups to use in the `RESTORE` or `RECOVER` operation.

If you do not explicitly specify a tag with the `TAG` parameter of the `BACKUP` command, then RMAN implicitly creates a default tag for backups (except for control file autobackups). The format of the tag is `TAG`*`YYYYMMDD`*`T`*`HHMMSS`*, where *`YYYY`* is the year, *`MM`* is the month, *`DD`* is the day, *`HH`* is the hour (in 24-hour format), *`MM`* is the minutes, and *`SS`* is the seconds. For example, a backup of datafile `1` may get the tag `TAG20070208T133437`. The date and time refer to when RMAN started the backup, in the time zone of the instance performing the backup. If multiple backup sets are created by one `BACKUP` command, then each backup piece has the same default tag.

Tags are stored in uppercase, regardless of the case used when entering them. The maximum length of a backup tag is 30 bytes. Tags cannot use operating system environment variables or use special formats such as `%T` or `%D`.

> **See Also:** *Oracle Database Backup and Recovery Reference* for the default format description in `BACKUP . . . TAG`

Specifying Tags for Backup Sets and Image Copies

The characters in a tag must be limited to the characters that are legal in filenames on the target database file system. For example, **Automatic Storage Management (ASM)** does not support the use of the dash (`-`) in the filenames it uses internally, so a tag including a dash (such as `weekly-incr`) is not a legal tag name for backups in ASM disk groups.

When you tag a backup set, the tag is an attribute of each backup piece in a given copy of a backup set. If you create a **multiplexed backup set**, then each copy of the backup set gets the same tag. Example 8–4 creates one backup set with tag `MONDAYBKP`.

Example 8–4 Applying a Tag to a Backup Set

```
BACKUP AS BACKUPSET
  COPIES 1
  DATAFILE 7
  TAG mondaybkp;
```

When you specify a tag for image copies, the tag applies to each individual copy. Example 8–5 shows that copies of datafiles in tablespaces `users` and `tools` get the tag `MONDAYCPY`.

Example 8–5 Applying Tags to Image Copies

```
BACKUP AS COPY
  TABLESPACE users, tools
  TAG mondaycpy;
```

You can use `FROM TAG` to copy an image copy with a specific tag, and then use `TAG` to assign the output copy a different tag. Example 8–6 creates new copies of all image copies of the database that have the tag `full_cold_copy` and gives the new copies the tag `new_full_cold_copy`.

Example 8–6 Assigning Tags to Output Copies

```
BACKUP AS COPY
  COPY OF DATABASE
    FROM TAG full_cold_copy
  TAG new_full_cold_copy;
```

Making Compressed Backups

For any use of the `BACKUP` command that creates backup sets, you can take advantage of RMAN support for **binary compression** of backup sets. Specify the `AS COMPRESSED BACKUPSET` option to the `BACKUP` command. The resulting backup sets are compressed with an algorithm optimized for efficient compression of Oracle Database files. No extra uncompression steps are required during recovery.

> **Note:** If you are backing up to tape and your tape device performs its own compression, then you should not use both RMAN backup set compression and the media manager vendor's compression. In most instances you will get better results using the media manager's compression. See the discussion of tuning RMAN's tape backup performance in Chapter 21, "Tuning RMAN Performance".

Example 8–7 backs up the entire database and archived logs to the configured default backup destination (disk or tape), producing compressed backup sets.

Example 8–7 Making Compressed Backups

```
BACKUP
  AS COMPRESSED BACKUPSET
  DATABASE PLUS ARCHIVELOG;
```

Binary compression creates some performance overhead during backup and restore operations. Binary compression consumes CPU resources, so compressed backups should not be scheduled when CPU usage is already high. However, the following circumstances may warrant paying the performance penalty:

- You are using disk-based backups when disk space in your flash recovery area or other disk-based backup destination is limited.
- You are performing your backups to some device over a network when reduced network bandwidth is more important than CPU usage.
- You are using some archival backup media such as CD or DVD, where reducing backup sizes saves on media costs and archival storage.

See Also: The `AS COMPRESSED BACKUPSET` option of the `BACKUP` command in *Oracle Database Backup and Recovery Reference* to learn about performance when using binary compression of backup sets

Backing Up Database Files with RMAN

This section contains the following topics:

- Making Whole Database Backups with RMAN
- Backing Up Tablespaces and Datafiles with RMAN
- Backing Up Control Files with RMAN
- Backing Up Server Parameter Files with RMAN
- Backing Up a Database in NOARCHIVELOG Mode

Making Whole Database Backups with RMAN

You can perform a **whole database backup** with the database mounted or open. To perform a whole database backup, from the RMAN prompt, use the `BACKUP DATABASE` command.

You may want to exclude specified tablespaces from a whole database backup. As explained in "Configuring Tablespaces for Exclusion from Whole Database Backups" on page 6-6, you can persistently skip tablespaces across RMAN sessions by executing the `CONFIGURE EXCLUDE` command for each tablespace that you always want to skip. You can override the configured setting with `BACKUP ... NOEXCLUDE`.

To back up the database:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Make sure the database is mounted or open.
3. Issue the `BACKUP DATABASE` command at the RMAN prompt.

   The simplest form of the command requires no options or parameters, as shown in the following example:

   ```
   BACKUP DATABASE;
   ```

   The following example backs up the database, switches the online redo logs, and includes archived logs in the backup:

   ```
   BACKUP DATABASE PLUS ARCHIVELOG;
   ```

   By archiving the logs immediately after the backup, you ensure that you have a full set of archived logs through the time of the backup. In this way, you guarantee that you can perform media recovery after restoring this backup.

See Also:

- *Oracle Database Backup and Recovery Reference* to learn about the `BACKUP` command and *Oracle Database Backup and Recovery Reference* to learn about the `CONNECT` command
- "Skipping Offline, Read-Only, and Inaccessible Files" on page 9-6 to learn how to use `BACKUP ... SKIP` to skip inaccessible datafiles or datafiles that are in offline or read-only tablespaces
- "Configuring Tablespaces for Exclusion from Whole Database Backups" on page 6-6 to learn how to exclude specific tablespaces from whole database backups

Backing Up Tablespaces and Datafiles with RMAN

You can back up one or more tablespaces with the `BACKUP TABLESPACE` command or one or more datafiles with the `BACKUP DATAFILE` command. When you specify tablespaces, RMAN translates the tablespace name internally into a list of datafiles. The database can be mounted or open. Tablespaces can be read/write or read-only.

Note: Transportable tablespaces do not have to be in read/write mode for backup as in previous releases.

RMAN automatically backs up the control file and the server parameter file (if the instance was started with a server parameter file) when datafile 1 is included in the backup. If **control file autobackup** is enabled, then RMAN writes the current control file and server parameter file to a separate autobackup piece. Otherwise, RMAN includes these files in the backup set that contains datafile 1.

To back up tablespaces or datafiles:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. If the database instance is not started, then either mount or open the database.
3. Run the `BACKUP TABLESPACE` command or `BACKUP DATAFILE` command at the RMAN prompt.

   The following example backs up the `users` and `tools` tablespaces to tape:

   ```
   BACKUP
     DEVICE TYPE sbt
     TABLESPACE users, tools;
   ```

   The following example uses an SBT channel to back up datafiles `1` through `4` and a datafile copy stored at `/tmp/system01.dbf` to tape:

   ```
   BACKUP
     DEVICE TYPE sbt
     DATAFILE 1,2,3,4
     DATAFILECOPY '/tmp/system01.dbf';
   ```

Backing Up Control Files with RMAN

You can back up the control file when the database is mounted or open. RMAN uses a **snapshot control file** to ensure a read-consistent version. If the `CONFIGURE CONTROLFILE AUTOBACKUP` command is set to `ON` (by default it is `OFF`), then RMAN automatically backs up the control file and server parameter file after every backup

and after database structural changes. The **control file autobackup** contains metadata about the previous backup, which is crucial for **disaster recovery**.

> **Note:** You can restore a backup of a control file made on one Data Guard database to any other database in the environment. Primary and standby control file backups are interchangeable. See *Oracle Data Guard Concepts and Administration* to learn how to use RMAN to restore files on a standby database.

If the autobackup feature is not set, then you must manually back up the control file in one of the following ways:

- Run `BACKUP CURRENT CONTROLFILE`
- Include a backup of the control file within any backup by using the `INCLUDE CURRENT CONTROLFILE` option of the `BACKUP` command
- Back up datafile `1`, because RMAN automatically includes the control file and server parameter file in backups of datafile 1

> **Note:** If the control file block size is not the same as the block size for datafile 1, then the control file cannot be written into the same backup set as the datafile. RMAN writes the control file into a backup set by itself if the block size is different. The `V$CONTROLFILE.BLOCK_SIZE` column indicates the control file block size, whereas the `DB_BLOCK_SIZE` initialization parameter indicates the block size of datafile 1.

Making a Manual Backup of the Control File

A manual backup of the control file is not the same as a control file autobackup. RMAN makes a control file autobackup after the files specified in the `BACKUP` command are backed up. Thus, the autobackup—unlike a manual control file backup—contains metadata about the backup that just completed. Also, RMAN can automatically restore autobackups without the use of a recovery catalog.

To make a manual backup, you can either specify `INCLUDE CURRENT CONTROLFILE` when backing up other files or specify `BACKUP CONTROLFILE`. You can also back up control files copies on disk by specifying the `CONTROLFILECOPY` parameter.

To manually back up the control file:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Make sure the target database is mounted or open.
3. Execute the `BACKUP` command with the desired control file clause.

   The following example backs up tablespace `users` to tape and includes the current control file in the backup:

   ```
   BACKUP DEVICE TYPE sbt
     TABLESPACE users
     INCLUDE CURRENT CONTROLFILE;
   ```

   The following example backs up the current control file to the default disk device:

   ```
   BACKUP AS COPY
     CURRENT CONTROLFILE
     FORMAT '/tmp/control01.ctl';
   ```

The following example backs up the control file copy created in the previous example to tape:

```
BACKUP AS COPY
  CURRENT CONTROLFILE
  FORMAT '/tmp/control01.ctl';
BACKUP DEVICE TYPE sbt
  CONTROLFILECOPY '/tmp/control01.ctl';
```

If the control file autobackup feature is enabled, then RMAN makes two control file backups in these examples: the explicit backup of the files specified in the BACKUP command and the control file and server parameter file autobackup.

See Also: *Oracle Database Backup and Recovery Reference* to learn about the CONFIGURE CONTROLFILE AUTOBACKUP command

Backing Up Server Parameter Files with RMAN

As explained in "Backing Up Control Files with RMAN" on page 8-8, RMAN automatically backs up the current server parameter file in certain cases. The BACKUP SPFILE command backs up the parameter file explicitly. The server parameter file that is backed up is the one currently in use by the instance.

To back up the server parameter file:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Make sure the target database is mounted or open.

   The database must have been started with a server parameter file. If the instance is started with a client-side initialization parameter file, then RMAN issues an error if you execute BACKUP ... SPFILE.
3. Execute the BACKUP ... SPFILE command.

   The following example backs up the server parameter file to tape:

   ```
   BACKUP DEVICE TYPE sbt SPFILE;
   ```

Backing Up a Database in NOARCHIVELOG Mode

The only legal backup of a NOARCHIVELOG database is a closed, consistent backup. This script puts the database into the correct mode for a consistent, whole database backup and then backs up the database. The script assumes that **control file autobackup** is enabled for the database.

Example 8–8 Backing Up a Database in NOARCHIVELOG Mode

```
SHUTDOWN IMMEDIATE;
# Start up the database in case it suffered instance failure or was
# closed with SHUTDOWN ABORT before starting this script.
STARTUP FORCE DBA;
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
# this example uses automatic channels to make the backup
BACKUP
  INCREMENTAL LEVEL 0
  MAXSETSIZE 10M
  DATABASE
  TAG 'BACKUP_1';
```

```
# Now that the backup is complete, open the database.
ALTER DATABASE OPEN;
```

You can skip tablespaces, such as read-only tablespaces, but any skipped tablespace that has not been offline or read-only since its last backup is lost if the database has to be restored from a backup.

Backing Up Archived Redo Logs with RMAN

Archived redo logs are the key to successful media recovery. You should back them up regularly.

About Backups of Archived Redo Logs

Several features of RMAN backups are specific to archived redo logs. For example, you can use `BACKUP ... DELETE` to delete one or all copies of archived redo logs from disk after backing them up to backup sets.

Archived Redo Log Failover

Even if your redo logs are being archived to multiple destinations and you use RMAN to back up archived redo logs, RMAN selects only one copy of the archived redo log file to include in the backup set. Because logs with the same log sequence number are identical, RMAN does not need to include more than one log copy.

The **archived redo log failover** feature enables RMAN to complete a backup even when some archiving destinations are missing logs or contain logs with corrupt blocks. If at least one log corresponding to a given log sequence and thread is available in the flash recovery area or any of the archiving destinations, then RMAN tries to back it up. If RMAN finds a corrupt block in a log file during backup, it searches other destinations for a copy of that log without corrupt blocks.

For example, assume that you archive logs 121 through 124 to two destinations: `/arch1` and `/arch2`. Table 8–1 shows the archived redo log records in the control file.

Table 8–1 Sample Archived Redo Log Records

Sequence	Filename in /arch1	Filename in /arch2
121	`/arch1/archive1_121.arc`	`/arch2/archive1_121.arc`
122	`/arch1/archive1_122.arc`	`/arch2/archive1_122.arc`
123	`/arch1/archive1_123.arc`	`/arch2/archive1_123.arc`
124	`/arch1/archive1_124.arc`	`/arch2/archive1_124.arc`

However, unknown to RMAN, a user deletes logs 122 and 124 from the `/arch1` directory. Afterward, you run the following backup:

```
BACKUP ARCHIVELOG
  FROM  SEQUENCE 121
  UNTIL SEQUENCE 125;
```

With failover, RMAN completes the backup, using logs 122 and 124 in `/arch2`.

Online Redo Log Switching

Another important RMAN feature is automatic online redo log switching. To make an open database backup of archived redo logs that includes the most recent online redo log, you can execute the BACKUP command with any of the following clauses:

- PLUS ARCHIVELOG
- ARCHIVELOG ALL
- ARCHIVELOG FROM ...

Before beginning the backup, RMAN switches out of the current redo log group, and archives all online redo logs that have not yet been archived, up to and including the redo log group that was current when the command was issued. This feature ensures that the backup contains all redo generated before the start of the command.

One of the most effective ways of backing up archived redo logs is the BACKUP ... PLUS ARCHIVELOG clause, which causes RMAN to do the following:

1. Runs the ALTER SYSTEM ARCHIVE LOG CURRENT statement.
2. Runs BACKUP ARCHIVELOG ALL. If **backup optimization** is enabled, then RMAN skips logs that it has already backed up to the specified device.
3. Backs up the rest of the files specified in BACKUP command.
4. Runs the ALTER SYSTEM ARCHIVE LOG CURRENT statement.
5. Backs up any remaining archived logs generated during the backup. If backup optimization is not enabled, then RMAN backs up the logs generated in step 1 plus all the logs generated during the backup.

The preceding steps guarantee that datafile backups taken during the command are recoverable to a consistent state. Also, unless the online redo log is archived at the end of the backup, DUPLICATE is not possible with the backup.

Backing Up Archived Redo Log Files

To back up archived logs, use the BACKUP ARCHIVELOG command. Note that if **backup optimization** is enabled, then RMAN skips backups of archived logs that have already been backed up to the specified device.

To back up archived redo log files:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Make sure the target database is mounted or open.
3. Execute the BACKUP ARCHIVELOG or BACKUP ... PLUS ARCHIVELOG command.

   The following example backs up the database and all archived redo logs:

   ```
   BACKUP DATABASE PLUS ARCHIVELOG;
   ```

   The following example uses a configured disk or SBT channel to back up one copy of each log sequence number for all archived redo logs:

   ```
   BACKUP ARCHIVELOG ALL;
   ```

   You can also specify a range of archived redo logs by time, SCN, or log sequence number, as in the following example:

   ```
   BACKUP ARCHIVELOG
     FROM TIME  'SYSDATE-30'
   ```

```
UNTIL TIME 'SYSDATE-7';
```

Backing Up Only Archived Redo Logs That Need Backups

You can indicate that RMAN should *automatically* skip backups of archived redo logs in the following ways:

- Configuring **backup optimization**

  As explained in "Configuring Backup Optimization" on page 5-23, if you enable **backup optimization**, then the BACKUP ARCHIVELOG command skips backing up files when an identical archived log has already been backed up to the specified device type. An archived log is considered identical to another when it has the same DBID, thread, sequence number, and RESETLOGS SCN and time.

- Configure an **archived redo log deletion policy**

  As explained in "Configuring an Archived Redo Log Deletion Policy" on page 5-26, if the deletion policy is configured with the BACKED UP *integer* TIMES clause, then a BACKUP ARCHIVELOG command copies the logs unless *integer* backups already exist on the specified device type. If *integer* backups of the logs exist, then the BACKUP ARCHIVELOG command skips the logs.

The BACKUP ... NOT BACKED UP *integer* TIMES command specifies that RMAN should back up only those archived log files that have not been backed up at least *integer* times to the specified device. To determine the number of backups for a file, RMAN only considers backups created on the same device type as the current backup.

The BACKED UP clause is a convenient way to back up archived logs to a specified device type. For example, you can specify that RMAN should keep two copies of each archived redo log on tape and skip additional backups.

To back up archived redo logs that need backups:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Make sure the target database is mounted or open.
3. Make sure that appropriate channels are configured for the backup.
4. Execute the BACKUP ARCHIVELOG command with the NOT BACKED UP clause.

   ```
   BACKUP ARCHIVELOG ALL NOT BACKED UP 2 TIMES;
   ```

> **See Also:** "Using Backup Optimization to Skip Files" on page 9-3 for scenarios using NOT BACKED UP

Deleting Archived Redo Logs After Backups

The BACKUP ARCHIVELOG ... DELETE INPUT command deletes archived log files after they are backed up. This command eliminates the separate step of manually deleting archived redo logs.

With DELETE INPUT, RMAN deletes only the specific copy of the archived log chosen for the backup set. With DELETE ALL INPUT, RMAN deletes each backed-up archived redo log file from all log archiving destinations.

As explained in "Configuring an Archived Redo Log Deletion Policy" on page 5-26, the BACKUP ... DELETE INPUT and DELETE ARCHIVELOG commands obey the **archived redo log deletion policy** for logs in all archiving locations. For example, if you specify that logs should only be deleted when backed up at least twice to tape, then BACKUP ... DELETE honors this policy.

For the following procedure, assume that you archive to `/arc_dest1`, `/arc_dest2`, and the flash recovery area.

To delete archived redo logs after a backup:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Make sure that the target database is mounted or open.
3. Run the `BACKUP` command with the `DELETE INPUT` clause.

   Assume that you run the following `BACKUP` command:

   ```
   BACKUP DEVICE TYPE sbt
     ARCHIVELOG ALL
       DELETE ALL INPUT;
   ```

   In this case, RMAN backs up only one copy of each log sequence number in these archiving locations. RMAN does not delete any logs from the flash recovery area, but it deletes *all* copies of any log that it backed up from the other archiving destinations. The logs in the flash recovery area that are eligible for deletion will be automatically removed by the database when space is needed.

   If you had specified `DELETE INPUT` rather than `DELETE ALL INPUT`, then RMAN would only delete the specific archived redo log files that it backed up. For example, RMAN would delete the logs in `/arc_dest1` if these files were used as the source of the backup, but leave the contents of the `/arc_dest2` intact.

   See Also:

   - *Oracle Data Guard Concepts and Administration* to learn about archived redo log management with standby databases
   - *Oracle Database Backup and Recovery Reference* to learn about the `CONFIGURE ARCHIVELOG DELETION POLICY` command
   - *Oracle Database Backup and Recovery Reference* to learn about the `DELETE ARCHIVELOG` command
   - "Deleting RMAN Backups and Archived Redo Logs" on page 11-19

Making and Updating Incremental Backups

As explained in "Incremental Backups" on page 7-13, an **incremental backup** copies only datafile blocks that have changed since a specified previous backup. An incremental backups is either a **cumulative incremental backup** or **differential incremental backup**.

Although the content of the backups are the same, `BACKUP DATABASE` and `BACKUP INCREMENTAL LEVEL 0 DATABASE` are different. A **full backup** is not usable as part of an incremental strategy, whereas a **level 0 incremental backup** is the basis of an incremental strategy. No RMAN command can change a full backup into a level 0.

As with full backups, RMAN can make incremental backups of an `ARCHIVELOG` mode database that is open or mounted. If the database is in `NOARCHIVELOG` mode, then RMAN can only make incremental backups after a consistent shutdown.

Purpose of Incremental Backups

The primary reasons for making incremental backups part of your strategy are:

- Faster daily backups if **block change tracking** is enabled (see "Using Block Change Tracking to Improve Incremental Backup Performance" on page 8-20).
- Ability to roll forward datafile image copies, thereby reducing recovery time and avoiding repeated full backups.
- Less bandwidth consumption when backing up over a network
- Improved performance when the aggregate tape bandwidth for tape write I/Os is much less than the aggregate disk bandwidth for disk read I/Os.
- Possibility of recovering changes to objects created with the `NOLOGGING` option.

  For example, direct load inserts do not create redo log entries, so their changes cannot be reproduced with media recovery. Direct load inserts do change data blocks, however, and so these blocks are captured by incremental backups.
- Synchronize a physical standby database with the primary database. You can use the RMAN `BACKUP INCREMENTAL FROM SCN` command to create a backup on the primary database that starts at the current SCN of the standby, which you can then use to roll forward the standby database. See *Oracle Data Guard Concepts and Administration* to learn how to apply incremental backups to a standby database.

> **See Also:** *Oracle Database Administrator's Guide* for more information about `NOLOGGING` mode

Planning an Incremental Backup Strategy

Choose a backup strategy according to an acceptable **MTTR** (mean time to recover). For example, you can implement a three-level backup scheme so that a full or level 0 backup is taken monthly, a cumulative level 1 is taken weekly, and a differential level 1 is taken daily. In this strategy, you never have to apply more than a day of redo for complete recovery.

When deciding how often to take full or level 0 backups, a good rule of thumb is to take a new level 0 backup whenever 20% or more of the data has changed. If the rate of change to your database is predictable, then you can observe the size of your incremental backups to determine when a new level 0 is appropriate. The following SQL query determines the number of blocks written to an incremental level 1 backup of each datafile with at least 20% of its blocks backed up:

```
SELECT   FILE#, INCREMENTAL_LEVEL, COMPLETION_TIME,
         BLOCKS, DATAFILE_BLOCKS
FROM     V$BACKUP_DATAFILE
WHERE    INCREMENTAL_LEVEL > 0
AND      BLOCKS / DATAFILE_BLOCKS > .2
ORDER BY COMPLETION_TIME;
```

Compare the number of blocks in level 1 backups to a level 0 backup. For example, if you only create level 1 cumulative backups, then take a new level 0 backup when the most recent level 1 backup is about half of the size of the level 0 backup.

An effective way to conserve disk space is to make incremental backups to disk, and then offload the backups to tape with `BACKUP AS BACKUPSET`. Incremental backups are generally smaller than full backups, which limits the space required to store them until they are moved to tape. When the incremental backups on disk are backed up to tape, the tape is more likely to stream because all blocks of the incremental backup are copied to tape. There is no possibility of delay due to time required for RMAN to locate changed blocks in the datafiles.

Another strategy is to use incrementally updated backups as explained in "Incrementally Updating Backups" on page 8-17. In this strategy, you create an image copy of each datafile, then periodically roll forward this copy by making and then applying a level 1 incremental backup. In this way you avoid the overhead of making repeated full image copies of your datafiles, but enjoy all of the advantages.

In a Data Guard environment, you can offload incremental backups to a **physical standby database**. Incremental backups of a standby and primary database are interchangeable. Thus, you can apply an incremental backup of a standby database to a primary database, or apply an incremental backup of a primary database to a standby database. You can also enable block change tracking at a standby database, regardless of whether it is enabled at any other database in the Data Guard environment. In this way, RMAN automatically optimizes incremental backups of the standby database.

> **See Also:** *Oracle Data Guard Concepts and Administration* to learn how to back up a standby database with RMAN

Making Incremental Backups

After starting RMAN, run the `BACKUP INCREMENTAL` command at the RMAN prompt. By default incremental backups are differential.

To make an incremental backup:

1. Start RMAN and connect to a target database and a recovery catalog (if used).
2. Make sure the target database is mounted or open.
3. Execute the `BACKUP INCREMENTAL` command with the desired options.

   Use the `LEVEL` parameter to indicate the incremental level.The following example makes a level 0 incremental database backup.

   ```
   BACKUP
     INCREMENTAL LEVEL 0
     DATABASE;
   ```

   The following example makes a **differential incremental backup** at level 1 of the `SYSTEM` and `tools` tablespaces. It will only back up those data blocks changed since the most recent level 1 or level 0 backup.

   ```
   BACKUP
     INCREMENTAL LEVEL 1
     TABLESPACE SYSTEM, tools;
   ```

   The following example makes a **cumulative incremental backup** at level 1 of the tablespace `users`, backing up all blocks changed since the most recent level 0 backup.

   ```
   BACKUP
     INCREMENTAL LEVEL 1 CUMULATIVE
     TABLESPACE users;
   ```

Making Incremental Backups of a VSS Snapshot

You can use the **Volume Shadow Copy Service (VSS)** in conjunction with the Oracle VSS writer to make a **shadow copy** or snapshot of files in an Oracle Database. You must use a third-party backup program other than RMAN to make VSS snapshots with the Oracle VSS writer. In this case, the **flash recovery area** automates

management of files that are already backed up in a VSS snapshot and deletes them as needed.

You can use the `BACKUP INCREMENTAL LEVEL 1 ... FROM SCN` command in RMAN to create incremental backups in the flash recovery area. Thus, you can use this command to create an incremental level 1 backup of a VSS shadow copy. RMAN can apply incremental backups during recovery transparently.

> **See Also:** *Oracle Database Platform Guide for Microsoft Windows* to learn how to make VSS backups with RMAN

Incrementally Updating Backups

By incrementally updating backups, you can avoid the overhead of making full image copy backups of datafiles, while also minimizing time required for media recovery of your database. For example, if you run a daily backup script, then you never have more than one day of redo to apply for media recovery.

To incrementally update datafile backups:

1. Make a full image copy backup of a datafile.
2. At regular intervals (such as daily), make level 1 incremental backups of the datafile and apply them to the image copy backup.

This technique rolls forward the backup to the time when the level 1 incremental was made. RMAN can restore this **incrementally updated backup** and apply changes from the redo log. The result is the same as restoring a datafile backup taken at the SCN of the most recently applied incremental level 1 backup.

> **Note:** If you run the `RECOVER COPY` command daily, then a continuously updated image copy cannot satisfy a **recovery window** of more than one day. The incrementally updated backup feature is an optimization for fast media recovery.

Incrementally Updating Backups: Basic Example

To create incremental backups for use in an incrementally updated backups strategy, use the `BACKUP ... FOR RECOVER OF COPY WITH TAG` form of the `BACKUP` command. The command is best understood in the context of a sample script that implements the strategy.

This script in Example 8–9, run on a regular basis, is all that is required to implement a strategy based on incrementally updated backups.

Example 8–9 Basic Incremental Update Script

```
RUN
{
  RECOVER COPY OF DATABASE
    WITH TAG 'incr_update';
  BACKUP
    INCREMENTAL LEVEL 1
    FOR RECOVER OF COPY WITH TAG 'incr_update'
    DATABASE;
}
```

To understand the script and the strategy, it is necessary to understand the effects of these two commands when no datafile copies or incremental backups exist. Note two important features:

- The `BACKUP` command in Example 8–9 does not always create a level 1 incremental backup.
- The `RECOVER` command in Example 8–9 causes RMAN to apply any available incremental level 1 backups to a set of datafile copies with the specified tag.

The effect of the script is shown in the following table. The columns show the effects of the first run of the script, the second run of the script, and then the third and all subsequent run of the script.

Table 8–2 Effect of Script When Run Multiple Times

Command	Run 1	Run 2	Run 3 Onward
`RECOVER`	Because there is no incremental backup or datafile copy, the command generates a message (but not an error). That is, the command has no effect.	A datafile copy now exists, but no incremental level 1 backup exists with which to recover it. Thus, the `RECOVER` command has no effect.	A datafile copy and level 1 incremental exist from the previous run. The level 1 incremental is applied to the datafile copy, bringing the copy up to the checkpoint SCN of the level 1 incremental.
`BACKUP`	Because no level 0 image copy backup exists, the command creates an image copy backup of the datafile on disk with the specified tag. In other words, the script creates the image copy needed to begin the cycle of incremental updates. **Note:** If the script sets `DEVICE TYPE sbt`, then the first run creates the copy on disk, and not on tape. Subsequent runs make level 1 backups on tape.	The command makes an incremental level 1 backup containing block changes for the previous day.	The command makes an incremental level 1 backup containing block changes for the previous day.

Note also the following details about how Example 8–9 works:

- Each time a datafile is added to the database, an image copy of the new datafile is created the next time the script runs. The next run makes the first level 1 incremental for the added datafile. On all subsequent runs the new datafile is processed like any other datafile.
- You must use tags to identify the incremental level 0 datafile copies created for use in this strategy so that they do not interfere with other backup strategies you implement. If you use multiple incremental backup strategies, then RMAN cannot unambiguously create incremental level 1 backups unless you tag level 0 backups.

  The incremental level 1 backups to apply to those image copies are selected based upon the checkpoint SCNs of the image copy datafiles and the available incremental level 1 backups. The tag used on the image copy being recovered is not a factor in the selection of the incremental level backups.

In practice, you would schedule the script in Example 8–9 to run after each day, possibly at midnight. After the third run of the script, the following files would be available for a point-in-time recovery:

- An image copy of the database, as of the checkpoint SCN of the preceding run of the script, 24 hours earlier

- An incremental backup for the changes after the checkpoint SCN of preceding run
- Archived redo logs including all changes between the checkpoint SCN of the image copy and the current time

If at some point during the following 24 hours you need to restore and recover your database, then you can restore the datafiles from the incrementally updated datafile copies. You can then apply changes from the most recent incremental level 1 and the redo logs to reach the desired SCN. At most, you have 24 hours of redo to apply, which limits how long point-in-time recovery will take.

> **See Also:** *Oracle Database 2 Day DBA* to see how this technique is used in the Oracle-suggested backup strategy in Enterprise Manager

Incrementally Updated Backups: Advanced Example

You can extend the basic script in Example 8–9 to provide fast recoverability to a window greater than 24 hours. Example 8–10 shows how to maintain a window of seven days by specifying the beginning time of your window of recoverability in the `RECOVER` command.

Example 8–10 Advanced Incremental Update Script

```
RUN
{
  RECOVER COPY OF DATABASE
    WITH TAG 'incr_update'
    UNTIL TIME 'SYSDATE - 7';
  BACKUP
    INCREMENTAL LEVEL 1
    FOR RECOVER OF COPY WITH TAG 'incr_update'
    DATABASE;
}
```

The effect of the script is shown in the following table. The columns show the effects of the first run of the script, the second through seventh run of the scripts, and then each subsequent run of the script.

Table 8–3 Effect of Script When Run Multiple Times

	Run 1	Run 2-7	Run 8 Onward
`RECOVER` command	Does nothing because no backup exists to recover	Does nothing because `UNTIL TIME SYSDATE-7` specifies a time in the future	Applies the level 1 incremental backup made 7 days ago to the existing database copy
`BACKUP` command	Makes an incremental level 0 copy because no level 0 copy exists	Makes an incremental level 1 backup containing block changes for the previous day	Makes an incremental backup with the changes from the previous day

As with the basic script in Example 8–9, you have fast recoverability to any point in time between the SCN of the datafile copies and the present. RMAN can use both block changes from the incremental backups and individual changes from the redo logs. Because you have the daily level 1 incrementals, you still never need to apply more than one day of redo.

> **See Also:** *Oracle Database Backup and Recovery Reference* to learn about the `RECOVER` command

Using Block Change Tracking to Improve Incremental Backup Performance

The **block change tracking** feature for incremental backups improves backup performance by recording changed blocks for each datafile.

About Block Change Tracking

If block change tracking is enabled on a primary or standby database, then RMAN uses a **block change tracking file** to identify changed blocks for incremental backups. By reading this small bitmap file to determine which blocks changed, RMAN avoids having to scan every block in the datafile that it is backing up.

Block change tracking is disabled by default. Nevertheless, the benefits of avoiding full datafile scans during backup are considerable, especially if only a small percentage of data blocks are changed between backups. If your backup strategy involves incremental backups, then block change tracking is recommended. Block change tracking in no way changes the commands used to perform incremental backups. The change tracking files themselves generally require little maintenance after initial configuration.

Space Management in the Block Change Tracking File The change tracking file maintains bitmaps that mark changes in the datafiles between backups. The database performs a bitmap switch before each backup. Oracle Database automatically manages space in the change tracking file to retain block change data that covers the 8 most recent backups. After the maximum of 8 bitmaps is reached, the most recent bitmap is overwritten by the bitmap that tracks the current changes.

The first level 0 incremental backup scans the entire datafile. Subsequent incremental backups use the block change tracking file to scan only the blocks that have been marked as changed since the last backup. An incremental backup can be optimized only when it is based on a parent backup that was made after the start of the oldest bitmap in the block change tracking file.

Consider the 8-bitmap limit when developing your incremental backup strategy. For example, if you make a level 0 database backup followed by 7 differential incremental backups, then the block change tracking file now includes 8 bitmaps. If you then make a cumulative level 1 incremental backup, then RMAN cannot optimize the backup because the bitmap corresponding to the parent level 0 backup is overwritten with the bitmap that tracks the current changes.

Location of the Block Change Tracking File One block change tracking file is created for the whole database. By default, the change tracking file is created as an **Oracle-managed file** in the destination specified by the `DB_CREATE_FILE_DEST` initialization parameter. You can also specify the name of the block change tracking file by placing it in any location you choose. You can use a raw device (that is, a disk without a file system) as a block changing file, but be sure that the device is sufficiently large.

> **Note:** In an Oracle RAC environment, the change tracking file must be located on shared storage accessible from all nodes in the cluster.

RMAN does not support backup and recovery of the change tracking file. The database resets the change tracking file when it determines that the change tracking file is invalid. If you restore and recover the whole database or a subset, then the database clears the block change tracking file and starts tracking changes again. After you make a level 0 incremental backup, the next incremental backup is able to use change tracking data.

Size of the Block Change Tracking File The size of the block change tracking file is proportional to the size of the database and the number of enabled threads of redo. The size of the block change tracking file can increase and decrease as the database changes. The size is not related to the frequency of updates to the database.

Typically, the space required for block change tracking is approximately 1/30,000 the size of the data blocks to be tracked. The following factors that may cause the file to be larger than this estimate suggests:

- To avoid the overhead of allocating space as your database grows, the block change tracking file size starts at 10 MB. New space is allocated in 10 MB increments. Thus, for any database up to approximately 300 GB, the file size is no smaller than 10 MB, for up to approximately 600 GB the file size is no smaller than 20 MB, and so on.
- For each datafile, a minimum of 320 KB of space is allocated in the block change tracking file, regardless of the size of the datafile. Thus, if you have a large number of relatively small datafiles, the change tracking file is larger than for databases with a smaller number of larger datafiles containing the same data.

Enabling and Disabling Block Change Tracking

You can enable block change tracking when the database is either open or mounted. This section assumes that you intend to create the block change tracking file as an Oracle-managed file in the **database area**, which is where the database maintains active database files such as datafiles, control files, and online redo log files. See "Overview of the Flash Recovery Area" on page 5-14 to learn about the database area and **flash recovery area**.

To enable block change tracking:

1. Start SQL*Plus and connect to a target database with administrator privileges.
2. Make sure that the `DB_CREATE_FILE_DEST` initialization parameter is set.

   ```
   SHOW PARAMETER DB_CREATE_FILE_DEST
   ```

   If the parameter is not set, and if the database is open, then you can set the parameter with the following form of the `ALTER SYSTEM` statement:

   ```
   ALTER SYSTEM SET
     DB_CREATE_FILE_DEST = '/disk1/bct/'
     SCOPE=BOTH SID='*';
   ```

3. Enable block change tracking.

   Execute the following `ALTER DATABASE` statement:

   ```
   ALTER DATABASE ENABLE BLOCK CHANGE TRACKING;
   ```

   You can also create the change tracking file in a location you choose yourself by using the following form of SQL statement:

   ```
   ALTER DATABASE ENABLE BLOCK CHANGE TRACKING
     USING FILE '/mydir/rman_change_track.f' REUSE;
   ```

   The `REUSE` option tells Oracle Database to overwrite any existing block change tracking file with the specified name.

Disabling Block Change Tracking

This section assumes that the block change tracking feature is currently enabled. When you disable block change tracking, the database removes the block change tracking file from the operating system.

To disable block change tracking:

1. Start SQL*Plus and connect to a target database with administrator privileges.
2. Ensure that the target database is mounted or open.
3. Disable block change tracking.

   Execute the following `ALTER DATABASE` statement:

   ```
   ALTER DATABASE DISABLE BLOCK CHANGE TRACKING;
   ```

Checking Whether Change Tracking is Enabled

You can query the `V$BLOCK_CHANGE_TRACKING` view to determine whether change tracking is enabled, and if it is, the filename of the block change tracking file.

To determine whether change tracking is enabled:

- Enter the following query in SQL*Plus (sample output included):

```
COL STATUS   FORMAT A8
COL FILENAME FORMAT A60

SELECT STATUS, FILENAME
FROM   V$BLOCK_CHANGE_TRACKING;

STATUS   FILENAME
-------- ------------------------------------------------------------
ENABLED  /disk1/bct/RDBMS/changetracking/o1_mf_2f71np5j_.chg
```

Changing the Location of the Block Change Tracking File

To move the change tracking file, use the `ALTER DATABASE RENAME FILE` statement. The database must be mounted. The statement updates the control file to refer to the new location and preserves the contents of the change tracking file. If you cannot shut down the database, then you can disable and enable block change tracking. In this case, you lose the contents of the existing block change tracking file.

To change the location of the change tracking file:

1. Start SQL*Plus and connect to a target database.
2. If necessary, determine the current name of the change tracking file:

   ```
   SQL> SELECT FILENAME FROM V$BLOCK_CHANGE_TRACKING;
   ```

3. If possible, shut down the database. For example:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

   If you shut down the database, then skip to the next step. If you choose not to shut down the database, then execute the following SQL statements and skip all remaining steps:

   ```
   SQL> ALTER DATABASE DISABLE BLOCK CHANGE TRACKING;
   SQL> ALTER DATABASE ENABLE BLOCK CHANGE TRACKING USING FILE 'new_location';
   ```

In this case you lose the contents of the block change tracking file. Until the next time you complete a level 0 incremental backup, RMAN must scan the entire file.

4. Using host operating system commands, move the change tracking file to its new location.
5. Mount the database and move the change tracking file to a location that has more space. For example:

```
ALTER DATABASE RENAME FILE
   '/disk1/bct/RDBMS/changetracking/o1_mf_2f71np5j_.chg' TO
   '/disk2/bct/RDBMS/changetracking/o1_mf_2f71np5j_.chg';
```

This statement changes the location of the change tracking file while preserving its contents.

6. Open the database:

```
SQL> ALTER DATABASE OPEN;
```

See Also: *Oracle Database SQL Language Reference* to learn about the `ALTER DATABASE` statement and the `ALTER SYSTEM` statement

Making Database Backups for Long-Term Storage

This section explains the basic concepts and tasks involved in making backups for long-term storage.

Purpose of Archival Backups

You can use `BACKUP . . . KEEP` to create a backup that is both all-inclusive and exempt from the **backup retention policy**. The backup is all-inclusive because every file needed to restore and recover the database is backed up to a single disk or tape location. The `KEEP` option also specifies that the backup should be exempt from the retention policy either forever or for a specified period of time. The general name for a backup created with `BACKUP . . . KEEP` is an **archival backup**.

As explained in "Data Preservation" on page 1-3, one purpose of a backup and recovery strategy is to preserve data. You can use `BACKUP . . . KEEP` to retain a database backup for longer than the time dictated by the retention policy. For example, you can back up the database on the first day of every year to satisfy a regulatory requirement and store the media offsite. Years after you make the archival backup, you could restore and recover it to query the data as it appeared at the time of the backup.

Another purpose of an archival backup is to create a backup that you want to restore for testing purposes and then delete. For example, you can back up the database, restore the database in a test environment, and then discard the archival backup after the test database is operational. A related purpose is to create a self-contained backup that you can delete after transferring it to another user or host. For example, another user might want a copy of the database for reporting or testing.

Basic Concepts of Archival Backups

You can exempt a backup from the retention policy by using the `KEEP` option with the `BACKUP` command. You can also use the `KEEP` and `NOKEEP` options of the `CHANGE` command to change the status of an existing backup. Backups with `KEEP` attributes are valid backups that can be recovered like any other backups.

You can specify an end date for an archival backup with the `KEEP UNTIL TIME` clause, or specify that the backup should be kept `FOREVER`. If you specify `UNTIL`, then

RMAN marks the backup as obsolete when the UNTIL time has passed, regardless of any configured retention policy. For example, if you specify KEEP UNTIL TIME '01-JAN-08', then the backup is obsolete one second after midnight on January 1. If you specify an UNTIL TIME of 9:00 p.m, then the backup is obsolete at 9:01 p.m.

When you specify KEEP on the BACKUP command, RMAN generates multiple backup sets. Note the following characteristics of the BACKUP ... KEEP command:

- It automatically backs up the datafiles, control file (even if the **control file autobackup** is disabled), and the server parameter file.
- It automatically generates an archived redo log backup to ensure that the database backup can be recovered to a consistent state.
- If the FORMAT, POOL, or TAG parameters are specified, then they are used for all backups. For this reason, the FORMAT string must allow for the creation of multiple backup pieces. Specifying the %U substitution variable is the easiest way to meet this requirement.
- It supports an optional RESTORE POINT clause that creates a **normal restore point**, which is a label for an SCN to which the backup must be recovered to be made consistent. The SCN is captured just after the datafile backups complete. RMAN resynchronizes restore points with the **recovery catalog** and maintains the restore points as long as the backup exists. "Listing Restore Points" on page 10-9 explains how to display restore points.

See Also: *Oracle Database Backup and Recovery Reference* for CHANGE syntax and *Oracle Database Backup and Recovery Reference* for BACKUP ... KEEP syntax

Making an Archival Backup for Long-Term Storage

This section describes how to make a database backup for archival purposes. Typically, you make an archival backup to tape. Because your data protection backups will most likely be on a set of tapes that remain accessible and are recycled, it is advisable to earmark a set of tapes for the archival backup. You can write the archival backup to this special set of tapes and then place them in offsite storage.

This section explains the standard technique for creating an archival backup. You can vary the procedure to create a **stored script** or shell script that you can update dynamically. When you run the script, you can dynamically set the name of the restore point, backup format, and so on.

See Also:

- "Using DUPLICATE to Restore an Archival Backup" on page 23-21 to learn the recommended technique for restoring an archival backup
- "Using Substitution Variables in Command Files" on page 4-4 to learn how to make archival backups with RMAN command files
- "Creating and Executing Dynamic Stored Scripts" on page 12-18 to learn how to make archival backups with RMAN command files

Making an Archival Backup

This scenario makes a long-term archival backup with a backup **tag** of QUARTERLY and assigns it to a special family of Oracle Secure Backup tapes reserved for long-term storage. Note the following features of this example:

- The FOREVER keyword indicates that this backup is never eligible for deletion by the backup **retention policy**.
- The BACKUP command creates the restore point named FY06Q4 to match the SCN at which this backup will be consistent.

To make a long-term archival backup:

1. Start RMAN and connect to a target database and recovery catalog.

   The target database can be open or mounted. A recovery catalog is required for KEEP FOREVER, but is not required for any other KEEP option.

2. Run BACKUP ... KEEP to make the backup.

   The following example generates a datafile and archived log backup and creates a **normal restore point**. Note that the specified restore point must not already exist.

   The log backup contains just those archived logs needed to restore this backup to a consistent state. The database performs a online redo log switch to archive the redo that is in the current online logs and is necessary to make this new backup consistent. The control file autobackup has a copy of the restore point, so it can be referenced as soon as the control file is restored.

```
RUN
{
  ALLOCATE CHANNEL c1
    DEVICE TYPE sbt
    PARMS 'ENV=(OB_MEDIA_FAMILY=archival_backup)';
  BACKUP DATABASE
    TAG quarterly
    KEEP FOREVER
    RESTORE POINT FY06Q4;
}
```

   The following variation keeps the backup for 365 days instead of keeping it forever. After a year has passed, the backup becomes obsolete regardless of the backup retention policy settings.

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
    PARMS 'ENV=(OB_MEDIA_FAMILY=archival_backup)';
  BACKUP DATABASE
    TAG quarterly
    KEEP UNTIL TIME 'SYSDATE+365'
    RESTORE POINT FY06Q4;
}
```

   See Also: "About Restore Points and Flashback Database" on page 5-28 to learn about restore points

Making a Temporary Archival Backup

One purpose of an archival backup is for creating a test database. The technique for making a test database is essentially the same as the technique described in "Making an Archival Backup for Long-Term Storage" on page 8-24. The difference is that you intend to delete the backup soon after creating it.

You can specify the temporary status of the backup with the BACKUP ... KEEP UNTIL parameter. Assume that you want to make a backup and then restore it to a new host the same day. In this case, you can specify KEEP UNTIL TIME SYSDATE+1

to indicate that RMAN should override the retention policy for this backup for only one day. After one day, the backup becomes obsolete, regardless of any configured backup retention policy.

The command in Example 8–11 makes an archival backup on a temporary disk with the tag TESTDB. The example creates a normal restore point, which is a label for the time to which the backup should be recovered. RMAN only backs up the archived redo logs if the database is open during the backup. Archived logs are not needed for offline backups and so are not backed up.

Example 8–11 Creating a Temporary Archival Backup

```
BACKUP DATABASE
  FORMAT '/disk1/oraclebck/%U'
  TAG TESTDB
  KEEP UNTIL TIME 'SYSDATE+1'
  RESTORE POINT TESTDB06;
```

The recommended technique for restoring an archival backup is to use the DUPLICATE command. Refer to "Using DUPLICATE to Restore an Archival Backup" on page 23-21.

Backing Up RMAN Backups

This section explains how to back up backup sets and image copies.

About Backups of Backups

You can use the BACKUP BACKUPSET command to back up backup sets produced by other backup jobs. You can also use BACKUP RECOVERY AREA to back up recovery files created in the current and all previous **flash recovery area** destinations. Recovery files are full and incremental backup sets, control file autobackups, datafile copies, and archived redo logs. Only SBT backups are supported for BACKUP RECOVERY AREA.

The preceding commands are especially useful in the following scenarios:

- Ensuring that all backups exist both on disk and on tape.
- Moving backups from disk to tape and then freeing space on disk. This task is especially important when the database uses a flash recovery area so that the space can be reused as needed.

> **Note:** You cannot duplex backups when running BACKUP BACKUPSET. RMAN always makes one and only one copy on the specified media when performing BACKUP BACKUPSET.

You can also the BACKUP COPY OF command to back up image copies of datafiles, control files, and archived redo logs. The output of this command can be either backup sets or image copies, so you can generate backup sets from image copies. This form of backup is used to back up a database backup created as image copies on disk to tape.

Multiple Copies of Backup Sets

BACKUP BACKUPSET creates additional copies of backup pieces in a backup set, but does not create a new backup set. Thus, BACKUP BACKUPSET is similar to using the DUPLEX or MAXCOPIES option of BACKUP (see "Duplexing Backup Sets" on page 9-6). The extra copy of a backup set created by BACKUP BACKUPSET is not a new backup

set, just as copies of a backup set produced by other forms of the BACKUP command are not separate backup sets.

Effect of a Backup Retention Policy on Backups of Backups

For the purposes of a backup **retention policy** based on **redundancy**, a backup set is counted as one instance of a backup. This statement is true even if there are multiple copies of the backup pieces that make up backup set, such as when a backup set has been backed up from disk to tape.

For the purposes of a recovery window retention policy, either all of the copies of a backup set are obsolete, or none of them are. This point is easiest to grasp when viewing the output of the LIST and REPORT commands.

To view the effect of a backup retention policy on backups of backups:

1. Back up a datafile.

   The following example backs up datafile 5:

   ```
   BACKUP AS BACKUPSET DATAFILE 5;
   ```

2. Run the LIST command for the datafile backup in the preceding step.

   For example, run the following command (sample output included).

   ```
   LIST BACKUP OF DATAFILE 5 SUMMARY;

   List of Backups
   ===============
   Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
   ------- -- -- - ----------- --------------- ------- ------- ---------- ---
   18      B  F  A DISK        04-AUG-07       1       1       NO
   TAG20070804T160 134
   ```

3. Use the backup set key from the previous step to back up the backup set.

   For example, enter the following command:

   ```
   BACKUP BACKUPSET 18;
   ```

4. Run the same LIST command that you ran in step 2.

   For example, run the following command (sample output included).

   ```
   LIST BACKUP OF DATAFILE 5 SUMMARY;

   List of Backups
   ===============
   Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
   ------- -- -- - ----------- --------------- ------- ------- ---------- ---
   18      B  F  A DISK        04-AUG-07       1       2       NO
   TAG20070804T160 134
   ```

   Only one backup set is shown in this output, but there are now two copies of it.

5. Generate a report to see the effect of these copies under a redundancy-based backup retention policy.

   For example, issue the following command:

   ```
   REPORT OBSOLETE REDUNDANCY 1;
   ```

None of the copies is reported as obsolete because both copies of the backup set have the same values for `set_stamp` and `set_count`.

6. Generate a report to see the effect of these copies under a recovery window-based backup retention policy.

   For example, issue the following command:

   ```
   REPORT OBSOLETE RECOVERY WINDOW 1 DAY;
   ```

   None of the copies of the backup set is reported as obsolete or based on the `CHECKPOINT_CHANGE#` of this backup set, with respect to the current time and the availability of other backups.

See Also:

- "Configuring a Redundancy-Based Retention Policy" on page 5-21
- Chapter 10, "Reporting on RMAN Operations" to learn how to use the `LIST` and `REPORT` commands

Backing Up Backup Sets with RMAN

This section explains how to use the `BACKUP BACKUPSET` command to copy backup sets from disk to tape. The procedure assumes that you have already configured an **SBT** device as your default device.

To back up backup sets from disk to tape:

1. If you are backing up a subset of available backup sets, then execute the `LIST BACKUPSET` command to obtain their primary keys.

   The following example lists the backup sets in summary form:

   ```
   RMAN> LIST BACKUPSET SUMMARY;

   List of Backups
   ===============
   Key TY LV S Device Type Completion Time #Pieces #Copies Comp Tag
   --- -- -- - ----------- --------------- ------- ------- ---- ---
   1   B  F  A DISK        28-MAY-07       1       1       NO   TAG20070528T132432
   2   B  F  A DISK        29-MAY-07       1       1       NO   TAG20070529T132433
   3   B  F  A DISK        30-MAY-07       1       1       NO   TAG20070530T132434
   ```

   The following example lists details about backup set 3 in verbose form:

   ```
   RMAN> LIST BACKUPSET 3;

   List of Backup Sets
   ===================

   BS Key  Type LV Size       Device Type Elapsed Time Completion Time
   ------- ---- -- ---------- ----------- ------------ ---------------
   3       Full    8.33M      DISK        00:00:01     30-MAY-07
           BP Key: 3   Status: AVAILABLE  Compressed: NO  Tag: TAG20070530T132434
           Piece Name: /disk1/oracle/dbs/c-35764265-20070530-02
     Control File Included: Ckp SCN: 397221       Ckp time: 30-MAY-07
     SPFILE Included: Modification time: 30-MAY-07
     SPFILE db_unique_name: PROD
   ```

2. Execute the `BACKUP BACKUPSET` command.

The following example backs up all disk backup sets to tape and then deletes the input disk backups:

```
BACKUP BACKUPSET ALL
  DELETE INPUT;
```

The following example backs up only the backup sets with the primary key 1 and 2 to tape and then deletes the input disk backups:

```
BACKUP BACKUPSET 1,2
  DELETE INPUT;
```

3. Optionally, execute the `LIST` command to see a listing of backup sets and pieces.

   The output lists all copies, including backup piece copies created by `BACKUP BACKUPSET`.

Backing Up Image Copy Backups with RMAN

This section explains how to use the `BACKUP` command to back up image copies to tape. It is assumed that you have configured an **SBT** device as your default device.

Note that when backing up image copies that have multiple copies of the datafiles, specifying tags for the backups makes it easier to identify the input image copy. All image copies of datafiles have tags. The tag of an image copy is inherited by default when the image copy is backed up as a new image copy.

To back up image copies from disk to tape:

1. Issue the `BACKUP . . . COPY OF` or `BACKUP DATAFILECOPY` command.

   The following example backs up datafile copies that have the tag `DBCopy`:

   ```
   BACKUP DATAFILE COPY FROM TAG monDBCopy;
   ```

   The following example backs up the latest image copies of a database to tape, assigns the tag `QUARTERLY_BACKUP`, and deletes the input disk backups:

   ```
   BACKUP DEVICE TYPE sbt
     TAG "quarterly_backup"
     COPY OF DATABASE
     DELETE INPUT;
   ```

2. Optionally, issue the `LIST` command to see a listing of backup sets. The output lists all copies, including backup piece copies created by `BACKUP BACKUPSET`.

9

Backing Up the Database: Advanced Topics

This chapter explains advanced RMAN backup procedures. This chapter contains the following topics:

- Limiting the Size of RMAN Backup Sets
- Using Backup Optimization to Skip Files
- Skipping Offline, Read-Only, and Inaccessible Files
- Duplexing Backup Sets
- Making Split Mirror Backups with RMAN
- Encrypting RMAN Backups
- Restarting RMAN Backups
- Managing Backup Windows

> **See Also:** Chapter 8, "Backing Up the Database" for basic backup procedures

Limiting the Size of RMAN Backup Sets

As explained in "Configuring the Maximum Size of Backup Sets" on page 6-4, you can use the `CONFIGURE` command to create persistent settings that govern backup set size. This control is helpful when backing up very large files. If you do not have a backup set size persistently configured, then you can also use the `BACKUP ... MAXSETSIZE` command to limit the size of backup sets.

Note that you can use the `CONFIGURE` command, but not the `BACKUP` command, to set a limit for the size of individual backup pieces. This control is especially useful when you use a media manager that has restrictions on the sizes of files, or when you need to back up very large files. See "Configuring the Maximum Size of Backup Pieces" on page 6-4 for more information.

About Backup Set Size

The `MAXSETSIZE` parameter of the `BACKUP` command specifies a maximum size for a backup set in units of bytes (default), kilobytes, megabytes, or gigabytes. Thus, to limit a backup set to 305 MB, specify `MAXSETSIZE 305M`. RMAN attempts to limit all backup sets to this size.

You can use `BACKUP ... MAXSETSIZE` to limit the size of backup sets so that the database is divided among more than one backup set. If the backup fails partway through, then you can use the restartable backup feature to back up only those files

that were not backed up during the previous attempt. See "Restarting RMAN Backups" on page 9-12 to learn how to restart RMAN backups.

In some cases the MAXSETSIZE value may be too small to contain the largest file that you are backing up. When determining whether MAXSETSIZE is too small, RMAN uses the size of the original datafile rather than the file size after compression. RMAN displays an error stack such as the following:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of backup command at 11/03/06 14:40:33
RMAN-06182: archive log larger than MAXSETSIZE: thread 1 seq 1
            /oracle/oradata/trgt/arch/archive1_1.dbf
```

See Also: *Oracle Database Backup and Recovery Reference* to learn about the CONFIGURE MAXSETSIZE command

Limiting the Size of Backup Sets with BACKUP ... MAXSETSIZE

Backup piece size is an issue in those situations where it exceeds the maximum file size of the file system or media management software. Use the MAXSETSIZE parameter of the CONFIGURE CHANNEL or ALLOCATE CHANNEL command to limit the size of backup pieces.

To limit the size of backup sets:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Execute the BACKUP command with the MAXSETSIZE parameter.

   The following example backs up archived logs to tape, limiting the size of each backup set to 100 MB:

   ```
   BACKUP DEVICE TYPE sbt
     MAXSETSIZE 100M
     ARCHIVELOG ALL;
   ```

Dividing the Backup of a Large Datafile into Sections

If you specify the SECTION SIZE parameter on the BACKUP command, then RMAN creates a backup set in which each backup piece contains the blocks from one file section. A **file section** is a contiguous range of blocks in a file. This type of backup is called a **multisection backup**.

Note: You cannot specify SECTION SIZE in conjunction with MAXPIECESIZE.

The purpose of multisection backups is to enable RMAN channels to back up a single large file in parallel. RMAN divides the work among multiple channels, with each channel backing up one file section in a file. Backing up a file in separate sections can improve the performance of backups of large datafiles.

If a multisection backup completes successfully, then none of the backup sets generated during the backup contain a partial datafile. If a multisection backup is unsuccessful, then it is possible for the RMAN metadata to contain a record for a partial backup set. RMAN does not consider partial backups for restore and recovery. You must use the DELETE command to delete the partial backup set.

If you specify a section size that is larger than the size of the file, then RMAN does not use multisection backup for the file. If you specify a small section size that would produce more than 256 sections, then RMAN increases the section size to a value that results in exactly 256 sections.

To make a multisection backup:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. If necessary, configure channel parallelism so that RMAN can parallelize the backup.
3. Execute `BACKUP` with the `SECTION SIZE` parameter.

   For example, suppose that the `users` tablespace contains a single datafile of 900 MB. Also assume that three SBT channels are configured, with the parallelism setting for the SBT device set to 3. You can break up the datafile in this tablespace into file sections as shown in the following example:

   ```
   BACKUP
     SECTION SIZE 300M
     TABLESPACE users;
   ```

   In this example, each of the three SBT channels backs up a 300 MB file section of the `users` datafile.

   See Also: "Parallelizing the Validation of a Datafile" on page 15-5 to learn how to validate sections of a large datafile

Using Backup Optimization to Skip Files

As explained in "Configuring Backup Optimization" on page 5-23, you run the `CONFIGURE BACKUP OPTIMIZATION` command to enable **backup optimization**. When certain criteria are met, RMAN skips backups of files that are identical to files that are already backed up.

For the following scenarios, assume that you configure backup optimization and a retention policy as shown in the following example.

Example 9–1 Configuring Backup Optimization

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 4 DAYS;
```

With RMAN configured as shown in Example 9–1, you run the following command every night to back up the database to tape:

```
BACKUP DATABASE;
```

Because backup optimization is configured, RMAN skips backups of offline and read-only datafiles only if the most recent backups were made on or after the earliest point in the recovery window. RMAN does not skip backups when the most recent backups are older than the window. For example, optimization ensures you do not end up with a new backup of a read-only datafile every night, so long as one backup set containing this file exists within the recovery window.

See Also:

- "Backup Optimization for SBT Backups with Recovery Window Retention Policy" on page 5-24 for a scenario involving backup optimization and recovery windows
- *Oracle Database Backup and Recovery Reference* for a detailed description of criteria used by `CONFIGURE BACKUP OPTIMIZATION` to determine whether a file is identical and can potentially be skipped

Optimizing a Daily Archived Log Backup to a Single Tape: Scenario

Assume that you want to back up all the archived logs every night, but you do not want to have multiple copies of each log sequence number. With RMAN configured as shown in Example 9–1, you run the following command in a script nightly at 1 a.m.:

```
BACKUP DEVICE TYPE sbt
  ARCHIVELOG ALL;
```

RMAN skips all logs except those produced in the last 24 hours. In this way, you keep only one copy of each archived log on tape.

Optimizing a Daily Archived Log Backup to Multiple Media Families: Scenario

In Oracle Secure Backup, a **media family** is a named group of volumes with a set of shared, user-defined attributes. In this scenario, you back up logs that are not already on tape to one media family, then back up the same logs to a second media family. Finally, you delete old logs.

With RMAN configured as shown in Example 9–2, run the following script at the same time every night to back up the logs generated during the previous day to two separate media families.

Example 9–2 Backing Up Archived Redo Logs to Multiple Media Families

```
# The following command backs up just the logs that are not on tape. The
# first copies are saved to the tapes from the media family "log_family1".
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
    PARMS 'ENV=(OB_MEDIA_FAMILY=log_family1)';
  BACKUP ARCHIVELOG ALL;
}
# Make one more copy of the archived logs and save them to tapes from a
# different media family
RUN
{
  ALLOCATE CHANNEL c2 DEVICE TYPE sbt
    PARMS 'ENV=(OB_MEDIA_FAMILY=log_family2)';
  BACKUP ARCHIVELOG
    NOT BACKED UP 2 TIMES;
}
```

If your goal is to delete logs from disk that have been backed up two times to SBT, then the simplest way to achieve the goal is with an **archived redo log deletion policy**. The following one-time configuration specifies that archived redo logs are eligible for deletion from disk if two archived log backups exist on tape:

```
CONFIGURE ARCHIVELOG DELETION POLICY
```

```
  TO BACKED UP 2 TIMES TO DEVICE TYPE sbt;
```

After running the script in Example 9–2, you can delete unneeded logs by executing `DELETE ARCHIVELOG ALL`.

Creating a Weekly Secondary Backup of Archived Logs: Example

Assume a more sophisticated scenario in which your goal is to back up the archived logs to tape every day. You are worried about tape failure, however, so you want to ensure that you have more than copy of each log sequence number on an separate tape before you perform your weekly deletion of logs from disk. This scenario assumes that the database is not using a flash recovery area.

First, perform a one-time configuration as follows:

```
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE default DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS 'ENV=(OB_MEDIA_FAMILY=first_copy);
```

Because you have optimization enabled, you can run the following command every evening to back up all archived logs to the `first_copy` media family that have not already been backed up:

```
BACKUP ARCHIVELOG ALL TAG first_copy;
```

Every Friday evening you create an additional backup of all archived logs in a different media family. At the end of the backup, you want to delete all archived logs that already have at least two copies on tape. So you run the following script:

```
RUN
{
  # manually allocate a channel, in order to specify that the backup run by this
  # channel should go to both media families "first_copy" and "second_copy"
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
      PARMS 'ENV=(OB_MEDIA_FAMILY=second_copy)';
  ALLOCATE CHANNEL c2 DEVICE TYPE sbt
      PARMS 'ENV=(OB_MEDIA_FAMILY=first_copy)';
  BACKUP
    CHANNEL c1
    ARCHIVELOG
    UNTIL TIME 'SYSDATE'
    NOT BACKED UP 2 TIMES # back up only logs without 2 backups on tape
    TAG SECOND_COPY;
  BACKUP
    CHANNEL c2
    ARCHIVELOG
    UNTIL TIME 'SYSDATE'
    NOT BACKED UP 2 TIMES # back up only logs without 2 backups on tape
    TAG FIRST_COPY;
}

# now delete from disk all logs that have been backed up to tape at least twice
DELETE
  ARCHIVELOG ALL
  BACKED UP 2 TIMES TO DEVICE TYPE sbt;
```

The following table explains the effects of the daily and weekly backup scripts.

Table 9–1 Effects of Daily and Weekly Scripts

Script	Tape Contents After Script	Disk Contents After Script
Daily	Archived logs that have not yet been backed up are now in media family `first_copy`.	All archived logs created since the last `DELETE` command are still on disk.
Weekly	Archived logs that have fewer than two backups on tape are now in media families `first_copy` and `second_copy`.	All archived logs that have been backed up at least twice to tape are deleted.

After the weekly backup, you can send the tape from the media family `second_copy` to offsite storage. You should use this tape backup only if the primary tape from pool `first_copy` is damaged. Because the secondary tape is offsite, you do not want RMAN to use it for recovery, so you can mark the backup as unavailable:

```
CHANGE BACKUP OF ARCHIVELOG TAG SECOND_COPY UNAVAILABLE;
```

See Also:

- Chapter 11, "Maintaining RMAN Backups and Repository Records" to learn how to change the status of and delete backups
- *Oracle Database Backup and Recovery Reference* to learn about the `CHANGE` and `DELETE` commands

Skipping Offline, Read-Only, and Inaccessible Files

By default, the `BACKUP` command terminates when it cannot access a datafile. You can specify parameters to prevent termination, as listed in Table 9–2.

Table 9–2 BACKUP ... SKIP Options

If you specify . . .	Then RMAN skips . . .
`SKIP INACCESSIBLE`	Datafiles that RMAN cannot be read.
`SKIP OFFLINE`	Offline datafiles. Some offline datafiles can still be read because they exist on disk. Others have been deleted or moved and so cannot be read, making them inaccessible.
`SKIP READONLY`	Datafiles in read-only tablespaces.

The following example uses an automatic channel to back up the database, and skips all datafiles that might cause the backup job to terminate.

Example 9–3 Skipping Files During an RMAN Backup

```
BACKUP DATABASE
  SKIP INACCESSIBLE
  SKIP READONLY
  SKIP OFFLINE;
```

Duplexing Backup Sets

RMAN can make up to four copies of a backup set simultaneously, each an exact duplicate of the others. A copy of a **duplexed backup set** is a copy of each backup piece in the backup set, with each copy getting a unique copy number (for example, `0tcm8u2s_1_1` and `0tcm8u2s_1_2`). Note that it is not possible to duplex backup sets to the **flash recovery area**.

You can use BACKUP . . . COPIES or CONFIGURE . . . BACKUP COPIES to duplex backup sets. RMAN can duplex backups to either disk or tape, but cannot duplex backups to tape and disk simultaneously. For DISK channels, specify multiple values in the FORMAT option to direct the multiple copies to different physical disks. For **SBT** channels, if you use a media manager that supports Version 2 of the SBT API, then the media manager automatically writes each copy to a separate medium (for example, a separate tape). When backing up to tape, ensure that the number of copies does not exceed the number of available tape devices.

Duplexing applies only to backup sets, not image copies. It is an error to specify the BACKUP... COPIES when creating image copy backups, and the CONFIGURE... BACKUP COPIES setting is ignored for image copy backups.

> **See Also:** "Multiple Copies of RMAN Backups" on page 7-10 for a conceptual overview of RMAN backup copies

Duplexing Backup Sets with CONFIGURE BACKUP COPIES

As explained in "Configuring Backup Duplexing" on page 6-5, the CONFIGURE . . . BACKUP COPIES command specifies the number of identical backup sets that you want to create on the specified device type. This setting applies to all backup sets except control file autobackups (because the autobackup of a control file always produces one copy) and backup sets when backed up with the BACKUP BACKUPSET command.

To duplex a backup with CONFIGURE ... BACKUP COPIES**:**

1. Configure the number of copies on the desired device type for datafiles and archived redo logs on the desired device types.

   By default, CONFIGURE . . . BACKUP COPIES is set to 1 for each device type. The following example configures duplexing for datafiles and archived logs on tape and also duplexing for datafiles (but not archived redo logs) on disk:

   ```
   CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
   CONFIGURE DEFAULT DEVICE TYPE TO sbt;
   CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/disk1/%U', '/disk2/%U';
   CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE sbt TO 2;
   CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE sbt TO 2;
   CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 2;
   ```

2. Execute the BACKUP command.

   The following command backs up the database and archived logs to tape, making two copies of each datafile and archived log:

   ```
   BACKUP DATABASE PLUS ARCHIVELOG; # uses default sbt channel
   ```

   Because of the configured formats for the disk channel, the following command backs up the database to disk, placing one copy of the backup sets produced in the /disk1 directory and the other in the /disk2 directory:

   ```
   BACKUP DEVICE TYPE DISK AS COPY DATABASE;
   ```

   Note that if the FORMAT clause were not configured on CONFIGURE CHANNNEL, then you could specify FORMAT on the BACKUP command itself. For example, you could issue the following command:

   ```
   BACKUP DATABASE
     FORMAT '/disk1/%U',
            '/disk2/%U';
   ```

3. Issue a `LIST BACKUP` command to see a listing of backup sets and pieces.

   For example, enter the following command:

   ```
   LIST BACKUP SUMMARY;
   ```

   The `#Copies` column shows the number of backup sets, which may have been produced by duplexing or by multiple backup commands.

   > **See Also:** "Configuring Backup Duplexing" on page 6-5 to learn about the `CONFIGURE BACKUP COPIES` command, and "Configuring the Environment for RMAN Backups" on page 5-1 to learn about basic backup configuration options

Duplexing Backup Sets with BACKUP ... COPIES

The `COPIES` option of the `BACKUP` command overrides every other `COPIES` or `DUPLEX` setting to control duplexing of backup sets.

To duplex a backup with BACKUP ... COPIES:

1. Specify the number of identical copies with the `COPIES` option of the `BACKUP` command. For example, run the following to make three copies of each backup set in the default `DISK` location:

   ```
   BACKUP AS BACKUPSET DEVICE TYPE DISK
     COPIES 3
     INCREMENTAL LEVEL 0
     DATABASE;
   ```

   Because you specified `COPIES` in the `BACKUP` command, RMAN makes three backup sets of each datafile regardless of the `CONFIGURE DATAFILE COPIES` setting.

2. Issue a `LIST BACKUP` command to see a listing of backup sets and pieces (the `#Copies` column shows the number of copies, which may have been produced through duplexing or through multiple invocations of the `BACKUP` command). For example, enter:

   ```
   LIST BACKUP SUMMARY;
   ```

Making Split Mirror Backups with RMAN

Many sites keep a backup of the database stored on disk in case a **media failure** occurs on the primary database or an incorrect user action requires **point-in-time recovery**. A datafile backup on disk simplifies the restore step of recovery, making recovery much quicker and more reliable.

> **Caution:** Never make backups, split mirror or otherwise, of online redo logs. Restoring online redo log backups can create two archived logs with the same sequence number but different contents. Also, it is best to use the `BACKUP CONTROLFILE` command rather than a split mirror to make control file backups.

One way of creating a datafile backup on disk is to use disk **mirroring**. For example, the operating system can maintain three identical copies of each file in the database. In this configuration, you can split off a mirrored copy of the database to use as a backup.

RMAN does not automate the splitting of mirrors, but can make use of split mirrors in backup and recovery. For example, RMAN can treat a split mirror of a datafile as a datafile copy, and can also back up this copy to disk or tape. The procedure in this section explains how to make a **split mirror backup** with the `ALTER SYSTEM SUSPEND/RESUME` functionality.

Some mirroring technology does not require Oracle Database to suspend all I/O before a mirror can be separated and used as a backup. Refer to your storage manager, volume manager, or file system documentation for information about whether you need to suspend I/O from the database instance.

To make a split mirror backup of a tablespace by using SUSPEND/RESUME:

1. Start RMAN and then place the tablespaces that you want to back up into backup mode with the `ALTER TABLESPACE ... BEGIN BACKUP` statement. (To place all tablespaces in backup mode, you can the `ALTER DATABASE BEGIN BACKUP` instead.)

   For example, to place tablespace `users` in backup mode, you could connect RMAN to a target database and run the following `SQL` command:

   ```
   SQL 'ALTER TABLESPACE users BEGIN BACKUP';
   ```

2. Suspend I/O if your mirroring software or hardware requires it. For example, enter the following command in RMAN:

   ```
   SQL 'ALTER SYSTEM SUSPEND';
   ```

3. Split the mirrors for the underlying datafiles contained in these tablespaces.
4. Take the database out of the suspended state. For example, enter the following command in RMAN:

   ```
   SQL 'ALTER SYSTEM RESUME';
   ```

5. Take the tablespaces out of backup mode. For example, enter:

   ```
   SQL 'ALTER TABLESPACE users END BACKUP';
   ```

   You could also use `ALTER DATABASE END BACKUP` to take all tablespaces out of backup mode.

6. Catalog the user-managed mirror copies as datafile copies with the `CATALOG` command. For example, enter:

   ```
   CATALOG DATAFILECOPY '/dk2/oradata/trgt/users01.dbf'; # catalog split mirror
   ```

7. Back up the datafile copies. For example, run the `BACKUP DATAFILECOPY` command at the prompt:

   ```
   BACKUP DATAFILECOPY '/dk2/oradata/trgt/users01.dbf';
   ```

8. When you are ready to resilver the split mirror, first use the `CHANGE ... UNCATALOG` command to uncatalog the datafile copies you cataloged in step 6. For example, enter:

   ```
   CHANGE DATAFILECOPY '/dk2/oradata/trgt/users01.dbf' UNCATALOG;
   ```

9. Resilver the split mirror for the affected datafiles.

See Also:

- "Making User-Managed Backups in SUSPEND Mode" on page 27-11
- *Oracle Database Administrator's Guide* for more information about the `SUSPEND/RESUME` feature
- *Oracle Database SQL Language Reference* for the `ALTER SYSTEM SUSPEND` syntax

Encrypting RMAN Backups

As explained in "Configuring Backup Encryption" on page 6-7, you can protect RMAN backup sets with **backup encryption**. Encrypted backups cannot be read if they are obtained by unauthorized users. The RMAN backup encryption feature requires the Enterprise Edition of the database.

About RMAN Backup Encryption Settings

Backup encryption is performed based on the encryption settings specified with the following commands:

- `CONFIGURE ENCRYPTION`

  You can use this command to persistently configure transparent encryption. You cannot persistently configure dual mode or password mode encryption.

- `SET ENCRYPTION`

  You can use this command to configure dual mode or password mode encryption at the **RMAN session** level.

> **Note:** Wallet-based encryption is more secure than password-based encryption because no passwords are involved. You should use password-based encryption only when absolutely necessary because your backups need to be transportable.

The database uses a new encryption key for every encrypted backup. The backup encryption key is then encrypted with either the password, the database master key, or both, depending on the chosen encryption mode. Individual backup encryption keys or passwords are never stored in clear text.

A single restore operation can process backups encrypted in different modes. For each backup piece that it restores, RMAN checks whether it is encrypted. Transparently encrypted backups need no intervention if the Oracle wallet is open and available.

If password encryption is detected, then RMAN searches for a matching key in the list of passwords entered in the `SET DECRYPTION` command. If RMAN finds a usable key, then the restore operation proceeds. Otherwise, RMAN searches for a key in the Oracle wallet. If RMAN finds a usable key, then the restore operation proceeds; otherwise, RMAN signals an error that the backup piece cannot be decrypted.

> **Note:** If RMAN restores a set of backups created with different passwords, then all required passwords must be included with `SET DECRYPTION`.

Encryption can have a negative effect upon backup performance. Because encrypted backups consume more CPU resource than unencrypted backups, you can improve performance of encrypted backups to disk by using more RMAN channels.

See Also:

- "Performing Complete Database Recovery" on page 17-9 to learn how to restore password-encrypted backups
- "Determining the Encryption Status of Backup Pieces" on page 10-16
- *Oracle Database Backup and Recovery Reference* to learn about the `ENCRYPTION` and `DECRYPTION` options of the `SET` command

Making Transparent-Mode Encrypted Backups

If you have configured transparent encryption with the `CONFIGURE` command as explained in "Configuring RMAN Backup Encryption Modes" on page 6-9, then no additional commands are required to make encrypted backups. Make RMAN backups as normal.

Making Password-Mode Encrypted Backups

You can set an encryption password in an RMAN session by executing the `SET ENCRYPTION BY PASSWORD` command. If transparent encryption is configured, then specify the `ONLY` keyword to indicate that the backups should be protected with a password and not with the configured transparent encryption.

Note: Create a password that is secure. See *Oracle Database Security Guide* for more information.

To make password-encrypted backups:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Execute the `SET ENCRYPTION ON IDENTIFIED BY` *`password`* `ONLY` command.

   The following example sets the encryption password for all tablespaces (where *password* is a placeholder for the actual password that you enter) in the backup and specifies `ONLY` to indicate that the encryption is password-only:

   ```
   SET ENCRYPTION IDENTIFIED BY password ONLY ON FOR ALL TABLESPACES;
   ```

3. Back up the database.

   For example, enter the following command:

   ```
   BACKUP DATABASE PLUS ARCHIVELOG;
   ```

Making Dual-Mode Encrypted Backups

Use the `SET ENCRYPTION BY PASSWORD` command at the RMAN prompt to make password-protected backups. If transparent encryption is configured, then *omit* the `ONLY` keyword to indicate that the backups should be protected with a password and also with the configured transparent encryption.

Note: Create a password that is secure. See *Oracle Database Security Guide* for more information.

To make dual-mode encrypted backups:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Execute the `SET ENCRYPTION BY PASSWORD` command, making sure to omit the `ONLY` keyword.

   The following example sets the encryption password for all tablespaces (where *password* is a placeholder for the actual password that you enter) in the backup and omits `ONLY` to indicate dual-mode encryption:

   ```
   SET ENCRYPTION IDENTIFIED BY password ON FOR ALL TABLESPACES;
   ```

3. Back up the database.

   For example, enter the following command:

   ```
   BACKUP DATABASE PLUS ARCHIVELOG;
   ```

Restarting RMAN Backups

With the **restartable backup** feature, RMAN backs up only those files that were not backed up after a specified date.

About Restartable Backups

The minimum unit of restartability is a datafile. However, if a backup set contains one backup piece, and if this piece contains blocks from multiple datafiles, then the unit of restartability is the backup piece. The unit of restartability for image copies is a datafile.

The benefit of restartable backups is that if the backup generates multiple backup sets, then the backup sets that completed successfully do not have to be rerun. However, if the entire database is written into one backup set, and if the backup fails halfway through, then the entire backup has to be restarted.

Any I/O errors that RMAN encounters when reading files or writing to the backup pieces or image copies cause RMAN to terminate the backup job in progress. For example, if RMAN tries to back up a datafile but the datafile is not on disk, then RMAN terminates the backup. If multiple channels are being used or redundant copies of backups are being created, however, then RMAN may be able to continue the backup without user intervention.

RMAN can back up only those files that have not been backed up since a specified date. Use this feature after a backup fails to back up the parts of the database missed by the failed backup.

You can restart a backup by specifying the `SINCE TIME` clause on the `BACKUP` command. If the `SINCE TIME` is later than the completion time, then RMAN backs up the file. If you use `BACKUP DATABASE NOT BACKED UP` without the `SINCE TIME` parameter, then RMAN only backs up files that have never been backed up.

See Also: *Oracle Database Backup and Recovery Reference* for `BACKUP ... NOT BACKED UP` syntax

Restarting a Backup After It Partially Completes

Use the SINCE TIME parameter of the BACKUP command to specify a date after which a new backup is required. If the SINCE TIME is later than the completion time, then RMAN backs up the file. If you use BACKUP DATABASE NOT BACKED UP without the SINCE TIME parameter, then RMAN only backs up files that have never been backed up.

To only back up files that were not backed up after a specified date:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Execute the BACKUP ... NOT BACKED UP SINCE TIME command.

   Specify a valid date in the SINCE TIME parameter. The following example uses the default configured channel to back up all database files and archived redo logs that have not been backed up in the last two weeks:

```
BACKUP
  NOT BACKED UP SINCE TIME 'SYSDATE-14'
  DATABASE PLUS ARCHIVELOG;
```

See Also: *Oracle Database Backup and Recovery Reference* for an example of how to use the BACKUP command to restart a backup that did not complete

Managing Backup Windows

This section explains how to use backup windows to set limits for the time span in which a backup job can complete.

About Backup Windows

A **backup window** is a period of time during which a backup must complete. For example, you may want to restrict your database backups to a window of time when user activity on your system is low, such as between 2:00 a.m. and 6:00 a.m.

RMAN backs up the least recently backed up files first. By default, RMAN backs up the files at the maximum possible speed. Specifying a window does not mean that RMAN backs up data faster than normal in order to ensure that the backup completes before the window ends.

By default, if the backup is not complete within the DURATION time, then RMAN interrupts the backup and reports an error. If the BACKUP command is in a RUN command, then the RUN command terminates. Any completed backup sets are retained and can be used in restore operations, even if the entire backup is not complete. Thus, if you retry a job that was interrupted when the available duration expired, each successive attempt covers more of the files needing backup. Any incomplete backup sets are discarded.

Specifying a Backup Duration

Use the DURATION parameter of the BACKUP command to specify how long a given backup job is allowed to run.

To specify a backup duration:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Execute the BACKUP DURATION command.

For example, you could run the following command at 2:00 a.m. to specify that the backup should run until 6:00 a.m.:

```
BACKUP
  DURATION 4:00
  TABLESPACE users;
```

See Also: *Oracle Database Backup and Recovery Reference* for the syntax of the BACKUP command

Permitting Partial Backups in a Backup Window

When you specify PARTIAL, RMAN does not report an error when a backup is interrupted because of the end of the backup window. Instead, RMAN displays a message showing which files could not be backed up. If the BACKUP command is part of a RUN block, then the remaining commands in the RUN block continue to execute.

If you specify FILESPERSET 1, then RMAN puts each file into its own backup set. When a backup is interrupted at the end of the backup window, only the backup of the file currently being backed up is lost. All backup sets completed during the window are saved, minimizing the lost work caused by the end of the backup window.

To prevent RMAN from issuing an error when a backup partially completes:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Execute the BACKUP DURATION command with the PARTIAL option.

   For example, you could run the following command at 2:00 a.m. to specify that the backup should run until 6:00 a.m. and that each datafile should be in a separate backup set:

   ```
   BACKUP
     DURATION 4:00 PARTIAL
     TABLESPACE users
     FILESPERSET 1;
   ```

Minimizing Backup Load and Duration

When using DURATION you can run the backup with the maximum possible performance, or run as slowly as possible while still finishing within the allotted time, to minimize the performance impact of backup tasks. To maximize performance, use the MINIMIZE TIME option with DURATION, as shown in Example 9–4.

Example 9–4 Using MINIMIZE TIME with BACKUP DURATION

```
BACKUP
  DURATION 4:00 PARTIAL
  MINIMIZE TIME
  DATABASE
  FILESPERSET 1;
```

To extend the backup to use the full time available, use the MINIMIZE LOAD option, as in Example 9–5.

Example 9–5 Using MINIMIZE LOAD with BACKUP DURATION

```
BACKUP
  DURATION 4:00 PARTIAL
  MINIMIZE LOAD
  DATABASE
```

```
  FILESPERSET 1;
```

In Example 9–5, RMAN monitors the progress of the running backup, and periodically estimates how long the backup will take to complete at its present rate. If RMAN estimates that the backup will finish before the end of the backup window, then it slows down the rate of backup so that the full available duration will be used. This reduces the overhead on the database associated with the backup.

Note these issues when using `DURATION` and `MINIMIZE LOAD` with a tape backup:

- Efficient backup to tape requires tape streaming. If you use `MINIMIZE LOAD`, then RMAN may reduce the rate of backup to the point where tape streaming is not optimal.
- RMAN will hold the tape resource for the entire duration of the backup window. This prevents the use of the tape resource for any other purpose during the backup window.

Because of these concerns, it is not recommended that you use `MINIMIZE LOAD` when backing up to tape.

> **See Also:** "Media Manager Component of Write Phase for SBT" on page 21-8 for more details on efficient tape handling

Part IV

Managing RMAN Backups

The following chapters describe how to manage RMAN backups. This part of the book contains these chapters:

- Chapter 10, "Reporting on RMAN Operations"
- Chapter 11, "Maintaining RMAN Backups and Repository Records"
- Chapter 12, "Managing a Recovery Catalog"

10

Reporting on RMAN Operations

This chapter describes how to report on RMAN operations. This chapter includes the following topics:

- Overview of RMAN Reporting
- Listing Backups and Recovery-Related Objects
- Reporting on Backups and Database Schema
- Using V$ Views to Query Backup Metadata
- Querying Recovery Catalog Views

Overview of RMAN Reporting

This section explains the purpose and basic concepts of RMAN reporting.

Purpose of RMAN Reporting

As part of your backup and recovery strategy, you should periodically run reports that indicate what you have backed up. You should also determine which datafiles need backups or which files have not been backed up recently. Also, you can preview which backups RMAN would need to restore if a problem were to occur.

Another important aspect of backup and recovery is monitoring space usage. If you back up to disk, then it is possible for the disk to fill, which can create performance problems or even cause the database to halt. You can use RMAN to determine whether a backups is an **obsolete backup** and can therefore be deleted.

You may also need to obtain historical information about RMAN jobs. For example, you may want to know how many backup jobs have been issued, the status of each backup job (for example, whether it failed or completed), when a job started and finished, and what type of backup was performed.

Basic Concepts of RMAN Reporting

RMAN always stores its **RMAN repository** of metadata in the control file of each target database on which it performs operations. For example, suppose that you use RMAN to back up the `prod1` and `prod2` databases. RMAN stores the metadata for backups of `prod1` in the control file of `prod1`, and the metadata for backups of `prod2` in the control file of `prod2`.

Optionally, you can use RMAN with a **recovery catalog**. In this case, RMAN maintains an additional repository of metadata in a set of tables in a separate **recovery catalog database**. For example, you could create a recovery catalog in `prod3`. You can register

multiple target databases in this recovery catalog. For example, if you register `prod1` and `prod2` in the recovery catalog stored in `prod3`, then RMAN stores metadata about its backups of `prod1` and `prod2` in the **recovery catalog schema**.

You can access metadata from the RMAN repository in several different ways:

- The RMAN `LIST` and `REPORT` commands provide extensive information on available backups and how they can be used to restore and recover your database.

  The `LIST` command is described in "Listing Backups and Recovery-Related Objects" on page 10-3 and `REPORT` is described in "Reporting on Backups and Database Schema" on page 10-10.

- When the database is open, a number of `V$` views provide direct access to RMAN repository records in the control file of each target database.

  Some `V$` views such as `V$DATAFILE_HEADER`, `V$PROCESS`, and `V$SESSION` contain information not found in the recovery catalog views. The `V$` views are documented in *Oracle Database Reference*.

- If your database is registered in a recovery catalog, then `RC_` views provide direct access to the RMAN repository data stored in the recovery catalog.

  The `RC_` views that mostly correspond to the `V$` views. The `RC_` views are documented in *Oracle Database Backup and Recovery Reference*.

- The `RESTORE ... PREVIEW` and `RESTORE ... VALIDATE HEADER` commands lists the backups that RMAN could restore to the specified time.

  `RESTORE ... PREVIEW` queries the metadata and does not actually read the backup files. The `RESTORE ... VALIDATE HEADER` command performs the same work, but in addition to listing the files needed for restore and recovery, the command validates the backup file headers to determine whether the files on disk or in the media management catalog correspond to the metadata in the RMAN repository. These commands are documented in "Previewing Backups Used in Restore Operations" on page 17-5.

As explained in Chapter 11, "Maintaining RMAN Backups and Repository Records," the RMAN repository can sometimes fail to reflect the reality on disk and tape. For example, a user may delete a backup with an operating system utility, so that the RMAN repository incorrectly reports the backup as available.

You can use commands such as `CHANGE`, `CROSSCHECK`, and `DELETE` to update the RMAN repository to reflect the actual state of available backups. Otherwise, the output of the commands and views may be misleading, which means that RMAN may not be able to find the backups to restore and recover your database.

See Also:

- "Crosschecking the RMAN Repository" on page 11-12 to learn how to keep the RMAN repository current
- *Oracle Database Backup and Recovery Reference* for `LIST` syntax
- *Oracle Database Backup and Recovery Reference* for `REPORT` syntax
- *Oracle Database Backup and Recovery Reference* for `RESTORE PREVIEW` syntax

Reporting in a Data Guard Environment

As explained in "RMAN File Management in a Data Guard Environment" on page 3-8, every backup is associated with the primary or standby database that created it. For

example, if you backed up the database with the DB_UNIQUE_NAME of standby1, then the standby1 database is associated with this backup.

In a Data Guard environment, you can use the LIST, REPORT, and SHOW commands just as you can when not using Data Guard. You can run these commands with the FOR DB_UNIQUE_NAME clause to show the backups associated with a specified database. For example, the following command lists archived redo logs associated only with sfstandby:

```
LIST ARCHIVELOG ALL FOR DB_UNIQUE_NAME sfstandby;
```

If you use the LIST, REPORT, and SHOW commands in a Data Guard environment *without* specifying the FOR DB_UNIQUE_NAME clause, then RMAN shows the files that are accessible to the target database. "Association of Backups in a Data Guard Environment" on page 3-8 explains when backups are considered accessible to RMAN.

In a Data Guard environment, you must use RMAN with a recovery catalog. RMAN stores the metadata for all backup and recovery files in the Data Guard environment in the recovery catalog. When running the RMAN reporting commands, you can either connect RMAN as TARGET to a mounted or open database, or identify the database with the SET DBID command.

> **See Also:** *Oracle Data Guard Concepts and Administration* to report on RMAN operations in a Data Guard environment

Listing Backups and Recovery-Related Objects

The LIST command uses the information in the RMAN repository to provide lists of backups and other objects relating to backup and recovery. This section contains the following topics:

- About the LIST Command
- Listing Backups and Copies
- Listing Database Incarnations
- Listing Restore Points

About the LIST Command

The primary purpose of the LIST command is to list backup and copies. For example, you can list:

- Backups and proxy copies of a database, tablespace, datafile, archived redo log, or control file
- Backups that have expired
- Backups restricted by time, path name, device type, tag, or recoverability
- Archived redo log files and disk copies

The primary purpose of the LIST command is to list backup and copies. Besides backups and copies, the RMAN can list other types of data. The following table summarizes several useful objects that you can list.

Table 10–1 LIST Objects

Contents of List	Command	Description
Backup sets and proxy copies	`LIST BACKUP`	You can list all backup sets, copies, and proxy copies of a database, tablespace, datafile, archived redo log, control file, or server parameter file.
Image copies	`LIST COPY`	You can list datafile copies and archived redo log files. By default, `LIST COPY` displays copies of all database files and archived redo logs. Both usable and unusable image copies are included in the output, even those that cannot be restored or are expired or unavailable.
Archived redo log files	`LIST ARCHIVELOG`	You can list archived redo log files.
Database incarnations	`LIST INCARNATION`	You can list all incarnations of a database. A new database **incarnation** is created when you open with the `RESETLOGS` option.
Databases in a Data Guard environment	`LIST DB_UNIQUE_NAME`	A database in a Data Guard environment is distinguished by its `DB_UNIQUE_NAME` initialization parameter setting. You can list all databases that have the same DBID.
Backups and copies for a primary or standby database in a Data Guard environment	`LIST ... FOR DB_UNIQUE_NAME`	You can list all backups and copies for a specified database in a Data Guard environment or for all databases in the environment. RMAN restricts the output to files or objects associated exclusively with the database with the specified `DB_UNIQUE_NAME`. For example, you can use `LIST` with `FOR DB_UNIQUE_NAME` to display the list of archived redo log files associated with a particular standby or primary database. Note that objects that are not owned by any database (`SITE_KEY` column in the recovery catalog view is null) are not listed.
Restore points	`LIST RESTORE POINT`	You can list restore points known to the RMAN repository.
Names of stored scripts	`LIST SCRIPT NAMES`	You can list the names of recovery catalog scripts created with the `CREATE SCRIPT` or `REPLACE SCRIPT` command. A recovery catalog is required.
Failures for use with **Data Recovery Advisor**	`LIST FAILURE`	A **failure** is a persistent data corruption mapped to a **repair option**. Chapter 14, "Diagnosing and Repairing Failures with Data Recovery Advisor" explains how to use `LIST FAILURE` in conjunction with the `ADVISE` and `REPAIR` commands.

The `LIST` command supports a number of options that enables you to control how output is displayed. The following table summarizes the most common `LIST` options.

Table 10–2 Most Common LIST Options

LIST Option	Description
`LIST EXPIRED`	Lists backups or copies that are recorded in the RMAN repository but that were not present at the expected location on disk or tape during the most recent **crosscheck**. Such backups may have been deleted outside of RMAN.
`LIST ... BY FILE`	Lists backups of each datafile, archived redo log file, control file, and server parameter file. Each row describes a backup of a file.
`LIST ... SUMMARY`	Provides a one-line summary of each backup.

The `LIST` objects and options are not exhausted by the contents of the preceding tables. For example, you can list backups restricted by time, path name, device type, tag, or recoverability.

See Also: *Oracle Database Backup and Recovery Reference* for a complete description of the `LIST` command

Listing Backups and Copies

Specify the desired objects with the *`listObjList`* or *`recordSpec`* clause (refer to *Oracle Database Backup and Recovery Reference*). If you do not specify an object, then RMAN displays copies of all database files and archived redo log files.

By default, RMAN serially lists each backup or proxy copy and then identifies the files included in the backup. You can also list backups by file.

By default, RMAN lists in verbose mode, which means that it provides extensive, multiline information. You can also list backups in a summary mode if the verbose mode generates too much output.

To list backups and copies:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. To view a summary report of all backups and copies, execute the `LIST` command with the `SUMMARY` option.

   The following command prints a summary of all RMAN backups:

   ```
   LIST BACKUP SUMMARY;
   ```

   Sample output follows:

   ```
   List of Backups
   ===============
   Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
   ------- -- -- - ----------- --------------- ------- ------- ---------- ---
   1       B  A  A SBT_TAPE    21-OCT-07       1       1       NO
   TAG20071021T094505
   2       B  F  A SBT_TAPE    21-OCT-07       1       1       NO
   TAG20071021T094513
   3       B  A  A SBT_TAPE    21-OCT-07       1       1       NO
   TAG20071021T094624
   4       B  F  A SBT_TAPE    21-OCT-07       1       1       NO
   TAG20071021T094639
   5       B  F  A DISK        04-NOV-07       1       1       YES
   TAG20071104T195949
   ```

3. To view verbose output for backups and copies, execute the `LIST` command without the `SUMMARY` option.

   The following commands list RMAN backups and copies with the default verbose output:

   ```
   LIST BACKUP;
   LIST COPY;
   ```

   Sample output for `LIST BACKUP` and `LIST COPY` follows:

   ```
   List of Backup Sets
   ===================

   BS Key  Size       Device Type Elapsed Time Completion Time
   ------- ---------- ----------- ------------ ---------------
   7       136M       DISK        00:00:20     04-NOV-06
           BP Key: 7   Status: AVAILABLE  Compressed: NO  Tag: TAG20071104T200759
           Piece Name: /d2/RDBMS/backupset/2007_11_04/o1_mf_annnn_
   ```

```
TAG20071104T200759_ztjxx3k8_.bkp

  List of Archived Logs in backup set 7
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    1       173832     21-OCT-06 174750     21-OCT-06
  1    2       174750     21-OCT-06 174755     21-OCT-06
  1    3       174755     21-OCT-06 174758     21-OCT-06

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
8       Full    2M         DISK        00:00:01     04-NOV-06
        BP Key: 8   Status: AVAILABLE  Compressed: NO  Tag: TAG20071104T200829
        Piece Name: /disk1/oracle/dbs/c-774627068-20071104-01
  Controlfile Included: Ckp SCN: 631510       Ckp time: 04-NOV-06
  SPFILE Included: Modification time: 21-OCT-06

List of Datafile Copies
=======================

Key     File S Completion Time Ckp SCN    Ckp Time
------- ---- - --------------- ---------- ---------------
1       7    A 11-OCT-06       360072     11-OCT-06
        Name: /work/orcva/RDBMS/datafile/o1_mf_tbs_2_2lv7bf82_.dbf
        Tag: DF7COPY

2       8    A 11-OCT-06       360244     11-OCT-06
        Name: /work/orcva/RDBMS/datafile/o1_mf_tbs_2_2lv7qmcj_.dbf
        Tag: TAG20071011T184835

List of Control File Copies
===========================

Key     S Completion Time Ckp SCN    Ckp Time
------- - --------------- ---------- ---------------
3       A 11-OCT-06       360380     11-OCT-06
        Name: /d2/RDBMS/controlfile/o1_mf_TAG20071011T185335_2lv80zqd_.ctl
        Tag: TAG20071011T185335

List of Archived Log Copies for database with db_unique_name RDBMS
=====================================================================

Key     Thrd Seq     S Low Time
------- ---- ------- - ---------
1       1    1       A 11-OCT-06
        Name: /work/arc_dest/arcr_1_1_603561743.arc

2       1    2       A 11-OCT-06
        Name: /work/arc_dest/arcr_1_2_603561743.arc

3       1    3       A 11-OCT-06
        Name: /work/arc_dest/arcr_1_3_603561743.arc
```

4. To list backups by file, execute `LIST` with the `BY FILE` option, specifying the desired objects to list and options. For example, you can enter:

   ```
   LIST BACKUP BY FILE;
   ```

   Sample output follows:

   ```
   List of Datafile Backups
   ```

```
========================

File Key     TY LV S Ckp SCN    Ckp Time  #Pieces #Copies Compressed Tag
---- ------- -  -- - ---------- --------- ------- ------- ---------- ---
1    5       B  F  A 631092     04-NOV-06 1       1       YES
TAG20071104T195949
     2       B  F  A 175337     21-OCT-06 1       1       NO
TAG20071021T094513
2    5       B  F  A 631092     04-NOV-06 1       1       YES
TAG20071104T195949
     2       B  F  A 175337     21-OCT-06 1       1       NO
TAG20071021T094513

... some rows omitted

List of Archived Log Backups
============================

Thrd Seq     Low SCN    Low Time  BS Key  S #Pieces #Copies Compressed Tag
---- ------- ---------- --------- ------- - ------- ------- ---------- ---
1    1       173832     21-OCT-06 7       A 1       1       NO
TAG20071104T200759
                                   1       A 1       1       NO
TAG20071021T094505
1    2       174750     21-OCT-06 7       A 1       1       NO
TAG20071104T200759
                                   1       A 1       1       NO
TAG20071021T094505
... some rows omitted
1    38      575472     03-NOV-06 7       A 1       1       NO
TAG20071104T200759
1    39      617944     04-NOV-06 7       A 1       1       NO
TAG20071104T200759


List of Controlfile Backups
===========================
CF Ckp SCN Ckp Time  BS Key  S #Pieces #Copies Compressed Tag
---------- --------- ------- - ------- ------- ---------- ---
631510     04-NOV-06 8       A 1       1       NO         TAG20071104T200829
631205     04-NOV-06 6       A 1       1       NO         TAG20071104T200432

List of SPFILE Backups
======================
Modification Time BS Key  S #Pieces #Copies Compressed Tag
----------------- ------- - ------- ------- ---------- ---
21-OCT-06         8       A 1       1       NO         TAG20071104T200829
21-OCT-06         6       A 1       1       NO         TAG20071104T200432
```

See Also: *Oracle Database Backup and Recovery Reference* for an explanation of the various column headings in the `LIST` output

Listing Selected Backups and Copies

You can specify several different conditions to narrow your `LIST` output.

To list selected backups and copies:

1. Start RMAN and connect to a target database and recovery catalog (if used).

2. Run `LIST COPY` or `LIST BACKUP` with the *`listObjList`* or *`recordSpec`* clause. For example, enter any of the following commands:

```
# lists backups of all files in database
LIST BACKUP OF DATABASE;
# lists copy of specified datafile
LIST COPY OF DATAFILE 'ora_home/oradata/trgt/system01.dbf';
# lists specified backup set
LIST BACKUPSET 213;
# lists datafile copy
LIST DATAFILECOPY '/tmp/tools01.dbf';
```

   You can also restrict the search by specifying the *`maintQualifier`* or `RECOVERABLE` clause. For example, enter any of the following commands:

```
# specify a backup set by tag
LIST BACKUPSET TAG 'weekly_full_db_backup';
# specify a backup or copy by device type
LIST COPY OF DATAFILE 'ora_home/oradata/trgt/system01.dbf' DEVICE TYPE sbt;
# specify a backup by directory or path
LIST BACKUP LIKE '/tmp/%';
# specify a backup or copy by a range of completion dates
LIST COPY OF DATAFILE 2 COMPLETED BETWEEN '10-DEC-2002' AND '17-DEC-2002';
# specify logs backed up at least twice to tape
LIST ARCHIVELOG ALL BACKED UP 2 TIMES TO DEVICE TYPE sbt;
```

3. Examine the output.

   The output depends upon the options you pass to the `LIST` command. For example, the following lists copies of datafile `1`:

```
RMAN> LIST BACKUP OF DATAFILE 1;

List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
2       Full    230M       SBT_TAPE    00:00:49     21-OCT-06
        BP Key: 2   Status: AVAILABLE  Compressed: NO  Tag: TAG20071021T094513
        Handle: 02f4eatc_1_1   Media: /smrdir
  List of Datafiles in backup set 2
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 175337     21-OCT-06 /oracle/dbs/tbs_01.f

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
5       Full    233M       DISK        00:04:30     04-NOV-06
        BP Key: 5   Status: AVAILABLE  Compressed: NO  Tag: TAG20071104T195949
        Piece Name: /disk1/2007_11_04/o1_mf_nnndf_TAG20071104T195949_ztjxfvgz_
.bkp
  List of Datafiles in backup set 5
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 631092     04-NOV-06 /oracle/dbs/tbs_01.f
```

See Also:

- *Oracle Database Backup and Recovery Reference* for `listObjList` and `recordSpec` syntax
- *Oracle Database Backup and Recovery Reference* for an explanation of the columns in the `LIST` output

Listing Database Incarnations

Each time an `OPEN RESETLOGS` operation is performed on a database, this operation creates a new **incarnation** of the database. Database incarnations and their effect on database recovery are explained in "Database Incarnations" on page 13-5.

When performing incremental backups, RMAN can use a backup from a previous incarnation or the current incarnation as a basis for subsequent incremental backups. When performing restore and recovery, RMAN can use backups from a previous incarnation in restore and recovery operations just as it would use backups from the current incarnation, as long as all archived logs are available.

To list database incarnations:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Run the `LIST INCARNATION` command, as shown in the following example:

   ```
   LIST INCARNATION;
   ```

   If you are using a recovery catalog, and if you register multiple target databases in the same catalog, then you can distinguish them by using the `OF DATABASE` option:

   ```
   LIST INCARNATION OF DATABASE prod3;
   ```

   Refer to *Oracle Database Backup and Recovery Reference* for an explanation of the various column headings in the `LIST` output). Sample output follows:

   ```
   RMAN> LIST INCARNATION OF DATABASE;

   List of Database Incarnations
   DB Key  Inc Key DB Name  DB ID            STATUS  Reset SCN  Reset Time
   ------- ------- -------- ---------------- ------  ---------- ----------
   1       1       RDBMS    774627068        PARENT  1          21-OCT-06
   2       2       RDBMS    774627068        CURRENT 173832     21-OCT-06
   ```

   The preceding output indicates that a `RESETLOGS` was performed on database `trgt` at SCN 164378, resulting in a new incarnation. The incarnation is distinguished by incarnation key (represented in the `Inc Key` column).

Listing Restore Points

You can use the `LIST` command to list either a specific restore point or all restore points known to the RMAN repository. The variations of the command are as follows:

```
LIST RESTORE POINT restore_point_name;
LIST RESTORE POINT ALL;
```

RMAN indicates the SCN and time of the restore point, the type of restore point, and the name of the restore point. The following example shows sample output:

```
RMAN> LIST RESTORE POINT ALL;
```

```
using target database control file instead of recovery catalog
SCN              RSP Time  Type       Time      Name
---------------- --------- ---------- --------- ----
341859           28-JUL-06            28-JUL-06 NORMAL_RS
343690           28-JUL-06 GUARANTEED 28-JUL-06 GUARANTEED_RS
```

You can also see a list of the currently defined restore points by querying the V$RESTORE_POINT view as follows:

```
SELECT NAME, SCN, TIME, DATABASE_INCARNATION#,
       GUARANTEE_FLASHBACK_DATABASE,STORAGE_SIZE
FROM   V$RESTORE_POINT;
```

You can view the name of each restore point, the SCN, wall-clock time and database **incarnation** number at which the restore points were created, whether each restore point is a guaranteed restore point, and how much space in the recovery area is being used for data needed for Flashback Database operations to that restore point.

You can also use the following query to view only the guaranteed restore points:

```
SELECT NAME, SCN, TIME, DATABASE_INCARNATION#,
       GUARANTEE_FLASHBACK_DATABASE, STORAGE_SIZE
FROM   V$RESTORE_POINT
WHERE  GUARANTEE_FLASHBACK_DATABASE='YES';
```

For normal restore points, STORAGE_SIZE is zero. For guaranteed restore points, STORAGE_SIZE indicates the amount of disk space in the flash recovery area used to retain logs required to guarantee FLASHBACK DATABASE to that restore point.

See Also:

- *Oracle Database Reference* for information about V$RESTORE_POINT
- "Rewinding a Database with Flashback Database" on page 16-11

Reporting on Backups and Database Schema

The RMAN REPORT command analyzes the available backups and your database. This section contains the following topics:

- About Reports of RMAN Backups
- Reporting on Files Needing a Backup Under a Retention Policy
- Reporting on Datafiles Affected by Unrecoverable Operations
- Reporting on Obsolete Backups
- Reporting on the Database Schema

About Reports of RMAN Backups

You can use the REPORT command to answer important questions, such as:

- Which files need a backup?
- Which files have had unrecoverable operations performed on them?
- Which backups are obsolete and can be deleted?
- What was the physical schema of the target database or a database in the Data Guard environment at some previous time?

- Which files have not been backed up recently?

Reports enable you to confirm that your backup and recovery strategy is in fact meeting your requirements for database recoverability. The two major forms of `REPORT` used to determine whether your database is recoverable are:

- `REPORT NEED BACKUP`

  Reports which database files need to be backed up to meet a configured or specified retention policy

- `REPORT UNRECOVERABLE`

  Reports which database files require backup because they have been affected by some `NOLOGGING` operation such as a direct-path `INSERT`

The RMAN repository contains other information that you can access with the `REPORT` command. Table 10–3 summarizes the `REPORT` options.

Table 10–3 REPORT Options

Contents of Report	Command	Description
Obsolete backups	`REPORT OBSOLETE`	Full backups, datafile copies, and archived redo logs recorded in the RMAN repository that can be deleted because they are no longer needed.
Database schema	`REPORT SCHEMA`	The names of all datafiles (permanent and temporary) and tablespaces for the target database at the specified point in time. If you use RMAN in a Data Guard environment, then you can report the schema for a specified `DB_UNIQUE_NAME`.

See Also: *Oracle Database Backup and Recovery Reference* for a description of the `REPORT` command

Reporting on Files Needing a Backup Under a Retention Policy

Use the `REPORT NEED BACKUP` command to determine which database files need backup under a specific retention policy.

With no arguments, `REPORT NEED BACKUP` reports which objects need backup under the currently configured retention policy. The output for a configured retention policy of `REDUNDANCY 1` is similar to this example:

```
RMAN> REPORT NEED BACKUP;

RMAN retention policy will be applied to the command
RMAN retention policy is set to redundancy 1
Report of files with less than 1 redundant backups
File #bkps Name
---- ----- -----------------------------------------------------
2    0     /oracle/oradata/trgt/undotbs01.dbf
```

Note: If you disable the retention policy using `CONFIGURE RETENTION POLICY TO NONE`, then `REPORT NEED BACKUP` returns an error message, because without a retention policy, RMAN cannot determine which files need to be backed up.

Using RMAN REPORT NEED BACKUP with Different Retention Policies

You can specify different criteria for `REPORT NEED BACKUP`, using one of the following forms of the command:

- `REPORT NEED BACKUP RECOVERY WINDOW OF` *n* `DAYS`

  Displays objects requiring backup to satisfy a recovery window-based retention policy.

- `REPORT NEED BACKUP REDUNDANCY` *n*

  Displays objects requiring backup to satisfy a redundancy-based retention policy.

- `REPORT NEED BACKUP DAYS` *n*

  Displays files that require more than *n* days' worth of archived redo log files for recovery.

- `REPORT NEED BACKUP INCREMENTAL` *n*

  Displays files that require application of more than *n* incremental backups for recovery.

Using RMAN REPORT NEED BACKUP with Tablespaces and Datafiles

`REPORT NEED BACKUP` can check the entire database, skip specified tablespaces, or check only specific tablespaces or datafiles against different retention policies, as shown in the following examples:

```
REPORT NEED BACKUP RECOVERY WINDOW OF 2 DAYS DATABASE SKIP TABLESPACE TBS_2;
REPORT NEED BACKUP REDUNDANCY 2 DATAFILE 1;
REPORT NEED BACKUP TABLESPACE TBS_3; # uses configured retention policy
REPORT NEED BACKUP INCREMENTAL 2; # checks entire database
```

> **See Also:** *Oracle Database Backup and Recovery Reference* for all possible options for `REPORT NEED BACKUP` and an explanation of the various column headings in the output

Using REPORT NEED BACKUP with Backups on Tape or Disk Only

You can limit the backups tested by `REPORT NEED BACKUP` to disk-based or tape-based backups only, as shown in these examples:

```
REPORT NEED BACKUP RECOVERY WINDOW OF 2 DAYS DATABASE DEVICE TYPE sbt;
REPORT NEED BACKUP DEVICE TYPE DISK;
REPORT NEED BACKUP TABLESPACE TBS_3 DEVICE TYPE sbt;
```

Reporting on Datafiles Affected by Unrecoverable Operations

When a datafile has been changed by an unrecoverable operation, such as a direct load insert, normal media recovery cannot be used to recover the file, because an unrecoverable operation does not generate redo. You must perform either a full or incremental backup of affected datafiles after such operations, to ensure that data blocks affected by the unrecoverable operation can be recovered using RMAN.

To identify datafiles affected by an unrecoverable operation:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Execute the `REPORT UNRECOVERABLE` command.

   The following example includes sample output:

   ```
   RMAN> REPORT UNRECOVERABLE;
   ```

```
Report of files that need backup due to unrecoverable operations
File Type of Backup Required Name
---- ----------------------- -----------------------------------
1    full                    /oracle/oradata/trgt/system01.dbf
```

Reporting on Obsolete Backups

You can report backup sets, backup pieces and datafile copies that are obsolete, that is, not needed to meet a specified retention policy, by specifying the OBSOLETE keyword.

To report obsolete backups:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Execute the CROSSCHECK command to update the status of backups in the repository compared to their status on disk.

   In the simplest case, you could crosscheck all backups on disk, tape or both, using any one of the following commands:

   ```
   CROSSCHECK BACKUP DEVICE TYPE DISK;
   CROSSCHECK BACKUP DEVICE TYPE sbt;
   CROSSCHECK BACKUP; # crosshecks all backups on all devices
   ```

   See Chapter 11, "Maintaining RMAN Backups and Repository Records" for more details on how to update the RMAN repository record to contain the actual set of available backups.
3. Run REPORT OBSOLETE to identify which backups are obsolete because they are no longer needed for recovery.

   If you do not specify any other options, then REPORT OBSOLETE displays the backups that are obsolete according to the current retention policy, as shown in the following example:

   ```
   RMAN> REPORT OBSOLETE;

   Datafile Copy        44     08-FEB-06          /backup/ora_df549738566_s70_s1
   Datafile Copy        45     08-FEB-06          /backup/ora_df549738567_s71_s1
   Datafile Copy        46     08-FEB-06          /backup/ora_df549738568_s72_s1
   Backup Set           26     08-FEB-06
     Backup Piece       26     08-FEB-06          /backup/ora_df549738682_s76_s1
   .
   .
   .
   ```

   You can also check which backups are obsolete under different recovery window-based or redundancy-based retention policies, by using REPORT OBSOLETE with RECOVERY WINDOW and REDUNDANCY options, as shown in these examples:

   ```
   REPORT OBSOLETE RECOVERY WINDOW OF 3 DAYS;
   REPORT OBSOLETE REDUNDANCY 1;
   ```

See Also:

- "Configuring the Backup Retention Policy" on page 5-21 for a conceptual overview of RMAN backup retention policy
- "Deleting Expired RMAN Backups and Copies" on page 11-23 for information on deleting RMAN backups and deleting records of RMAN backups from the RMAN repository

Reporting on the Database Schema

The `REPORT SCHEMA` command lists and displays information about the database files, tablespaces, and so on. See *Oracle Database Backup and Recovery Reference* for a description of the `REPORT SCHEMA` output.

If you do not specify `FOR DB_UNIQUE_NAME` with `REPORT SCHEMA`, then a recovery catalog connection is optional, but a target database connection is required. In a Data Guard environment, you can specify `REPORT SCHEMA FOR DB_UNIQUE_NAME` to report the schema for a database in the environment. In this case, an RMAN connection to a target database is not required. You can connect RMAN to the recovery catalog and set the DBID instead.

To report on the database schema:

1. Start RMAN and connect to the desired databases.
2. If you did not connect RMAN to a target database in the previous step, and you intend to specify the `FOR DB_UNIQUE_NAME` clause on `REPORT SCHEMA`, then set the database DBID. For example, enter the following command:

   ```
   RMAN> SET DBID 28014364;
   ```

3. Run the `REPORT SCHEMA` command as shown in the following example:

   ```
   RMAN> REPORT SCHEMA;

   Report of database schema for database with db_unique_name DGRDBMS

   List of Permanent Datafiles
   ===========================
   File Size(MB) Tablespace           RB segs Datafile Name
   ---- -------- -------------------- ------- ------------------------
   1    450      SYSTEM               YES     /disk1/oracle/dbs/t_db1.f
   2    141      SYSAUX               NO      /disk1/oracle/dbs/t_ax1.f
   3    50       UD1                  YES     /disk1/oracle/dbs/t_undo1.f
   4    50       TBS_11               NO      /disk1/oracle/dbs/tbs_111.f
   5    50       TBS_11               NO      /disk1/oracle/dbs/tbs_112.f

   List of Temporary Files
   =======================
   File Size(MB) Tablespace           Maxsize(MB) Tempfile Name
   ---- -------- -------------------- ----------- --------------------
   1    40       TEMP                 32767       /disk1/oracle/dbs/t_tmp1.f
   ```

   If you use a recovery catalog, then you can use the *atClause* to specify a past time, SCN, or log sequence number, as shown in these examples of the command:

   ```
   RMAN> REPORT SCHEMA AT TIME 'SYSDATE-14';     # schema 14 days ago
   RMAN> REPORT SCHEMA AT SCN 1000;              # schema at scn 1000
   RMAN> REPORT SCHEMA AT SEQUENCE 100 THREAD 1; # schema at sequence 100
   RMAN> REPORT SCHEMA FOR DB_UNIQUE_NAME standby1; # schema for database standby1
   ```

Using V$ Views to Query Backup Metadata

In some cases, V$ views supply information that is not available through use of the LIST and REPORT commands. This section describes cases in which V$ views are particularly useful.

Querying Details of Past and Current RMAN Jobs

An **RMAN job** is the set of commands executed within an **RMAN session**. Thus, one RMAN job can contain multiple commands. For example, you may execute two separate BACKUP commands and a RECOVER COPY command in a single session. An **RMAN backup job** is the set of BACKUP commands executed in one RMAN job. For example, a BACKUP DATABASE and BACKUP ARCHIVELOG ALL command executed in the same RMAN job make up a single RMAN backup job.

The views V$RMAN_BACKUP_JOB_DETAILS and V$RMAN_BACKUP_SUBJOB_DETAILS and their corresponding recovery catalog versions provide details on RMAN backup jobs. For example, the views show how long a backup took, how many backup jobs have been issued, the status of each backup job (for example, whether it failed or completed), when a job started and finished, and what type of backup was performed. The SESSION_KEY column is the unique key for the RMAN session in which the backup job occurred.

Note that RMAN backups often write less than they read. Because of RMAN compression, the OUTPUT_BYTES_PER_SEC column cannot be used as measurement of backup speed. The appropriate column to measure backup speed is INPUT_BYTES_PER_SEC. The ratio between read and written data is described in the COMPRESSION_RATIO column.

To query details about past and current RMAN jobs:

1. Connect SQL*Plus to the database whose backup history you intend to query.
2. Query the V$RMAN_BACKUP_JOB_DETAILS view for information about the backup type, status, and start and end time.

   The following query shows the backup job history ordered by session key, which is the primary key for the RMAN session:

   ```
   COL STATUS FORMAT a9
   COL hrs    FORMAT 999.99
   SELECT SESSION_KEY, INPUT_TYPE, STATUS,
          TO_CHAR(START_TIME,'mm/dd/yy hh24:mi') start_time,
          TO_CHAR(END_TIME,'mm/dd/yy hh24:mi')   end_time,
          ELAPSED_SECONDS/3600                   hrs
   FROM V$RMAN_BACKUP_JOB_DETAILS
   ORDER BY SESSION_KEY;
   ```

   The following sample output shows the backup job history:

   ```
   SESSION_KEY INPUT_TYPE    STATUS    START_TIME     END_TIME           HRS
   ----------- ------------- --------- -------------- -------------- -------
             9 DATAFILE FULL COMPLETED 04/18/07 18:14 04/18/07 18:15     .02
            16 DB FULL       COMPLETED 04/18/07 18:20 04/18/07 18:22     .03
           113 ARCHIVELOG    COMPLETED 04/23/07 16:04 04/23/07 16:05     .01
   ```

3. Query the V$RMAN_BACKUP_JOB_DETAILS view for the rate of backup jobs in an RMAN session.

The following query shows the backup job speed ordered by session key, which the primary key for the RMAN session. The columns `in_sec` and `out_sec` columns display the data input and output per second.

```
COL in_sec FORMAT a10
COL out_sec FORMAT a10
COL TIME_TAKEN_DISPLAY FORMAT a10
SELECT SESSION_KEY,
       OPTIMIZED,
       COMPRESSION_RATIO,
       INPUT_BYTES_PER_SEC_DISPLAY in_sec,
       OUTPUT_BYTES_PER_SEC_DISPLAY out_sec,
       TIME_TAKEN_DISPLAY
FROM   V$RMAN_BACKUP_JOB_DETAILS
ORDER BY SESSION_KEY;
```

The following sample output shows the speed of the backup jobs:

```
SESSION_KEY OPT COMPRESSION_RATIO IN_SEC     OUT_SEC    TIME_TAKEN
----------- --- ----------------- ---------- ---------- ----------
          9 NO                  1     8.24M      8.24M  00:01:14
         16 NO         1.32732239     6.77M      5.10M  00:01:45
        113 NO                  1     2.99M      2.99M  00:00:44
```

4. Query the `V$RMAN_BACKUP_JOB_DETAILS` view for the size of the backups in an RMAN session.

   If you run `BACKUP DATABASE`, then `V$RMAN_BACKUP_JOB_DETAILS.OUTPUT_BYTES` shows the total size of backup sets written by the backup job for the database that you are backing up. To view backup set sizes for all registered databases, query `RC_RMAN_BACKUP_JOB_DETAILS`.

   The following query shows the backup job speed ordered by session key, which the primary key for the RMAN session. The columns `in_sec` and `out_sec` columns display the data input and output per second.

   ```
   COL in_size  FORMAT a10
   COL out_size FORMAT a10
   SELECT SESSION_KEY,
          INPUT_TYPE,
          COMPRESSION_RATIO,
          INPUT_BYTES_DISPLAY in_size,
          OUTPUT_BYTES_DISPLAY out_size
   FROM   V$RMAN_BACKUP_JOB_DETAILS
   ORDER BY SESSION_KEY;
   ```

   The following sample output shows the speed of the backup jobs:

   ```
   SESSION_KEY INPUT_TYPE    COMPRESSION_RATIO IN_SIZE    OUT_SIZE
   ----------- ------------- ----------------- ---------- ----------
            10 DATAFILE FULL                 1   602.50M    602.58M
            17 DB FULL              1.13736669   634.80M    558.13M
   ```

   See Also: *Oracle Database Reference* to learn about the `V$RMAN_BACKUP_JOB_DETAILS` view

Determining the Encryption Status of Backup Pieces

The `ENCRYPTED` column of `V$BACKUP_PIECE` and `RC_BACKUP_PIECE` indicates whether a backup piece is encrypted (`YES`) or unencrypted (`NO`). For example, you can run the following query in SQL*Plus to determine which backup pieces are encrypted:

```
COL BS_REC    FORMAT 99999
COL BP_REC    FORMAT 99999
COL MB        FORMAT 9999999
COL ENCRYPTED FORMAT A7
COL TAG       FORMAT A25

SELECT S.RECID AS "BS_REC", P.RECID AS "BP_REC", P.ENCRYPTED,
       P.TAG, P.HANDLE AS "MEDIA_HANDLE"
FROM   V$BACKUP_PIECE P, V$BACKUP_SET S
WHERE  P.SET_STAMP = S.SET_STAMP
AND    P.SET_COUNT = S.SET_COUNT;
```

The following sample output shows that the backups are encrypted:

```
BS_REC BP_REC ENCRYPT TAG
------ ------ ------- -------------------------
MEDIA_HANDLE
--------------------------------------------------------------------------------
     1      1 YES     TAG20070711T140124
/disk1/c-39525561-20070711-00

     2      2 YES     TAG20070711T140128
/disk1/c-39525561-20070711-01

     3      3 YES     TAG20070711T140130
/disk1/c-39525561-20070711-02
```

> **See Also:** *Oracle Database Reference* to learn about the V$BACKUP_PIECE view

Querying Recovery Catalog Views

The LIST, REPORT, and SHOW commands provide the easiest means of accessing the data in the control file and the recovery catalog. Nevertheless, you can sometimes also obtain useful information from the recovery catalog views, which reside in the recovery catalog schema and use the RC_ prefix.

About Recovery Catalog Views

RMAN obtains backup and recovery metadata from a target database control file and stores it in the tables of the recovery catalog. The recovery catalog views are derived from these tables. Note that the recovery catalog views are not normalized or optimized for user queries.

In general, the recovery catalog views are not as user-friendly as the RMAN reporting commands. For example, when you start RMAN and connect to a target database, you obtain the information for this target database only when you issue LIST, REPORT, and SHOW commands. If you have ten different target databases registered in the same recovery catalog, then any query of the catalog views show the metadata for all incarnations of all ten databases. You often have to perform complex selects and joins among the views to extract usable information about a database **incarnation**.

Most of the catalog views have a corresponding V$ view in the database. For example, RC_BACKUP_PIECE corresponds to V$BACKUP_PIECE. The primary difference between the recovery catalog view and corresponding V$ view is that each recovery catalog view contains metadata about *all* the target databases registered in the recovery catalog. The V$ view contains information only about itself.

> **See Also:** *Oracle Database Backup and Recovery Reference* for a description of recovery catalog views

Unique Identifiers for Registered Databases

Most recovery catalog views contain the columns DB_KEY and DBINC_KEY. Each database registered in the recovery catalog can be uniquely identified by either the primary key, which is the DB_KEY column value, or the DBID, which is the 32-bit unique database identifier. Each incarnation of a database is uniquely identified by the DBINC_KEY column.

You can use DB_KEY and DBINC_KEY to retrieve the records of a specific incarnation of a target database. Then, you can perform joins with most of the other catalog views to isolate records belonging to this incarnation.

An important difference between catalog and V$ views is that a different system of unique identifiers is used for backup and recovery files. For example, many V$ views such as V$ARCHIVED_LOG use the RECID and STAMP columns to form a concatenated primary key. The corresponding recovery catalog view uses a derived value as its primary keys and stores this value in a single column. For example, the primary key in RC_ARCHIVED_LOG is the AL_KEY column. The AL_KEY column value is the primary key that RMAN displays in the LIST command output.

Unique Identifiers in a Data Guard Environment

Special considerations apply when querying the recovery catalog in a Data Guard environment. In a Data Guard environment, multiple databases share the same DBID. Several views contain a DB_UNIQUE_NAME column, which indicates the DB_UNIQUE_NAME of the database incarnation to which the record belongs. All databases in a Data Guard environment share the same DBID but different DB_UNIQUE_NAME values.

The value of DB_UNIQUE_NAME is null when the database name is not known to the catalog, as for Oracle9*i* databases that are registered in a recovery catalog. Also, the column value is null when a database is upgraded to Oracle Database 11*g*, but the recovery catalog schema has not reconciled the database names for all files.

In the recovery catalog views, the primary database and its associated standby databases share the same DB_KEY. However, every database in a Data Guard environment has a unique RC_SITE.SITE_KEY value. For example, a primary database prod and its standby database standby1 might both have the DB_KEY value of 1, while the SITE_KEY of prod is 3 and the SITE_KEY of standby1 is 30.

Some recovery catalog views do not have a DB_UNIQUE_NAME column, but include a SITE_KEY column. You can use the SITE_KEY column to join with RC_SITE.SITE_KEY to determine the DB_UNIQUE_NAME of the database associated with a file. As explained in "RMAN File Management in a Data Guard Environment" on page 3-8, every file in a Data Guard environment is associated with the primary or standby database that created it.

> **See Also:** *Oracle Data Guard Concepts and Administration* to learn how to report on and manage files in a Data Guard environment

Querying Catalog Views for the Target DB_KEY or DBID Values

The DB_KEY value, which is the primary key for a registered database, is used only in the recovery catalog. The easiest way is to obtain the DB_KEY is to use the DBID of the target database, which is displayed whenever you connect RMAN to a database as TARGET. The DBID distinguishes databases registered in the RMAN recovery catalog.

Assume that you want to obtain information about one of the databases registered in the recovery catalog.

To query the catalog for information about the current incarnation of a database:

1. Determine the DBID for the database whose records you want to view.

   You can obtain the DBID by looking at the output displayed when RMAN connects to the database, querying `V$RMAN_OUTPUT`, or querying a `V$DATABASE` view. The following example connects SQL*Plus to the desired database and queries the DBID:

   ```
   SQL> CONNECT / AS SYSDBA
   SQL> SELECT DBID
     2  FROM   V$DATABASE;

   DBID
   ---------
   598368217
   ```

2. Start SQL*Plus and connect to the recovery catalog database as the owner of the recovery catalog.

3. Obtain the database key for the database whose DBID you obtained in step 1.

   You can then obtain the `DB_KEY` for a database by running the following query, where *`dbid_of_target`* is the DBID obtained in step 1:

   ```
   SELECT DB_KEY
   FROM   RC_DATABASE
   WHERE  DBID = dbid_of_target;
   ```

4. Query the records for the current incarnation of the database whose DBID you obtained in step 1.

   To obtain information about the current incarnation of a target database, specify the target database `DB_KEY` value and perform a join with `RC_DATABASE_INCARNATION`. Use a `WHERE` condition to specify that the `CURRENT_INCARNATION` column value is set to `YES`. For example, to obtain information about backup sets in the current incarnation of a target database with the `DB_KEY` value of `1`, query as follows:

   ```
   SELECT BS_KEY, BACKUP_TYPE, COMPLETION_TIME
   FROM   RC_DATABASE_INCARNATION i, RC_BACKUP_SET b
   WHERE  i.DB_KEY = 1
   AND    i.DB_KEY = b.DB_KEY
   AND    i.CURRENT_INCARNATION = 'YES';
   ```

See also:

- *Oracle Database Backup and Recovery Reference* for details about the `RC_DATABASE_INCARNATION` view
- "Database Incarnations" on page 13-5

Querying RC_BACKUP_FILES

You can query the view `RC_BACKUP_FILES` for information about all backups of any database registered in the recovery catalog. Before querying `RC_BACKUP_FILES`, however, you must call `DBMS_RCVMAN.SETDATABASE`. Specify the DBID of one of the databases registered in the recovery catalog, as shown in the following example:

```
SQL> CALL DBMS_RCVMAN.SETDATABASE(null,null,null,2283997583);
```

The fourth parameter must be the DBID of a database registered in the recovery catalog. The other parameters must all be `NULL`.

See Also:

- *Oracle Database Backup and Recovery Reference* for details about the `RC_BACKUP_FILES` view
- "Determining the DBID of the Database" on page 17-5 for techniques for determining the DBID of a database

11

Maintaining RMAN Backups and Repository Records

This chapter describes how to manage the RMAN repository records as well as RMAN backups and copies. This chapter also explains maintenance tasks related to the flash recovery area. This chapter contains the following topics:

- Overview of RMAN Backup and Repository Maintenance
- Maintaining the Control File Repository
- Maintaining the Flash Recovery Area
- Updating the RMAN Repository
- Deleting RMAN Backups and Archived Redo Logs
- Dropping a Database

> **See Also:** Chapter 12, "Managing a Recovery Catalog" for RMAN maintenance issues that are specific to a **recovery catalog**

Overview of RMAN Backup and Repository Maintenance

This section explains the purpose and basic concepts of RMAN repository maintenance.

Purpose of Backup and Repository Maintenance

The recommended maintenance strategy is to configure a **flash recovery area**, a **backup retention policy**, and an **archived redo log deletion policy**. In this case, the database automatically maintains and deletes backups and archived redo logs as needed. However, manual maintenance of database backups and archived redo logs is sometimes necessary.

Managing RMAN backups involves the following related tasks:

- Managing the database backups that are stored on disk or tape
- Managing the records of those backups in the RMAN repository

An important part of RMAN maintenance is deleting backups that are no longer needed. If you configure a **flash recovery area**, then the database automatically deletes unneeded files in this area automatically; even so, you may want to delete backups and copies from tape. You may even need to delete an entire database. You can use an RMAN command to perform these tasks.

The flash recovery area may require occasional maintenance. For example, the flash recovery area may become full, in which case you may need to add space to it. Alternatively, you may want to change the location of the recovery area.

It is possible for the **RMAN repository** to fail to reflect the true state of files on disk or tape. For example, a user may delete a backup from disk with an operating system utility. In this case, the RMAN repository shows that the file exists when it does not. In a similar situation, a tape containing RMAN backups may be corrupted. You can use **RMAN maintenance commands** to update the repository with true information.

Basic Concepts of Backup and Repository Maintenance

The RMAN maintenance commands are summarized as follows:

- The `CATALOG` command enables you to add records about RMAN and user-managed backups that are currently not recorded in the RMAN repository, or to remove records for backups that are recorded.
- The `CHANGE` command enables you to update the status of records in the RMAN repository.
- The `CROSSCHECK` command enables you to synchronize the logical backup records with the physical reality of files in backup storage.
- The `DELETE` command enables you to delete backups from the operating system.

Maintenance Commands and RMAN Repository Metadata

RMAN always stores its metadata in the control file of each target database on which it performs operations. If you register a target database in the recovery catalog, then RMAN stores the metadata for this target database in the recovery catalog. All of the RMAN maintenance commands work with or without a recovery catalog.

See Also: "Maintaining a Recovery Catalog" on page 12-21

Maintenance Commands in a Data Guard Environment

As explained in "RMAN File Management in a Data Guard Environment" on page 3-8, the database in a Data Guard environment that creates a backup or copy is associated with the file. For example, if RMAN is connected to target database `standby1` and backs it up, then this backup is associated with `standby1`.

As long as backups are accessible to RMAN according to the criteria specified in "RMAN File Management in a Data Guard Environment" on page 3-8, you can use RMAN maintenance commands such as `CHANGE`, `DELETE`, and `CROSSCHECK` for backups when connected to any primary or standby database.

Crosschecks in a Data Guard Environment For a crosscheck, RMAN can only update the status of a file from `AVAILABLE` to `EXPIRED` when connected to the database associated with the file. Thus, if RMAN crosschecks a file and does not find it, and if the file is associated with a database to which it is not connected as `TARGET`, then RMAN prompts you to perform the crosscheck when connected to the target database associated with the file. Note that RMAN performs a crosscheck when you run the `CROSSCHECK` or `CHANGE ... AVAILABLE` command.

Deletion in a Data Guard Environment RMAN can delete files when connected to any database. If RMAN is not connected as `TARGET` to the database associated with a file, and if RMAN cannot delete a file successfully, then RMAN prompts you to connect as

TARGET to the database associated with the file. You must then use DELETE ... FORCE to delete the file metadata.

Updates to RMAN Metadata in a Data Guard Environment If a maintenance command changes RMAN metadata only, then you can connect RMAN as TARGET to any database in the Data Guard environment. Commands that change only metadata include:

- CHANGE ... UNAVAILABLE or CHANGE ... UNCATALOG
- CHANGE ... KEEP or CHANGE ... NOKEEP
- CHANGE ... RESET DB_UNIQUE_NAME

By default, the CHANGE command only operates on files that are accessible according to the rules specified in "Accessibility of Backups in a Data Guard Environment" on page 3-8. However, you can change the status of files associated with a database other than the target database by using the FOR DB_UNIQUE_NAME option.

Files Not Associated with a Database In some cases the DB_UNIQUE_NAME may not be known for a specific file. For example, the value of DB_UNIQUE_NAME is null when the database name is not known to the recovery catalog, as for Oracle9*i* databases that are registered in a recovery catalog. Also, rows can have a DB_UNIQUE_NAME of null because a database has been upgraded to the current version, but the recovery catalog schema has not reconciled the DB_UNIQUE_NAME values for all files. By default, RMAN associates files whose SITE_KEY is null with the database to which RMAN is connected as TARGET. A backup remains associated with a database unless you explicitly use the CHANGE ... RESET DB_UNIQUE_NAME to associate the backup with a different database.

See Also:

- *Oracle Data Guard Concepts and Administration* to learn how to use RMAN to back up and restore files in a Data Guard environment
- *Oracle Database Backup and Recovery Reference* for descriptions of the RMAN maintenance commands

Maintaining the Control File Repository

RMAN is designed to work without a recovery catalog. If you choose not to use a recovery catalog, however, then the control file of each target database is the exclusive repository for RMAN metadata. You should know how information is stored in the control file and ensure that your backup and recovery strategy protects the control file.

See Also: *Oracle Database Administrator's Guide* for an overview of the control file and more details about managing control files

About Control File Records

The control file contains two types of records: **circular reuse records** and **noncircular reuse records**.

Circular reuse records contain noncritical information that is eligible to be overwritten if needed. These records contain information that is continually generated by the database. When all available record slots are full, the database either expands the control file to make room for a new record or overwrites the oldest record. The CONTROL_FILE_RECORD_KEEP_TIME initialization parameter specifies the minimum age in days of a record before it can be reused.

Noncircular reuse records contain critical information that does not change often and cannot be overwritten. Some examples of information in noncircular reuse records include datafiles, online redo log files, and redo threads.

As you make backups of a target database, the database records these backups in the control file. To prevent the control file from growing too large because of the addition of new records, records can be reused if they are older than a threshold you specify. The `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter determines the minimum age in days of a record before it can be overwritten:

```
CONTROL_FILE_RECORD_KEEP_TIME = integer
```

For example, if the parameter value is `14`, then any record aged 14 days and older is a candidate for reuse. Information in an overwritten record is lost. The oldest record available for reuse will be used first.

When the database needs to add new RMAN repository records to the control file, but no record is older than the threshold, the database attempts to expand the size of the control file. If the underlying operating system prevents the expansion of the control file (due to a disk full condition, for instance), then the database overwrites the oldest record in the control file.

The database records the overwrite in the alert log located in the **Automatic Diagnostic Repository (ADR)**. For each record that it overwrites, the database records an entry in the alert log similar to the following:

```
kccwnc: following control file record written over:
RECID #72 Recno 72 Record timestamp
07/28/06 22:15:21
Thread=1 Seq#=3460
Backup set key: stamp=372031415, count=17
Low scn: 0x0000.3af33f36
07/27/06 21:00:08
Next scn: 0x0000.3af3871b
07/27/06 23:23:54
Resetlogs scn and time
scn: 0x0000.00000001
```

Flash Recovery Area and Control File Records

When a control file record containing information about a file created in the flash recovery area is about to be reused, the database attempts to delete the file from the flash recovery area when the file is eligible for deletion. Otherwise, the database expands the size of the control file section containing the record for this file. The database logs the expansion in the alert log with a message like this example, where *nnnn* is the number of the control file record type:

```
kccwnc: trying to expand control file section nnnn for Oracle Managed Files
```

If the control file is at the maximum size supported under the host operating system, then the database cannot expand the control file. In such a situation, this warning appears in the alert log:

```
WARNING: Oracle Managed File filename is unknown to control file. This is the
result of limitation in control file size that could not keep all recovery area
files.
```

The preceding means that the control file cannot hold a record of all flash recovery area files needed to satisfy the configured retention policy. The next section explains how to respond to this situation.

See Also: *Oracle Database Reference* for information about the CONTROL_FILE_RECORD_KEEP_TIME initialization parameter

Preventing the Loss of Control File Records

The best way to prevent the loss of RMAN metadata because of overwritten control file records is to use a recovery catalog. If you cannot use a recovery catalog, then you can take the following measures:

- Set the CONTROL_FILE_RECORD_KEEP_TIME value to slightly longer than the oldest file that you need to keep. For example, if you back up the whole database once a week, then you need to keep every backup for at least seven days. Set CONTROL_FILE_RECORD_KEEP_TIME to a value such as 10 or 14. The default value of CONTROL_FILE_RECORD_KEEP_TIME is 7 days.

  Caution: Regardless of whether you use a recovery catalog, never use RMAN when CONTROL_FILE_RECORD_KEEP_TIME is set to 0. If you do, then you may lose backup records.

- Store the control file in a file system rather than **raw device** so that it can expand.
- Monitor the alert log to ensure that Oracle is not overwriting control file records. The alert log is located in the **Automatic Diagnostic Repository (ADR)**.

If you use a flash recovery area, then follow these guidelines to avoid a situation in which the control file cannot hold a record of all flash recovery area files needed to satisfy the backup retention policy:

- If the block size of the control file is not already at its maximum, then use a larger block size, preferably 32 KB.

  To achieve this aim, you must set the SYSTEM tablespace block size to be greater than or equal to the control file block size, and re-create the control file after changing DB_BLOCK_SIZE. Note that the files in the flash recovery area will be recataloged, but the records for files on tape will be lost.

- Make the files in the flash recovery area eligible for deletion by backing them up to tertiary storage such as tape.

  For example, you can use BACKUP RECOVERY AREA to specifically back up files in the flash recovery area to a media manager.

- If the backup retention policy is keeping backups and archived logs longer than your business requirements, then you can make more files in the flash recovery area eligible for deletion by changing the retention policy to a shorter recovery window or lower degree of redundancy.

Protecting the Control File

If you are not using a recovery catalog to store RMAN metadata, then it is doubly important that you protect each target database control file. You can use the following strategy to protect the control file.

To protect the control file:

1. Create redundant copies of control files through **multiplexing** or operating system **mirroring**.

In this way, the database can survive the loss of a subset of the control files without requiring you to restore a control file from backup. It is recommended that you use a minimum of two multiplexed or mirrored control files on separate disks.

2. Configure the **control file autobackup** feature.

   In this case, RMAN automatically backs up the control file when you run certain RMAN commands. As long as a control file autobackup is available, RMAN can restore the server parameter and backup control file, and mount the database. After the control file is mounted, you can restore the remainder of the database.

3. Keep a record of the database DBID.

   If you lose the control files, then you may need the DBID to recover the database.

See Also:

- "Backing Up Control Files with RMAN" on page 8-8 to learn about manual and automatic control file backups
- "Control File and Server Parameter File Autobackups" on page 7-12

Maintaining the Flash Recovery Area

While the flash recovery area is largely self-managing, some situations may require DBA intervention.

Deletion Rules for the Flash Recovery Area

"Overview of the Flash Recovery Area" on page 5-14 explains the contents of the flash recovery area and the difference between permanent and transient files. Review this section before proceeding. The following rules govern when files become eligible for deletion from the recovery area:

- Permanent files are never eligible for deletion.
- Files that are obsolete under the retention policy are eligible for deletion.

  "Configuring the Backup Retention Policy" on page 5-21 explains how to configure the retention policy.
- Transient files that have been copied to tape are eligible for deletion.
- Archived redo logs are not eligible for deletion until all the consumers of the logs have satisfied their requirements.

  "Configuring an Archived Redo Log Deletion Policy" on page 5-26 explains how to configure an **archived redo log deletion policy** that determines when logs are eligible to be deleted. Consumers of logs can include RMAN, standby databases, Oracle Streams databases, and the Flashback Database feature. See *Oracle Data Guard Concepts and Administration* to learn about archived redo log management in a Data Guard environment.
- Foreign archived logs that have been mined by a LogMiner session on a logical standby database are eligible for deletion. Unlike an ordinary archived redo log, a **foreign archived redo log** has a different DBID.

The safe and reliable way to control deletion of files from the flash recovery area is to configure your retention policy ("Configuring the Backup Retention Policy" on page 5-21) and archived log deletion policy ("Configuring an Archived Redo Log

Deletion Policy" on page 5-26). To increase the likelihood that files moved to tape are retained on disk, increase the flash recovery area quota.

Monitoring Flash Recovery Area Space Usage

You can use the V$RECOVERY_FILE_DEST and V$FLASH_RECOVERY_AREA_USAGE views to determine whether you have allocated enough space for your flash recovery area. Query the V$RECOVERY_FILE_DEST view to find out the current location, disk quota, space in use, space reclaimable by deleting files, and total number of files in the flash recovery area. For example, enter the query shown in Example 11–1 (sample output included). Note that the space columns specify the amount in bytes.

Example 11–1 Flash Recovery Area Space Consumption

```
SELECT * FROM V$RECOVERY_FILE_DEST;

NAME            SPACE_LIMIT SPACE_USED SPACE_RECLAIMABLE NUMBER_OF_FILES
--------------  ----------- ---------- ----------------- ---------------
/mydisk/rcva     5368709120 109240320             256000              28
```

Query the V$FLASH_RECOVERY_AREA_USAGE view to find out the percentage of the total disk quota used by different types of files. Also, you can determine how much space for each type of file can be reclaimed by deleting files that are obsolete, redundant, or already backed up to tape. For example, enter the following query (sample output included):

```
SELECT * FROM   V$FLASH_RECOVERY_AREA_USAGE;

FILE_TYPE    PERCENT_SPACE_USED PERCENT_SPACE_RECLAIMABLE NUMBER_OF_FILES
------------ ------------------ ------------------------- ---------------
CONTROLFILE                   0                         0               0
ONLINELOG                     2                         0              22
ARCHIVELOG                 4.05                      2.01              31
BACKUPPIECE                3.94                      3.86               8
IMAGECOPY                 15.64                     10.43              66
FLASHBACKLOG                .08                         0               1
```

When guaranteed restore points are defined on your database, you should monitor the amount of space used in your flash recovery area for files required to meet the guarantee. Use the query for viewing guaranteed restore points in "Listing Restore Points" on page 10-9 and refer to the STORAGE_SIZE column to determine the space required for files related to each guaranteed restore point.

> **See Also:** *Oracle Database Reference* for more details on the V$RECOVERY_FILE_DEST and V$FLASH_RECOVERY_AREA views.

Managing Space For Flashback Logs in the Flash Recovery Area

"Configuring Oracle Flashback Database and Restore Points" on page 5-27 explains the rules for flashback log deletion. You cannot manage the flashback logs in the flash recovery area directly other than by setting the flashback retention target or using guaranteed restore points. Nevertheless, you can manage flash recovery area space as a whole in order to maximize space available for retention of flashback logs. In this way you increase the likelihood of achieving the flashback target.

To make space for flashback logs, back up the other contents of your flash recovery area to tape with commands such as BACKUP RECOVERY AREA, BACKUP BACKUPSET, and so on. Oracle Database automatically removes obsolete files from the flash

recovery area. If offloading backups to tape still does not create enough space to satisfy the backup retention policy and flashback retention target, then allocate more space in the flash recovery area.

> **Note:** You cannot back up flashback logs. Thus, `BACKUP RECOVERY AREA` does not include the flashback logs when backing up the flash recovery area contents to tape.

Responding to a Full Flash Recovery Area

If the RMAN retention policy requires keeping a set of backups larger than the flash recovery area disk quota, or if the retention policy is set to `NONE`, then the flash recovery area can fill completely with no reclaimable space.

The database issues a warning alert when reclaimable space is less than 15% and a critical alert when reclaimable space is less than 3%. To warn the DBA of this condition, an entry is added to the alert log and to the `DBA_OUTSTANDING_ALERTS` table (used by Enterprise Manager). Nevertheless, the database continues to consume space in the flash recovery area until there is no reclaimable space left.

When the recovery area is completely full, the error you will receive is as follows, where *nnnnn* is the number of bytes required and *mmmm* is the disk quota:

```
ORA-19809: limit exceeded for recovery files
ORA-19804: cannot reclaim nnnnn bytes disk space from mmmmm limit
```

You have a number of choices on how to resolve a full flash recovery area when no files are eligible for deletion:

- Make more disk space available and increase `DB_RECOVERY_FILE_DEST_SIZE` to reflect the additional space.
- Move backups from the flash recovery area to tertiary storage such as tape.

  One convenient way to back up all of your recovery area files to tape at once is the `BACKUP RECOVERY AREA` command. After you transfer backups from the recovery area to tape, you can delete files from the flash recovery area (see "Deleting RMAN Backups and Archived Redo Logs" on page 11-19). Note that flashback logs cannot be backed up outside the recovery area and so are not backed up by `BACKUP RECOVERY AREA`.
- Run `DELETE` for any files that have been removed with an operating system utility.

  If you use host operating system commands to delete files, then the database will not be aware of the resulting free space. You can run the RMAN `CROSSCHECK` command to have RMAN re-check the contents of the flash recovery area and identify expired files, and then use the `DELETE EXPIRED` command to every **expired backup** from the RMAN repository.
- Make sure that your guaranteed restore points are necessary. If not, delete them as described in "Dropping Restore Points" on page 11-9.

  Flashback logs that are not needed for a guaranteed restore point are deleted automatically to gain space for other files in the flash recovery area. A guaranteed restore point forces the retention of flashback logs required to perform Flashback Database to the restore point SCN.
- Review your backup retention policy and, if using Data Guard, your archived redo log deletion policy.

> **See Also:** Chapter 8, "Backing Up the Database" to decide on a retention policy, and *Oracle Data Guard Concepts and Administration* for more on archived log deletion policy with Data Guard

Dropping Restore Points

When you are satisfied that you do not need an existing restore point, or when you want to create a new restore point with the name of an existing restore point, you can drop the restore point, using the `DROP RESTORE POINT` SQL statement. For example:

```
DROP RESTORE POINT before_app_upgrade;
```

The same statement is used to drop both normal and guaranteed restore points.

Normal restore points eventually age out of the control file, even if not explicitly dropped. The rules governing retention of restore points in the control file are:

- The most recent 2048 restore points are always kept in the control file, regardless of their age.
- Any restore point more recent than the value of `CONTROL_FILE_RECORD_KEEP_TIME` is retained, regardless of how many restore points are defined.

Normal restore points that do not meet either of the preceding conditions may age out of the control file. Guaranteed restore points never age out of the control file. They remain until they are explicitly dropped.

> **See also:** *Oracle Database SQL Language Reference* for reference information about the SQL `DROP RESTORE POINT` statement

Changing the Flash Recovery Area to a New Location

If you need to move the flash recovery area of your database to a new location, then follow this procedure:

1. Start a SQL*Plus on the target database and change the `DB_RECOVERY_FILE_DEST` initialization parameter. For example, enter the following command to set the destination to the ASM disk group `disk1`:

   ```
   ALTER SYSTEM SET DB_RECOVERY_FILE_DEST='+disk1' SCOPE=BOTH SID='*';
   ```

   After you change this parameter, all new flash recovery area files will be created in the new location.

2. Either leave or move the permanent files, flashback logs, and transient files in the old flash recovery location.

   If you leave the existing files in the flash recovery, then the database deletes the transient files from the old flash recovery area as they become eligible for deletion.

   If you need to move the old files to the new flash recovery area, then see Chapter 26, "Performing ASM Data Migration". The procedure for moving database files into and out of an ASM disk group with RMAN works when moving files into and out of a flash recovery area.

Disabling the Flash Recovery Area

Before disabling the flash recovery area, you must first drop all guaranteed restore points and then turn off the Flashback Database. Once these prerequisites have been met, you can disable the Flash Recovery Area by setting the `DB_RECOVERY_FILE_`

DEST initialization parameter to a null string. For example, use the following SQL statement to change the parameter on a running database:

```
ALTER SYSTEM SET DB_RECOVERY_FILE_DEST='' SCOPE=BOTH SID='*';
```

The database will no longer provide the space management features of the flash recovery area for the files stored in the old DB_RECOVERY_FILE_DEST location. The files will still be known to the RMAN repository, however, and available for backup and restore activities.

Responding to an Instance Crash During File Creation

As a rule, the flash recovery area is self-maintaining. When an instance crashes during the creation of a file in the flash recovery area, however, the database may leave the file in the flash recovery area. When this situation occurs, you will see the following error in the alert log, where *location* is the location of the flash recovery area:

```
ORA-19816: WARNING: Files may exist in location that are not known to database.
```

In such a situation, use the RMAN command CATALOG RECOVERY AREA to recatalog any such files. If the file header of the file in question is corrupted, then delete the file manually with an operating system utility.

Monitoring Flashback Database Performance Impact

The Automatic Workload Repository (AWR) automates database statistics gathering by collecting, processing, and maintaining performance statistics for database problem detection and self-tuning. There are several data analysis methods for monitoring the Flashback Database workload on your system. For example, you can compare AWR reports from before and after the Flashback Database was turned on. You can also review AWR snapshots to pinpoint system usage caused by flashback logging. For example, if flashback buf free by RVWR is the top wait event, then you know that Oracle Database cannot write flashback logs very quickly. In such a case, you may want to tune the file system and storage used by the flash recovery area, possibly using one of the techniques described in "Configuring the Environment for Optimal Flashback Database Performance" on page 5-34.

The V$FLASHBACK_DATABASE_STAT view shows the bytes of flashback data logged by the database. Each row in the view shows the statistics accumulated (typically over the course of an hour). The FLASHBACK_DATA and REDO_DATA columns describe bytes of flashback data and redo data written respectively during the time interval, while the DB_DATA column describe bytes of data blocks read and written. Note that FLASHBACK_DATA and REDO_DATA correspond to sequential writes, while DB_DATA corresponds to random reads and writes.

Because of the difference between sequential I/O and random I/O, a better indication of I/O overhead is the number of I/O operations issued for flashback logs. The V$SYSSTAT statistics shown in Table 11–1 can tell you the number of I/O operations your instance has issued for various purposes.

Table 11–1 V$SYSSTAT Statistics

Column Name	Column Meaning
Physical write I/O request	The number of write operations issued for writing data blocks
physical read I/O request	The number of read operations issued for reading data blocks
redo writes	The number of write operations issued for writing to the redo log.

Table 11–1 (Cont.) V$SYSSTAT Statistics

Column Name	Column Meaning
flashback log writes	The number of write operations issued for writing to flashback logs.

See Also:

- *Oracle Database Reference* for more details on columns in the `V$SYSSTAT` view
- *Oracle Database Performance Tuning Guide* to learn about the Automatic Workload Repository
- *Oracle Database 2 Day + Performance Tuning Guide* for more information on AWR reports

Flashback Writer (RVWR) Behavior With I/O Errors

When flashback is enabled or when there are guaranteed restore points, the background process RVWR writes flashback data to flashback database logs in the flash recovery area. If RVWR encounters an I/O error, the following behavior is expected:

- If there are any guaranteed restore points defined, then the instance will fail when RVWR encounters I/O errors.
- If no guaranteed restore points are defined, then the instance will not fail when RVWR encounters I/O errors. Note the following cases:
  - On a primary database, Oracle Database automatically disables Flashback Database while the database is open. All existing transactions and queries will proceed unaffected. This behavior is expected for both single-instance and Oracle RAC databases.
  - On a physical or logical standby, RVWR will appear to be hung, retrying the I/O periodically. This may eventually hang the logical standby or the managed recovery of the physical standby. (Oracle Database does not cause the standby instance to fail because it does not want to cause the primary database to fail in turn in max protection mode.) To resolve the hang, you can either issue `SHUTDOWN ABORT` or `ALTER DATABASE FLASHBACK OFF`.

Updating the RMAN Repository

This section explains how to make sure that the RMAN repository accurately reflects the reality of the RMAN-related files stored on disk and tape. Several situations can cause a discrepancy between the repository and the files that it records, including tape or disk failures and user-managed copies or deletions of RMAN-related files.

This section contains the following topics:

- Crosschecking the RMAN Repository
- Changing the Repository Status of Backups and Copies
- Adding Backup Records to the RMAN Repository
- Removing Records from the RMAN Repository

Crosschecking the RMAN Repository

To ensure that data about backups in the recovery catalog or control file is synchronized with corresponding data on disk or in the media management catalog, perform a **crosscheck**. The CROSSCHECK command operates only on files that are currently recorded in the RMAN repository.

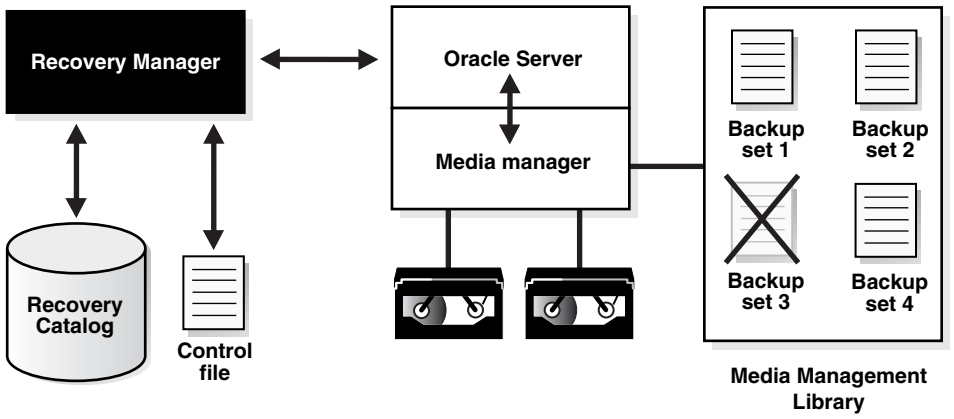
If you use a flash recovery area, backup retention policy, and archived redo log deletion policy, then you should not need to perform crosschecks very often. If you delete files by means other than RMAN, then you should perform a crosscheck periodically to make sure that the repository data stays current.

About RMAN Crosschecks

Crosschecks update outdated RMAN repository information about backups whose repository records do not match their physical status. For example, if a user removes archived logs from disk with an operating system command, the repository still indicates that the logs are on disk, when in fact they are not.

Figure 11–1 illustrates a crosscheck of a media manager. RMAN queries the RMAN repository for the names and locations of the four backup sets to be checked. RMAN sends this information to the target database server, which queries the media management software about the backups. The media management software then checks its media catalog and reports back to the server that backup set 3 is missing. RMAN updates the status of backup set 3 to EXPIRED in the repository. The record for backup set 3 will now be deleted if you run DELETE EXPIRED.

Figure 11–1 Crosschecking a Media Manager



Crosschecks are useful because they can do the following:

- Update outdated information about backups that disappeared from disk or tape or became corrupted
- Update the repository if you delete archived redo logs or other files with operating system commands

Use the crosscheck feature to check the status of a backup on disk or tape. If the backup is on disk, then CROSSCHECK checks whether the header of the file is valid. If a backup is on tape, then the command checks that the backups exist in the media management software catalog.

Backup pieces and image copies can have the status AVAILABLE, EXPIRED, or UNAVAILABLE. You can view the status of backups by running the RMAN LIST command or by querying V$BACKUP_FILES or recovery catalog views such as RC_

DATAFILE_COPY or RC_ARCHIVED_LOG. A crosscheck updates the RMAN repository so that all of these techniques provide accurate information. RMAN updates each backup in the RMAN to status EXPIRED if the backup is no longer available. If a new crosscheck determines that an **expired backup** is available again, then RMAN updates its status to AVAILABLE.

> **Note:** The CROSSCHECK command does *not* delete operating system files or remove repository records. You must use the DELETE command for these operations.

You can issue the DELETE EXPIRED command to delete all expired backups. RMAN removes the record for the expired file from the repository. If for some reason the file still exists on the media, then RMAN issues warnings and lists the mismatched objects that cannot be deleted.

See Also:

- *Oracle Database Backup and Recovery Reference* for CROSSCHECK syntax and a description of the possible status values
- *Oracle Database Backup and Recovery Reference* for DELETE syntax

Crosschecking All Backups and Copies

After connecting to the target database and recovery catalog (if you use one), run CROSSCHECK commands as needed to verify the status and availability of backups known to RMAN.

You can configure or manually allocate multiple channels before issuing CROSSCHECK or DELETE commands. RMAN searches for each backup on all channels that have the same device type as the channel used to create the backup. The multichannel feature is primarily intended for use when crosschecking or deleting backups on both disk and tape within a single command. For example, assume that you have an **SBT** channel configured as follows:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE sbt;
```

In this case you can run the following commands to crosscheck both disk and SBT:

```
CROSSCHECK BACKUP;
CROSSCHECK COPY;
```

RMAN uses both the SBT channel and the preconfigured disk channel to perform the crosscheck. Sample output follows:

```
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: sid=12 devtype=SBT_TAPE
channel ORA_SBT_TAPE_1: WARNING: Oracle Test Disk API
using channel ORA_DISK_1
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=/oracle/dbs/16c5esv4_1_1 recid=36 stamp=408384484
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=/oracle/dbs/c-674966176-20000915-01 recid=37 stamp=408384496
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=12c5erb2_1_1 recid=32 stamp=408382820
      .
      .
      .
```

If you do not have an automatic SBT channel configured, then can also manually allocate maintenance channels on disk and tape as in the following example:

```
RUN
{
  ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
  CROSSCHECK BACKUP;
  CROSSCHECK COPY;
}
```

You do not have to manually allocate a disk channel because RMAN uses the preconfigured disk channel.

Crosschecking Specific Backup Sets and Copies

You can use the `LIST` command to report your backups and then use the `CROSSCHECK` command to check that specific backups described in the `LIST` output still exist. The `DELETE EXPIRED` command deletes repository records for backups that fail the crosscheck.

To crosscheck specified backups:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Run a `LIST` command to identify the backups to be checked.

   For example, run the following command:

   ```
   LIST BACKUP;  # lists all backup sets, proxy copies, and image copies
   ```

3. Crosscheck the desired backups or copies.

   The following sample commands illustrate different types of crosschecks:

   ```
   CROSSCHECK BACKUP;  # checks backup sets, proxy copies, and image copies
   CROSSCHECK COPY OF DATABASE;
   CROSSCHECK BACKUPSET 1338, 1339, 1340;
   CROSSCHECK BACKUPPIECE TAG 'nightly_backup';
   CROSSCHECK BACKUP OF ARCHIVELOG ALL SPFILE;
   CROSSCHECK BACKUP OF DATAFILE "?/oradata/trgt/system01.dbf"
     COMPLETED AFTER 'SYSDATE-14';
   CROSSCHECK CONTROLFILECOPY '/tmp/control01.ctl';
   CROSSCHECK DATAFILECOPY 113, 114, 115;
   CROSSCHECK PROXY 789;
   ```

   See Also: *Oracle Database Backup and Recovery Reference* for more details on using `CROSSCHECK` to check backups of specific files

Changing the Repository Status of Backups and Copies

This section explains how to change the repository records for backups and copies. You may need to change the status of a backup if it becomes temporarily available or unavailable. For example, if a mounted disk undergoes maintenance, then you can update the records for backups on the disk to status `UNAVAILABLE`.

Updating a Backup to Status AVAILABLE or UNAVAILABLE

Run the `CHANGE . . . UNAVAILABLE` command when a backup cannot be found or has migrated offsite. RMAN does not use files with status `UNAVAILABLE` in `RESTORE` or `RECOVER` commands. If the file is later found or returns to the main site, then you can

update its status again by issuing CHANGE . . . AVAILABLE. Note that files in the flash recovery area cannot be marked as UNAVAILABLE.

To update the status of a file in the repository to UNAVAILABLE or AVAILABLE:

1. Issue a LIST command to determine the availability status of RMAN backups. For example, issue the following command to list all backups:

   ```
   LIST BACKUP;
   ```

2. Run CHANGE with the UNAVAILABLE or AVAILABLE keyword to update its status in the RMAN repository.

   The following examples illustrate forms of the CHANGE command:

   ```
   CHANGE DATAFILECOPY '/tmp/control01.ctl' UNAVAILABLE;
   CHANGE COPY OF ARCHIVELOG SEQUENCE BETWEEN 1000 AND 1012 UNAVAILABLE;
   CHANGE BACKUPSET 12 UNAVAILABLE;
   CHANGE BACKUP OF SPFILE TAG "TAG20020208T154556" UNAVAILABLE;
   CHANGE DATAFILECOPY '/tmp/system01.dbf' AVAILABLE;
   CHANGE BACKUPSET 12 AVAILABLE;
   CHANGE BACKUP OF SPFILE TAG "TAG20020208T154556" AVAILABLE;
   ```

   See Also: *Oracle Database Backup and Recovery Reference* for CHANGE command syntax

Changing the Status of an Archival Backup

As explained in "Making Database Backups for Long-Term Storage" on page 8-23, you can designate backups as exempt from the retention policy. This technique is useful for archiving backups to comply with business requirements. An **archival backup** is still a fully valid backup, however, and can be restored just as any other RMAN backup.

> **Note:** The KEEP FOREVER clause requires the use of a recovery catalog, because the control file cannot contain an infinitely large set of RMAN repository data.

You can use the CHANGE command to alter the KEEP status of an existing backup. For example, you may decide that you no longer want to keep a long-term backup. The same options available for BACKUP . . . KEEP are available with CHANGE . . . KEEP.

Note that you cannot set KEEP attributes for backup sets or files stored in the **flash recovery area**.

To alter the KEEP status of an archival backup:

1. Issue a LIST command to list the backups. For example, issue the following command to list all backups:

   ```
   LIST BACKUP;
   ```

2. Issue CHANGE . . . KEEP to define a different retention period for this backup, or CHANGE . . . NOKEEP to let the retention policy apply to this file.

   This example allows a backup set to be subject to the backup retention policy:

   ```
   CHANGE BACKUPSET 231 NOKEEP;
   ```

   This example makes a datafile copy exempt from the retention policy for 180 days:

   ```
   CHANGE DATAFILECOPY '/tmp/system01.dbf' KEEP UNTIL 'SYSDATE+180';
   ```

Adding Backup Records to the RMAN Repository

You can use the CATALOG command to make RMAN aware of the existence of archived logs not recorded in the repository or copies of database files that are created through means other than RMAN. This section contains the following topics:

- About Cataloging Operations
- Cataloging User-Managed Datafile Copies
- Cataloging Backup Pieces
- Cataloging All Files in a Disk Location

About Cataloging Operations

The target database control file keeps records of all archived redo logs generated by the target database as well as all RMAN backups. The purpose of the CATALOG command is to add metadata to the repository when it does not have a record of files that you want RMAN to know about.

Run the RMAN CATALOG command when:

- You use an operating system utility to make copies of datafiles, archived logs, or backup pieces. In the case, the repository has no record of them.
- You perform recovery with a backup control file and you change the archiving destination or format during recovery. In this situation, the repository will not have information about archived logs needed for recovery. Hence, you must catalog these logs.
- You want to catalog datafile copy as a level 0 backup, thus enabling you to perform an incremental backup later by using the datafile copy as the base of an incremental backup strategy
- You want to catalog user-managed copies of Oracle7 database files created before you migrated to a higher release, or of Oracle8 and higher database files created before you started to use RMAN. These datafile copies enable you to recover the database if it crashes after migration but before you have a chance to take a backup of the migrated database.

Whenever you make a user-managed copy, for example, by using the UNIX cp command to copy a datafile, make sure to catalog it. When making user-managed copies, you can use the ALTER TABLESPACE . . . BEGIN/END BACKUP statement to make datafile copies off an online tablespace. Although RMAN does not create such datafile copies, you can use the CATALOG command to add them to the recovery catalog so that RMAN is aware of them.

For a user-managed copy to be cataloged, it must be:

- Accessible on disk
- A complete image copy of a single file
- Either a datafile copy, control file copy, archived redo log copy, or backup piece copy

For example, if you store datafiles on mirrored disk drives, then you can create a user-managed copy by breaking the mirror. In this scenario, use the CATALOG command to notify RMAN of the existence of the user-managed copy after breaking the mirror. Before reforming the mirror, run a CHANGE . . . UNCATALOG command to notify RMAN that the file copy no longer exists.

Cataloging User-Managed Datafile Copies

Use the CATALOG command to propagate information about user-managed copies to the RMAN repository. After the files are cataloged, you can run LIST or query V$BACKUP_FILES to confirm.

To create and catalog a user-managed copy of a datafile:

1. Make a datafile copy with an operating system utility. ALTER TABLESPACE BEGIN/END BACKUP is necessary if the database is open and the datafiles are online while the backup is in progress. This example backs up an online datafile, using the SQL*Plus HOST command to issue an operating system command.

   ```
   SQL> ALTER TABLESPACE users BEGIN BACKUP;
   SQL> host cp $ORACLE_HOME/oradata/trgt/users01.dbf /tmp/users01.dbf;
   SQL> ALTER TABLESPACE users END BACKUP;
   ```

2. Start RMAN and connect to a target database and recovery catalog (if used).
3. Run the CATALOG command.

   For example, enter the following command to catalog a user-managed datafile copy:

   ```
   CATALOG DATAFILECOPY '/tmp/users01.dbf';
   ```

   If you try to catalog a datafile copy from a database other than the connected target database, then RMAN issues an error such as the following:

   ```
   RMAN-00571: ===========================================================
   RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
   RMAN-00571: ===========================================================
   RMAN-03009: failure of catalog command on default channel at 08/29/2007
   14:44:34
   ORA-19563: datafile copy header validation failed for file /tmp/tools01.dbf
   ```

   See Also: *Oracle Database Backup and Recovery Reference* for CATALOG command syntax

Cataloging Backup Pieces

You can catalog backup pieces on disk. This technique is useful if you use an operating system utility to copy backup pieces from location to another on the same host, or from one host to another. You can even catalog a backup piece from a prior **incarnation** of the database. RMAN can determine whether that backup piece can be used during a subsequent restore and recovery operation.

To catalog a backup piece:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Catalog the filenames of the backup pieces.

   For example, enter the following command:

   ```
   CATALOG BACKUPPIECE '/disk2/09dtq55d_1_2', '/disk2/0bdtqdou_1_1';
   ```

3. Optionally, run a LIST command or query V$ views to verify your changes.

   Views include V$BACKUP_PIECE, V$BACKUP_SET, V$BACKUP_DATAFILE, V$BACKUP_REDOLOG, and V$BACKUP_SPFILE. The following query shows the names of backup pieces:

   ```
   SELECT HANDLE
   ```

```
FROM   V$BACKUP_PIECE;
```

> **See Also:** *Oracle Database Backup and Recovery Reference* for `CATALOG BACKUPPIECE` restrictions

Cataloging All Files in a Disk Location

If you use Automatic Storage Management (ASM), an Oracle Managed Files framework, or the flash recovery area, then you may need a way to recatalog files that are known to the disk management system but are no longer listed in the RMAN repository. This situation can occur when the intended mechanisms for tracking filenames fails due to media failure, software bug, or user error.

The `CATALOG START WITH` command enables you to search through all files in an ASM disk group, Oracle Managed Files location, or traditional file system directory and investigate those that are not recorded in the RMAN repository. If the command can catalog a file, then it will. If it cannot catalog it, then it makes its best guess about the contents of the skipped file.

The `CATALOG RECOVERY AREA` command enables you to catalog all files in the recovery area. Typically, you would not need to run this command manually because RMAN automatically runs it when it is needed, for example, when you restore or create a control file. You can run this command when files are copied into the flash recovery area by means of operating system utilities.

To catalog all files in a disk location:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Run the `CATALOG` command, specifying the disk location whose files you want to catalog.

   For example, enter the following commands:

   ```
   CATALOG START WITH '+disk'; # catalog all files from an ASM disk group
   CATALOG START WITH '/fs1/datafiles/'; # catalog all files in directory
   ```

   > **Note:** Wildcard characters are not legal in the `START WITH` clause.

   Note that you can use the `CATALOG RECOVERY AREA` command to catalog all files in the recovery area. During this operation, any files in the recovery area not listed in the RMAN repository are added. For example:

   ```
   CATALOG RECOVERY AREA;
   ```

3. Run a `LIST` command to verify that the files were cataloged.

Removing Records from the RMAN Repository

This section explains how to remove records for files from the RMAN repository.

About Uncataloging Operations

Run the `CHANGE . . . UNCATALOG` command to perform the following actions on RMAN repository records:

- Update a backup record in the control file repository to status `DELETED`
- Delete a specific backup record from the recovery catalog (if you use one)

RMAN does not touch the specified physical files: it only alters the repository records for these files.

You can use this command when you have deleted a backup through a means other than RMAN. For example, if you delete archived redo logs with an operating system utility, then remove the record for this log from the repository by issuing CHANGE ARCHIVELOG . . . UNCATALOG.

Removing Records for Files Deleted with Operating System Utilities

In some circumstances, users may have removed backups or archived redo logs with operating system utilities. Unless you run CROSSCHECK, RMAN will not know about the deletion. You can use the CHANGE . . . UNCATALOG command to update the RMAN repository for the absent files.

To remove the catalog record for a backup or archived redo log:

1. Run a CHANGE . . . UNCATALOG command for the backups that you deleted from the operating system with operating system commands. This example deletes repository references to disk copies of the control file and datafile 1:

```
CHANGE CONTROLFILECOPY '/tmp/control01.ctl' UNCATALOG;
CHANGE DATAFILECOPY '/tmp/system01.dbf' UNCATALOG;
```

2. Optionally, view the relevant recovery catalog view, for example, RC_DATAFILE_COPY or RC_CONTROLFILE_COPY, to confirm that a given record was removed. For example, this query confirms that the record of copy 4833 was removed:

```
SELECT CDF_KEY, STATUS
FROM RC_DATAFILE_COPY
WHERE CDF_KEY = 4833;

CDF_KEY    STATUS
---------- ------
0 rows selected.
```

Deleting RMAN Backups and Archived Redo Logs

You can use the RMAN DELETE command to delete archived redo logs and RMAN backups. For backups on disk, deleting backups physically removes the backup file from disk. For backups on **SBT** devices, the RMAN DELETE command instructs the media manager to delete the backup pieces or proxy copies on tape. In either case, RMAN updates the RMAN repository to reflect the deletion.

Overview of RMAN Deletion

Every RMAN backup produces a corresponding record in the **RMAN repository**. This record is stored in the control file. If a recovery catalog is used, then the record can also be found in the recovery catalog after the recovery catalog is resynchronized from the control file. For example, if you generate a full database backup set, then you can view the record for this backup set in V$BACKUP_SET. If you use a recovery catalog, then you can also access the record in the RC_BACKUP_SET catalog view.

The V$ control file views and recovery catalog tables differ in the way that they store information, and this affects how RMAN handles repository records. The recovery catalog RMAN repository is stored in actual database tables, while the control file version of the repository is stored in an internal structure in the control file.

When you use an RMAN command to delete a backup or archived redo log file, RMAN does the following:

- Removes the physical file from the operating system (if the file is still present)
- Updates the file records in the control file to status DELETED
- Removes the file records from the recovery catalog tables (if RMAN is connected to a recovery catalog)

Because of the way that control file data is stored, RMAN cannot remove the record from the control file, only update it to DELETED status. Because the recovery catalog tables are ordinary database tables, however, RMAN deletes rows from them in the same way that rows are deleted from any table.

RMAN Deletion Commands

Table 11–2 describes the functionality of the RMAN commands that can cause backups to be deleted.

Table 11–2 RMAN Deletion Commands

Command	Purpose
DELETE	To delete backups, update the control file records to status DELETED, and remove their records from the recovery catalog (if a recovery catalog is used). You can specify that DELETE should remove backups that are EXPIRED or OBSOLETE. If you run DELETE EXPIRED on a backup that exists, then RMAN issues a warning and does not delete the backup. If you use the DELETE command with the optional FORCE keyword, then RMAN deletes the specified backups, but ignores any I/O errors, including those that occur when a backup is missing from disk or tape. It then updates the RMAN repository to reflect the fact that the backup is deleted, regardless of whether RMAN was able to delete the file or whether the file was already missing. RMAN uses all configured channels to perform the deletion. If you use DELETE for files on devices that are not configured for automatic channels, then you must use ALLOCATE CHANNEL FOR MAINTENANCE. For example, if you made a backup with an SBT channel, but only a disk channel is configured, then you must manually allocate an SBT channel for DELETE. An automatic or manually allocated maintenance channel is required when you use DELETE on a disk-only file.
BACKUP . . . DELETE [ALL] INPUT	To back up archived logs, datafile copies, or backup sets, then delete the input files from the operating system after the successful completion of the backup. RMAN also deletes and updates repository records for the deleted input files. If you specify DELETE INPUT (without ALL), then RMAN deletes only the specific files that it backs up. If you specify ALL INPUT, then RMAN deletes all copies of the files recorded in the RMAN repository.
CHANGE . . . UNCATALOG	To delete recovery catalog records for specified backups and change their control file records to status DELETED. Note that the CHANGE . . . UNCATALOG command only changes the RMAN repository record of backups, and does not actually delete backups.

The RMAN repository record for an object can sometimes fail to reflect the physical status of the object. For example, you back up an archived redo log to disk and then use an operating system utility to delete it. If you run DELETE without first running CROSSCHECK, then the repository erroneously lists the log as AVAILABLE. Refer to *Oracle Database Backup and Recovery Reference* for a description of DELETE behavior when mismatches occur between the RMAN repository and physical media.

If you run RMAN interactively, then RMAN asks for confirmation before deleting any files. You can suppress these confirmations by using the NOPROMPT keyword with any form of the BACKUP command:

```
DELETE NOPROMPT ARCHIVELOG ALL;
```

Deletion of Archived Redo Logs

As explained in "Basic Concepts of Backup and Repository Maintenance" on page 11-2, the recommended maintenance strategy is to configure a flash recovery area, a backup retention policy, and an **archived redo log deletion policy**. By default the deletion policy is configured to `NONE`, which means logs are eligible for deletion if they have been backed up at least once to disk or tape or the logs are obsolete according to the backup retention policy.

Archived redo logs can be deleted automatically by the database or as a result of user-initiated RMAN commands listed in Table 11–2. For logs in the recovery area, the database retains them as long as possible and automatically deletes eligible logs when disk space is required. You can delete eligible logs from any location, inside or outside the recovery area, with `BACKUP ... DELETE INPUT` or `DELETE ARCHIVELOG`. If `FORCE` is not specified, then these commands obey the archived log deletion policy. If `FORCE` is specified, then these commands ignore the archived log deletion policy.

See Also:

- "Configuring an Archived Redo Log Deletion Policy" on page 5-26
- The `CONFIGURE ARCHIVELOG DELETION POLICY` entry in *Oracle Database Backup and Recovery Reference* for detailed information about policy options

Deleting All Backups and Copies

In some circumstances, you may need to delete all backup sets, proxy copies, and image copies associated with a database. For example, you no longer need a database and want to remove all related files from the system. Note that an image copy is a file generated with `BACKUP AS COPY`, a log archived by the database, or a file cataloged with the `CATALOG` command.

To delete all backups and copies:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. If necessary, allocate maintenance channels for the devices containing the backups to be deleted.

   As explained in Table 11–2, RMAN uses all configured channels to perform the deletion. If channels are already configured, then you do not need to manually allocate maintenance channels.
3. Crosscheck the backups and copies to ensure that the logical records are synchronized with the physical media.

   ```
   CROSSCHECK BACKUP;
   CROSSCHECK COPY;
   ```
4. Delete the backups and copies.

   For example, enter the following commands and then enter `YES` when prompted:

   ```
   DELETE BACKUP;
   DELETE COPY;
   ```

   If disk and tape channels are configured, then RMAN uses both the configured SBT channel and the preconfigured disk channel when deleting. RMAN prompts you for confirmation before deleting any files.

Deleting Specified Backups and Copies

You can use both the DELETE and BACKUP ... DELETE commands to delete specific backups and copies. The BACKUP ... DELETE command backs up the files first, typically to tape, and then deletes the input files afterward.

Deleting Specified Backups and Copies

The DELETE command supports a wide range of options to identify objects to delete. For complete information about these options, see *Oracle Database Backup and Recovery Reference*. Note that when deleting archived redo logs, RMAN uses the configured settings to determine whether a log can be deleted (see "Configuring an Archived Redo Log Deletion Policy" on page 5-26).

To delete specified backups and copies:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. If necessary, allocate maintenance channels for the devices containing the backups to be deleted.

   As explained in Table 11–2, RMAN uses all configured channels to perform the deletion. If channels are already configured, then you do not need to manually allocate maintenance channels.
3. Delete the specified backups and copies.

   The following examples show many of the common ways to specify backups and archived logs to delete with the DELETE command:

   - Deleting backups using primary keys from LIST output:

     ```
     DELETE BACKUPPIECE 101;
     ```

   - Deleting backups by filename on disk:

     ```
     DELETE CONTROLFILECOPY '/tmp/control01.ctl';
     ```

   - Deleting archived redo logs:

     ```
     DELETE NOPROMPT ARCHIVELOG UNTIL SEQUENCE 300;
     ```

   - Deleting backups based on tags:

     ```
     DELETE BACKUP TAG 'before_upgrade';
     ```

   - Delete backups based on the objects backed up and the media or disk location where the backup is stored:

     ```
     DELETE BACKUP OF TABLESPACE users DEVICE TYPE sbt; # delete only from tape
     DELETE COPY OF CONTROLFILE LIKE '/tmp/%';
     ```

   - Delete backups and archived redo logs from disk based on whether they are backed up on tape:

     ```
     DELETE ARCHIVELOG ALL
        BACKED UP 3 TIMES TO sbt;
     ```

Deleting Specified Files with BACKUP ... DELETE

You can use BACKUP ... DELETE to back up archived redo logs, datafile copies, or backup sets and then delete the input files after successfully backing them up. Specifying the DELETE INPUT option is equivalent to issuing the DELETE command for the input files. As explained in "Configuring an Archived Redo Log Deletion

Policy" on page 5-26, RMAN uses the configured settings to determine whether an archived redo log can be deleted.

The ALL option in the DELETE ALL INPUT clause applies only to archived redo logs. If you run BACKUP ... DELETE ALL INPUT, then the command deletes all copies of corresponding archived redo logs or datafile copies that match the selection criteria in the BACKUP command.

Deleting Expired RMAN Backups and Copies

If you run CROSSCHECK, and if RMAN cannot locate the files, then it updates their records in the RMAN repository to EXPIRED status. You can then use the DELETE EXPIRED command to remove records of expired backups and copies from the RMAN repository.

The DELETE EXPIRED command issues warnings if any files marked as EXPIRED actually exist. In rare cases, the repository can mark a file as EXPIRED even though it exists. For example, a directory containing a file is corrupted at the time of the crosscheck, but is later repaired, or the media manager was not configured properly and reported some backups as not existing when they really existed.

To delete expired repository records:

1. If you have not performed a crosscheck recently, then issue a CROSSCHECK command. For example, issue:

   ```
   CROSSCHECK BACKUP;
   ```

2. Delete the expired backups. For example, issue:

   ```
   DELETE EXPIRED BACKUP;
   ```

Deleting Obsolete RMAN Backups Based on Retention Policies

The RMAN DELETE command supports an OBSOLETE option, which deletes backups that are no longer needed to satisfy specified recoverability requirements. You can delete files obsolete according to the configured default retention policy, or another retention policy that you specify as an option to the DELETE OBSOLETE command. As with other forms of the DELETE command, the files deleted are removed from backup media, deleted from the recovery catalog, and marked as DELETED in the control file.

If you specify the DELETE OBSOLETE command with no arguments, then RMAN deletes all obsolete backups defined by the configured retention policy. For example:

```
DELETE OBSOLETE;
```

DELETE OBSOLETE Behavior When KEEP UNTIL TIME Expires

As long as the KEEP UNTIL TIME period has not expired for an **archival backup**, RMAN does not consider the backup obsolete. As soon as the KEEP UNTIL period expires, however, the backup is immediately considered OBSOLETE, regardless of any configured **backup retention policy**. Thus, DELETE OBSOLETE will delete any backup created with BACKUP ... KEEP UNTIL TIME if the KEEP time has expired.

> **See Also:** *Oracle Database Backup and Recovery Reference* for *keepOption* syntax

Dropping a Database

You may need to remove a database from the operating system. In such a situation, you can use the DROP DATABASE command in RMAN. RMAN removes all datafiles, online redo logs, and control files belonging to the target database.

DROP DATABASE requires that RMAN be connected to the target database, and that the target database be mounted. The command does not require connection to the recovery catalog. If RMAN is connected to the recovery catalog, and if you specify the option INCLUDE COPIES AND BACKUPS, then RMAN also unregisters the database.

To delete a database:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Catalog all backups that are associated with the database. For example, the following commands catalogs files in the flash recovery area, and then in a secondary archiving destination:

   ```
   CATALOG START WITH '+disk1';    # all files from recovery area on ASM disk
   CATALOG START WITH '/arch_dest2';  # all files from second archiving location
   ```

3. Delete all backups and copies associated with the database. For example:

   ```
   DELETE BACKUPSET; # deletes all backups
   DELETE COPY; # delete all image copies (including archived logs)
   ```

4. Remove the database from the operating system.

   The following command deletes the database and automatically unregisters it from the recovery catalog (if used). RMAN prompts for confirmation.

   ```
   DROP DATABASE;
   ```

See Also:

- "Dropping a Database with SQL*Plus" on page 29-16 to learn how to use the SQL DROP DATABASE command
- *Oracle Database Backup and Recovery Reference* for the RMAN DROP DATABASE syntax

12

Managing a Recovery Catalog

This chapter explains how to manage an RMAN recovery catalog. The catalog is a database schema that contains the RMAN repository data for one or more target databases. This chapter contains the following topics:

- Overview of the Recovery Catalog
- Creating a Recovery Catalog
- Registering a Database in the Recovery Catalog
- Cataloging Backups in the Recovery Catalog
- Creating and Managing Virtual Private Catalogs
- Protecting the Recovery Catalog
- Managing Stored Scripts
- Maintaining a Recovery Catalog
- Dropping a Recovery Catalog

See Also:

- Chapter 11, "Maintaining RMAN Backups and Repository Records"to learn how to manage the RMAN repository as stored in the control file, without a recovery catalog
- The compatibility matrix in *Oracle Database Backup and Recovery Reference* describes supported interoperability scenarios

Overview of the Recovery Catalog

This section explains the basic concepts related to managing a recovery catalog.

Purpose of the Recovery Catalog

A **recovery catalog** is a database schema used by RMAN to store metadata about one or more Oracle databases. Typically, you store the catalog in a dedicated database. A recovery catalog provides the following benefits:

- A recovery catalog creates redundancy for the RMAN repository stored in the control file of each **target database**. The recovery catalog serves as a secondary metadata repository. If the target control file and all backups are lost, then the RMAN metadata still exists in the recovery catalog.

- A recovery catalog centralizes metadata for all your target databases. Storing the metadata in a single place makes reporting and administration tasks easier to perform.
- A recovery catalog can store metadata history much longer than the control file. This capability is useful if you have to do a recovery that goes further back in time than the history in the control file. The added complexity of managing a recovery catalog database can be offset by the convenience of having the extended backup history available.

Some RMAN features function only when you use a recovery catalog. For example, you can store RMAN scripts in a recovery catalog. The chief advantage of a **stored script** is that it is available to any RMAN client that can connect to the target database and recovery catalog. Command files are only available if the RMAN client has access to the file system on which they are stored.

A recovery catalog is required when you use RMAN in a Data Guard environment. By storing backup metadata for all primary and standby databases, the catalog enables you to offload backup tasks to one standby database while enabling you to restore backups on other databases in the environment.

Basic Concepts for the Recovery Catalog

The recovery catalog contains metadata about RMAN operations for each registered target database. When RMAN is connected to a recovery catalog, RMAN obtains its metadata exclusively from the catalog. The catalog includes the following types of metadata:

- Datafile and archived redo log backup sets and backup pieces
- Datafile copies
- Archived redo logs and their copies
- Database structure (tablespaces and datafiles)
- Stored scripts, which are named user-created sequences of RMAN commands
- Persistent RMAN configuration settings

Database Registration

The enrolling of a database in a recovery catalog for RMAN use is called **registration**. The recommended practice is to register every **target database** in your environment in a single recovery catalog. For example, you can register databases `prod1`, `prod2`, and `prod3` in a single catalog owned by `catowner` in the database `catdb`.

> **See Also:** "Registering a Database in the Recovery Catalog" on page 12-7

Centralization of Metadata in a Base Recovery Catalog

The owner of a centralized recovery catalog, which is also called the **base recovery catalog**, can grant or revoke restricted access to the catalog to other database users. Each restricted user has full read/write access to his own metadata, which is called a **virtual private catalog**. The RMAN metadata is stored in the schema of the virtual private catalog owner. The owner of the base recovery catalog determines which objects each virtual private catalog user can access.

You can use a recovery catalog in an environment in which you use or have used different versions of Oracle Database. As a result, your environment can have different

versions of the RMAN client, recovery catalog database, recovery catalog schema, and target database. "Importing and Moving a Recovery Catalog" on page 12-31 explains how to merge multiple recovery catalog schemas into one.

> **See Also:** "Creating and Managing Virtual Private Catalogs" on page 12-9

Recovery Catalog Resynchronization

For RMAN operations such as backup, restore, and crosscheck, RMAN always first updates the control file and then propagates the metadata to the recovery catalog. This flow of metadata from the mounted control file to the recovery catalog, which is known as recovery catalog **resynchronization**, ensures that the metadata that RMAN obtains from the control file is current.

> **See Also:** "Resynchronizing the Recovery Catalog" on page 12-22

Stored Scripts

You can use a **stored script** as an alternative to a command file for managing frequently used sequences of RMAN commands. The script is stored in the recovery catalog rather than on the file system.

A local stored script is associated with the target database to which RMAN is connected when the script is created, and can only be executed when you are connected to this target database. A global stored script can be run against any database registered in the recovery catalog. A virtual private catalog user has read-only access to global scripts. Creating or updating global scripts must be done while connected to the base recovery catalog.

> **Tip:** "Managing Stored Scripts" on page 12-15

Recovery Catalog in a Data Guard Environment

As explained in "RMAN in a Data Guard Environment" on page 3-7, you must use a recovery catalog to manage RMAN metadata for all physical databases, both primary and standby databases, in the Data Guard environment. RMAN uses the recovery catalog as the single source of truth for the Data Guard environment.

RMAN can use the recovery catalog to update a primary or standby control file in a **reverse resynchronization**. In this case, the metadata flows from the catalog to the control file rather than the other way around. RMAN automatically performs resynchronizations in most situations in which they are needed. Thus, you should not need to use the `RESYNC` command to manually resynchronize very often.

> **See Also:** *Oracle Data Guard Concepts and Administration* to learn how to configure the RMAN environment for use with a standby database

Basic Steps of Managing a Recovery Catalog

The basic steps for setting up a recovery catalog for use by RMAN are as follows:

1. Create the recovery catalog.

   "Creating a Recovery Catalog" on page 12-4 explains how to perform this task.

2. Register your target databases in the recovery catalog.

   This step enables RMAN to store metadata for the target databases in the recovery catalog. "Registering a Database in the Recovery Catalog" on page 12-7 explains this task.

3. If needed, catalog any older backups whose records are no longer stored in the target control file.

   "Cataloging Backups in the Recovery Catalog" on page 12-9 explains how to perform this task.

4. If needed, create virtual private catalogs for specific users and determine the metadata to which they are permitted access.

   "Creating and Managing Virtual Private Catalogs" on page 12-9 explains how to perform this task.

5. Protect the recovery catalog by including it in your backup and recovery strategy.

   "Protecting the Recovery Catalog" on page 12-21 explains how to back up and recover the catalog as well as increase its availability.

The remainder of the chapter explains how to manage the recovery catalog after it is operational. You can perform the following tasks:

- "Managing Stored Scripts" on page 12-15 explains how to store RMAN scripts in the recovery catalog and manage them.
- Chapter 10, "Reporting on RMAN Operations" explains how to report on RMAN operations. You can use the `LIST` and `REPORT` commands with or without a recovery catalog. "Querying Recovery Catalog Views" on page 10-17 explains how to report on RMAN operations by means of fixed views in the recovery catalog.
- "Maintaining a Recovery Catalog" on page 12-21 explains a variety of tasks for ongoing recovery catalog maintenance, including how to import one recovery catalog into another recovery catalog.

If you no longer want to maintain a recovery catalog, then see "Dropping a Recovery Catalog" on page 12-33.

Creating a Recovery Catalog

This section explains the phases of recovery catalog creation. This section contains the following topics:

- Configuring the Recovery Catalog Database
- Creating the Recovery Catalog Schema Owner
- Executing the CREATE CATALOG Command

Configuring the Recovery Catalog Database

When you use a recovery catalog, RMAN requires that you maintain a recovery catalog schema. The recovery catalog is stored in the default tablespace of the schema. Note that `SYS` cannot be the owner of the recovery catalog.

Decide which database you will use to install the recovery catalog schema, and also how you will back up this database. Also, decide whether to operate the catalog database in `ARCHIVELOG` mode, which is recommended.

> **Note:** Do not use the target database to be backed up as the database for the recovery catalog. The recovery catalog must be protected in the event of the loss of the target database.

Planning the Size of the Recovery Catalog Schema

You must allocate space to be used by the catalog schema. The size of the recovery catalog schema depends upon the number of databases monitored by the catalog. The schema also grows as the number of archived redo log files and backups for each database increases. Finally, if you use RMAN stored scripts stored in the catalog, some space must be allocated for those scripts.

For an example, assume that the `trgt` database has 100 files, and you back up the database once a day, producing 50 backup sets containing 1 backup piece each. If you assume that each row in the backup piece table uses the maximum amount of space, then one daily backup will consume less than 170 KB in the recovery catalog. So, if you back up once a day for a year, then the total storage in this period is about 62 MB. Assume approximately the same amount for archived logs. Thus, the worst case is about 120 MB for a year for metadata storage. For a more typical case in which only a portion of the backup piece row space is used, 15 MB for each year is realistic.

If you plan to register multiple databases in your recovery catalog, then remember to add up the space required for each one based on the previous calculation to arrive at a total size for the default tablespace of the recovery catalog schema.

Allocating Disk Space for the Recovery Catalog Database

If you are creating your recovery catalog in an existing database, then add enough room to hold the default tablespace to the recovery catalog schema. If you are creating a new database to hold your recovery catalog, then in addition to the space for the recovery catalog schema itself, allow space for other files in the recovery catalog database:

- `SYSTEM` and `SYSAUX` tablespaces
- Temporary tablespaces
- Undo tablespaces
- Online redo log files

Most of the space used in the recovery catalog database is devoted to supporting tablespaces, for example, the `SYSTEM`, temporary, and undo tablespaces. Table 12–1 describes typical space requirements.

Table 12–1 Typical Recovery Catalog Space Requirements for 1 Year

Type of Space	Space Requirement
`SYSTEM` tablespace	90 MB
Temp tablespace	5 MB
Rollback or undo tablespace	5 MB
Recovery catalog tablespace	15 MB for each database registered in the recovery catalog
Online redo logs	1 MB each (3 groups, each with 2 members)

Caution: Ensure that the recovery catalog and target databases do *not* reside on the same disk. If both your recovery catalog and your target database suffer hard disk failure, then your recovery process is much more difficult. If possible, take other measures as well to eliminate common points of failure between your recovery catalog database and the databases you are backing up.

Creating the Recovery Catalog Schema Owner

After choosing the recovery catalog database and creating necessary space, you are ready to create the owner of the recovery catalog and grant this user necessary privileges. Assume the following background information for the instructions in the following sections:

- User `SYS` has `SYSDBA` privileges on the recovery catalog database `catdb`.
- A tablespace called `tools` in the recovery catalog database `catdb` stores the recovery catalog. Note that to use an RMAN reserved word as a tablespace name, you must enclose it in quotes and put it in uppercase. (Refer to *Oracle Database Backup and Recovery Reference* for a list of RMAN reserved words.)
- A tablespace called `temp` exists in the recovery catalog database.

To create the recovery catalog schema in the recovery catalog database:

1. Start SQL*Plus and connect with administrator privileges to the database containing the recovery catalog. In this example, the database is `catdb`.
2. Create a user and schema for the recovery catalog. For example, you could enter the following SQL statement (replacing *password* with a user-defined password):

```
CREATE USER rman IDENTIFIED BY password
  TEMPORARY TABLESPACE temp
  DEFAULT TABLESPACE tools
  QUOTA UNLIMITED ON tools;
```

> **Note:** Create a password that is secure. See *Oracle Database Security Guide* for more information.

3. Grant the `RECOVERY_CATALOG_OWNER` role to the schema owner. This role provides the user with all privileges required to maintain and query the recovery catalog.

```
GRANT RECOVERY_CATALOG_OWNER TO rman;
```

Executing the CREATE CATALOG Command

After creating the catalog owner, create the catalog tables with the RMAN `CREATE CATALOG` command. The command creates the catalog in the default tablespace of the catalog owner.

To create the recovery catalog:

1. Start RMAN and connect to the database that will contain the catalog. Connect to the database as the recovery catalog owner.
2. Run the `CREATE CATALOG` command to create the catalog. The creation of the catalog can take several minutes. If the catalog tablespace is this user's default tablespace, then you can run the following command:

```
RMAN> CREATE CATALOG;
```

   You can specify the tablespace name for the catalog in the `CREATE CATALOG` command. For example:

```
RMAN> CREATE CATALOG TABLESPACE cat_tbs;
```

Note: If the tablespace name you wish to use for the recovery catalog is an RMAN reserved word, then it *must* be uppercase and enclosed in quotes. For example:

```
CREATE CATALOG TABLESPACE 'CATALOG';
```

3. You can check the results by using SQL*Plus to query the recovery catalog to see which tables were created:

```
SQL> SELECT TABLE_NAME FROM USER_TABLES;
```

See Also: *Oracle Database SQL Language Reference* for the SQL syntax for the `GRANT` and `CREATE USER` statements, and *Oracle Database Backup and Recovery Reference* for `CREATE CATALOG` command syntax

Registering a Database in the Recovery Catalog

This section describes how to maintain target database records in the recovery catalog. It contains the following sections:

- About Registration of a Database in the Recovery Catalog
- Registering a Database with the REGISTER DATABASE Command

About Registration of a Database in the Recovery Catalog

The enrolling of a **target database** in a recovery catalog is called **registration**. If a target database is not registered in the recovery catalog, then RMAN cannot use the catalog to store metadata for operations on this database. Note that you can still perform RMAN operations on an unregistered database: RMAN always stores its metadata in the control file of the target database.

If you are **not** using the recovery catalog in a Data Guard environment, then use the `REGISTER` command to register each database. Each database must have a unique **DBID**. If you use the `DUPLICATE` command or the `CREATE DATABASE` statement in SQL, then the database is assigned a unique DBID automatically. If you create a database by other means, then the copied database may have the same DBID as its source database. You can change the DBID with the `DBNEWID` utility so that you can register the source and copy databases in the same catalog.

You can use the `UNREGISTER` command to unregister a database from the recovery catalog.

About Standby Database Registration

In a Data Guard environment, the primary and standby databases share the same DBID and database name. To be eligible for registration in the recovery catalog, each database in the Data Guard environment must have different `DB_UNIQUE_NAME` values. The `DB_UNIQUE_NAME` parameter for a database is set in its initialization parameter file.

If you use RMAN in a Data Guard environment, then you can use the `REGISTER DATABASE` command only for the primary database. You can use the following techniques to register a standby database in the recovery catalog:

- When you connect to a standby database as `TARGET`, RMAN automatically registers the database in the recovery catalog.

- When you run the `CONFIGURE DB_UNIQUE_NAME` command for a standby database that is not known to the recovery catalog, RMAN automatically registers this standby database as long as its primary database is registered.

See Also:

- "Unregistering a Target Database from the Recovery Catalog" on page 12-26
- *Oracle Database Backup and Recovery Reference* for `DUPLICATE` syntax
- *Oracle Database Utilities* to learn how to use the `DBNEWID` utility to change the DBID
- *Oracle Data Guard Concepts and Administration* to learn about using RMAN in a Data Guard environment

Registering a Database with the REGISTER DATABASE Command

The first step in using a recovery catalog with a target database is registering the target database in the recovery catalog. If you use the catalog in a Data Guard environment, then you can only register the primary database in this way.

Use the following procedure:

1. Start RMAN and connect to a target database and recovery catalog. The recovery catalog database must be open.

   For example, issue the following to connect to the catalog database with the net service name `catdb` as user `rman` (who owns the catalog schema):

   ```
   % rman TARGET / CATALOG rman@catdb
   ```

2. If the target database is not mounted, then mount or open it:

   ```
   STARTUP MOUNT;
   ```

3. Register the target database in the connected recovery catalog:

   ```
   REGISTER DATABASE;
   ```

   RMAN creates rows in the catalog tables to contain information about the target database, then copies all pertinent data about the target database from the control file into the catalog, synchronizing the catalog with the control file.

4. Verify that the registration was successful by running `REPORT SCHEMA`:

   ```
   REPORT SCHEMA;

   Report of database schema
   File Size(MB)   Tablespace       RB segs Datafile Name
   ---- ---------- ---------------- ------- ------------------
   1        307200 SYSTEM             NO    /oracle/oradata/trgt/system01.dbf
   2         20480 UNDOTBS            YES   /oracle/oradata/trgt/undotbs01.dbf
   3         10240 CWMLITE            NO    /oracle/oradata/trgt/cwmlite01.dbf
   4         10240 DRSYS              NO    /oracle/oradata/trgt/drsys01.dbf
   5         10240 EXAMPLE            NO    /oracle/oradata/trgt/example01.dbf
   6         10240 INDX               NO    /oracle/oradata/trgt/indx01.dbf
   7         10240 TOOLS              NO    /oracle/oradata/trgt/tools01.dbf
   8         10240 USERS              NO    /oracle/oradata/trgt/users01.dbf
   ```

Cataloging Backups in the Recovery Catalog

If you have datafile copies, backup pieces, or archived logs on disk, then you can catalog them in the recovery catalog with the CATALOG command. When using a recovery catalog, cataloging older backups that have aged out of the control file lets RMAN use the older backups during restore operations. The following commands illustrate this technique:

```
CATALOG DATAFILECOPY '/disk1/old_datafiles/01_01_2003/users01.dbf';
CATALOG ARCHIVELOG '/disk1/arch_logs/archive1_731.dbf',
                   '/disk1/arch_logs/archive1_732.dbf';
CATALOG BACKUPPIECE '/disk1/backups/backup_820.bkp';
```

You can also catalog multiple backup files in a directory at once by using the CATALOG START WITH command, as shown in the following example:

```
CATALOG START WITH '/disk1/backups/';
```

RMAN lists the files to be added to the RMAN repository and prompts for confirmation before adding the backups. Be careful when creating your prefix with CATALOG START WITH. RMAN scans all paths for all files on disk which begin with the specified prefix. The prefix is not just a directory name. Using the wrong prefix can cause the cataloging of the wrong set of files.

For example, assume that a group of directories /disk1/backups, /disk1/backups-year2003, /disk1/backupsets, and /disk1/backupsets/test and so on, all contain backup files. The following command catalogs all files in all of these directories, because /disk1/backups is a prefix for the paths for all of these directories:

```
CATALOG START WITH '/disk1/backups';
```

To catalog only backups in the /disk1/backups directory, the correct command would be as follows:

```
CATALOG START WITH '/disk1/backups/';
```

See Also:

- *Oracle Database Backup and Recovery Reference* for REGISTER syntax
- *Oracle Database Upgrade Guide* for issues relating to database migration

Creating and Managing Virtual Private Catalogs

The recommended practice is to create one recovery catalog that serves as the central RMAN repository for all your databases. The recovery catalog as a whole is also termed the **base recovery catalog**. It can contain zero or more virtual private catalogs. A **virtual private catalog** is a set of synonyms and views that refer to a base recovery catalog.

About Virtual Private Catalogs

By default, only the owner of a base recovery catalog has access to its metadata. As the owner of a base recovery catalog, you can use the RMAN GRANT command to grant restricted access to the recovery catalog to other database users. The owner of the base recovery catalog decides which database users can share a recovery catalog and which databases they can access.

When you grant a catalog user restricted access, you give this user full read/write access to his own RMAN metadata, which is the virtual private catalog. Note that a virtual private catalog owner can create a local stored script, but only has read-only access to a global **stored script**. The set of views and synonyms that makes up the virtual private catalog is stored in the schema of the virtual private catalog owner. The mechanisms for virtual private catalogs exist in the recovery catalog schema itself. Security is provided by the recovery catalog database, not by the RMAN executable.

The basic steps for creating a virtual private catalog are as follows:

1. Create the database user who will own the virtual private catalog (if this user does not already exist) and grant this user access privileges.

   This task is described in "Creating and Granting Privileges to a Virtual Private Catalog Owner" on page 12-10.

2. Create the virtual private catalog.

   This task is described in "Creating a Virtual Private Catalog" on page 12-11.

After the virtual private catalog is created, you can revoke catalog access privileges as necessary. This task is described in "Revoking Privileges from a Virtual Private Catalog Owner" on page 12-12. "Dropping a Virtual Private Catalog" on page 12-12 explains how to drop a virtual private catalog.

Note that if the recovery catalog is a virtual private catalog, then the RMAN client connecting to it must be at patch level 10.1.0.6 or 10.2.0.3. Oracle9*i* RMAN clients cannot connect to a virtual private catalog. This version restriction does not affect RMAN client connections to an Oracle Database 11*g* base recovery catalog, even if it has some virtual private catalog users.

> **See Also:** *Oracle Database Backup and Recovery Reference* for details about RMAN version compatibility

Creating and Granting Privileges to a Virtual Private Catalog Owner

This section assumes that you created the base recovery catalog.

Assume that the following databases are registered in the base recovery catalog: `prod1`, `prod2`, and `prod3`. The database user who owns the base recovery catalog is `catowner`. You want to create database user `vpc1` and grant this user access privileges only to `prod1` and `prod2`. By default, a virtual private catalog owner has no access to the base recovery catalog.

To create and grant privileges to a virtual private catalog owner:

1. Start SQL*Plus and connect to the recovery catalog database with administrator privileges.
2. If the user that will own the virtual private catalog is not yet created, then create the user.

   For example, if you want to create database user `vpc1` to own the catalog, then you could execute the following command (replacing *password* with a user-defined password):

```
SQL> CREATE USER vpc1 IDENTIFIED BY password
  2  DEFAULT TABLESPACE vpcusers
  3  QUOTA UNLIMITED ON vpcusers;
```

Note: Create a password that is secure. See *Oracle Database Security Guide* for more information.

3. Grant the RECOVERY_CATALOG_OWNER role to the database user that will own the virtual private catalog, and then exit SQL*Plus.

   The following example grants the role to user vpc1:

   ```
   SQL> GRANT recovery_catalog_owner TO vpc1;
   SQL> EXIT;
   ```

4. Start RMAN and connect to the recovery catalog database as the base recovery catalog owner (*not* the virtual private catalog owner).

   The following example connects to the base recovery catalog as catowner:

   ```
   % rman
   RMAN> CONNECT CATALOG catowner@catdb;

   recovery catalog database Password: password
   connected to recovery catalog database
   ```

5. Grant desired privileges to the virtual private catalog owner.

   The following example gives user vpc1 access to the metadata for prod1 and prod2 (but not prod3):

   ```
   RMAN> GRANT CATALOG FOR DATABASE prod1 TO vpc1;
   RMAN> GRANT CATALOG FOR DATABASE prod2 TO vpc1;
   ```

   You can also use a DBID rather than a database name. Note that the virtual private catalog user does not have access to the metadata for any other databases registered in the recovery catalog.

   You can also grant the user the ability to register new target databases in the recovery catalog. For example:

   ```
   RMAN> GRANT REGISTER DATABASE TO vpc1;
   ```

Creating a Virtual Private Catalog

This section assumes that the virtual private catalog owner has been given the RECOVERY_CATALOG_OWNER database role. Also, the base recovery catalog owner used the GRANT command to give the virtual private catalog owner access to metadata in the base recovery catalog.

To create a virtual private catalog:

1. Start RMAN and connect to the recovery catalog database as the virtual private catalog owner (*not* the base recovery catalog owner).

   The following example connects to the recovery catalog as vpc1:

   ```
   % rman
   RMAN> CONNECT CATALOG vpc1@catdb;
   ```

2. Create the virtual private catalog.

   The following command creates the virtual private catalog:

   ```
   RMAN> CREATE VIRTUAL CATALOG;
   ```

3. If you intend to use a 10.2 or earlier release of RMAN with this virtual private catalog, then execute the following PL/SQL procedure (where *base_catalog_owner* is the database user who owns the base recovery catalog):

```
SQL> EXECUTE base_catalog_owner.DBMS_RCVCAT.CREATE_VIRTUAL_CATALOG;
```

Revoking Privileges from a Virtual Private Catalog Owner

This section assumes that you have already created a virtual private catalog.

Assume that two databases are registered in the base recovery catalog: `prod1` and `prod2`. As owner of the base recovery catalog, you have granted the `vpc1` user access privileges to `prod1`. You have also granted this user the right to register databases in his virtual private catalog. Now you want to revoke privileges from `vpc1`.

To revoke privileges from a virtual private catalog owner:

1. Start RMAN and connect to the recovery catalog database as the recovery catalog owner (*not* the virtual private catalog owner).

   The following example connects to the recovery catalog as `catowner`:

   ```
   % rman
   RMAN> CONNECT CATALOG catowner@catdb;
   ```

2. Revoke specified privileges from the virtual private catalog owner.

   The following command revokes access to the metadata for `prod1` from virtual private catalog owner `vpc1`:

   ```
   REVOKE CATALOG FOR DATABASE prod1 FROM vpc1;
   ```

   You can also specify a DBID rather than a database name. Note that `vpc1` retains all other granted catalog privileges.

   You can also revoke the privilege to register new target databases in the recovery catalog. For example:

   ```
   REVOKE REGISTER DATABASE FROM vpc1;
   ```

Dropping a Virtual Private Catalog

This section assumes that you have already created a virtual private catalog and now want to drop it. When you drop a virtual private catalog, you do not remove the base recovery catalog itself, but only drop the synonyms and views that refer to the base recovery catalog.

To drop a virtual private catalog:

1. Start RMAN and connect to the recovery catalog database as the virtual private catalog owner (*not* the base recovery catalog owner).

   The following example connects to the recovery catalog as user `vpc1`:

   ```
   % rman
   RMAN> CONNECT CATALOG vpc1@catdb;
   ```

2. Drop the catalog.

   If you are using an Oracle Database 11*g* or later RMAN executable, then drop the virtual private catalog with the `DROP CATALOG` command:

   ```
   RMAN> DROP CATALOG;
   ```

If you are using an Oracle Database 10*g* or earlier RMAN executable, then you cannot use the `DROP CATALOG` command. Instead, connect SQL*Plus to the catalog database as the virtual private catalog user, then execute the following PL/SQL procedure (where *base_catalog_owner* is the database user who owns the base recovery catalog):

```
SQL> EXECUTE base_catalog_owner.DBMS_RCVCAT.DELETE_VIRTUAL_CATALOG;
```

Protecting the Recovery Catalog

Include the recovery catalog database in your backup and recovery strategy. If you do not back up the recovery catalog and a disk failure occurs that destroys the recovery catalog database, then you may lose the metadata in the catalog. Without the recovery catalog contents, recovery of your other databases is likely to be more difficult.

Backing Up the Recovery Catalog

A single recovery catalog is able to store metadata for multiple target databases. Consequently, loss of the recovery catalog can be disastrous. You should back up the recovery catalog frequently. This section provides general guidelines for developing a strategy for protecting the recovery catalog.

Backing Up the Recovery Catalog Frequently

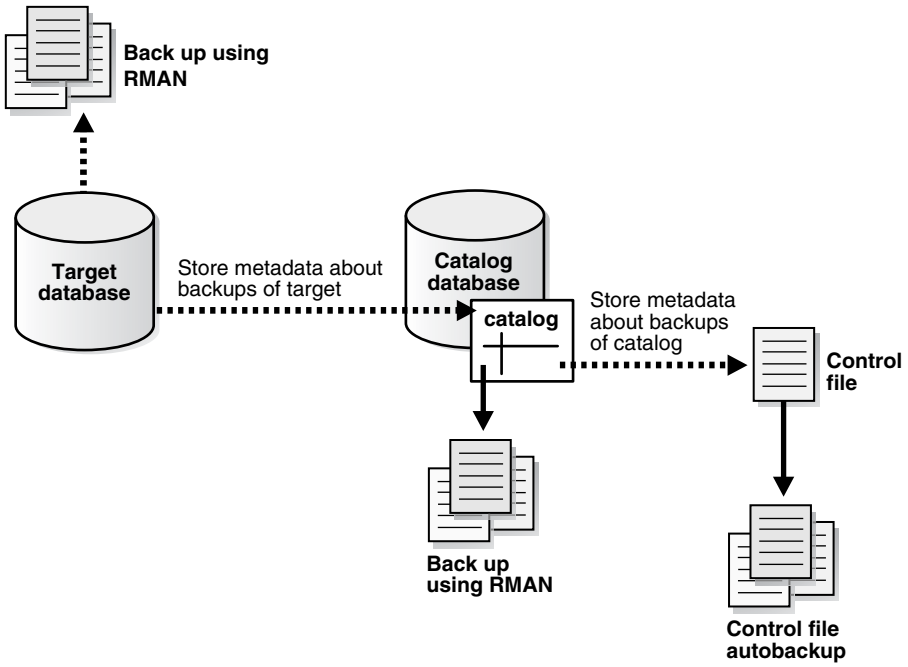
The recovery catalog database is a database like any other, and is also a key part of your backup and recovery strategy. Protect the recovery catalog as you would protect any other part of your database, by backing it up. The backup strategy for your recovery catalog database should be part of your overall backup and recovery strategy.

Back up the recovery catalog with the same frequency that you back up a target database. For example, if you make a weekly whole database backup of a target database, then back up the recovery catalog after the backup of the target database. This backup of the recovery catalog can help you in a disaster recovery scenario. Even if you have to restore the recovery catalog database with a control file autobackup, you can use the full record of backups in your restored recovery catalog database to restore the target database.

Choosing the Appropriate Technique for Physical Backups

When backing up the recovery catalog database, you can use RMAN to make the backups. As illustrated in Figure 12–1, start RMAN with the `NOCATALOG` option so that the repository for RMAN is the control file in the catalog database.

Figure 12–1 Using the Control File as the Repository for Backups of the Recovery Catalog



Follow these guidelines when developing an RMAN backup strategy for the recovery catalog database:

- Run the recovery catalog database in `ARCHIVELOG` mode so that you can do point-in-time recovery if needed.
- Set the retention policy to a `REDUNDANCY` value greater than `1`.
- Back up the database to two separate media (for example, disk and tape).
- Run `BACKUP DATABASE PLUS ARCHIVELOG` at regular intervals, to a media manager if available, or just to disk.
- Do not use another recovery catalog as the repository for the backups.
- Configure the control file autobackup feature to `ON`.

With this strategy, the control file autobackup feature ensures that the recovery catalog database can always be recovered, so long as the control file autobackup is available.

> **See Also:** "Performing Disaster Recovery" on page 19-8 for more information for recovery with a control file autobackup

Separating the Recovery Catalog from the Target Database

A recovery catalog is only effective when separated from the data that it is designed to protect. Thus, you should never store a recovery catalog containing the RMAN repository for a database in the same database as the target database. Also, do not store the catalog database on the same disks as the target database.

To illustrate why data separation is advised, assume that you store the catalog for database `prod1` in `prod1`. If `prod1` suffers a total media failure, and if the recovery catalog for `prod1` is also stored in `prod1`, then if you lose the database you also lose the recovery catalog. At this point the only option is to restore an autobackup of the control file for `prod1` and use it to restore and recover the database without the benefit of any information stored in the recovery catalog.

Exporting the Recovery Catalog Data for Logical Backups

Logical backups of the RMAN recovery catalog created with the Data Pump Export utility can be a useful supplement for physical backups. In the event of damage to a recovery catalog database, you can use Data Pump Import to quickly reimport the exported recovery catalog data into another database and rebuild the catalog.

Recovering the Recovery Catalog

Restoring and recovering the recovery catalog database is much like restoring and recovering any other database with RMAN. You can restore the control file and server parameter file for the recovery catalog database from an autobackup, then restore and perform complete recovery on the rest of the database. If you are in a situation where you are using multiple recovery catalogs, then you can also use another recovery catalog to record metadata about backups of this recovery catalog database.

If recovery of the recovery catalog database through the normal Oracle recovery procedures is not possible, then you must re-create the catalog. Examples of this worst-case scenario include:

- A recovery catalog database that has never been backed up
- A recovery catalog database that has been backed up, but cannot be recovered because the datafile backups or archived logs are not available

You have the following options for partially re-creating the contents of the missing recovery catalog:

- Use the `RESYNC CATALOG` command to update the recovery catalog with any RMAN repository information from the control file of the target database or a control file copy. Note that any metadata from control file records that aged out of the control file is lost.
- Issue `CATALOG START WITH...` commands to recatalog any available backups.

To minimize the likelihood of this worst-case scenario, your backup strategy should at least include backing up the recovery catalog. This technique is described in "Backing Up the Recovery Catalog" on page 12-13.

See Also:

- *Oracle Database Backup and Recovery Reference* for information about the `CATALOG` command
- *Oracle Database Backup and Recovery Reference* for information about the `CROSSCHECK` command

Managing Stored Scripts

As explained in "About Stored Scripts" on page 12-15, you can store scripts in the recovery catalog. This section explains how to create and manage stored scripts.

About Stored Scripts

You can use a **stored script** as an alternative to a command file for managing frequently used sequences of RMAN commands. The script is stored in the recovery catalog rather than on the file system.

Stored scripts can be local or global. A local script is associated with the target database to which RMAN is connected when the script is created, and can only be executed when you are connected to that target database. A global stored script can be

run against any database registered in the recovery catalog, if the RMAN client is connected to the recovery catalog and a target database.

The commands allowable within the brackets of the CREATE SCRIPT command are the same commands supported within a RUN block. Any command that is legal within a RUN command is permitted in the stored script. The following commands are not legal within stored scripts: RUN, @, and @@.

When specifying a script name, RMAN permits but generally does not require that you use quotes around the name of a stored script. If the name begins with a digit or is an RMAN reserved word, however, then you must put quotes around the name to use it as a stored script name. Consider avoiding stored script names that begin with nonalphabetic characters or that are the same as RMAN reserved words.

Consider using a naming convention to avoid confusion between global and local stored scripts. For the EXECUTE SCRIPT, DELETE SCRIPT and PRINT SCRIPT commands, if the script name passed as an argument is not the name of a script defined for the connected target instance, then RMAN looks for a global script by the same name. For example, if the global script global_backup is in the recovery catalog, but no local stored script global_backup is defined for the target database, then the following command deletes the global script:

```
DELETE SCRIPT global_backup;
```

Note that to use commands related to stored scripts, even global scripts, you must be connected to both a recovery catalog and a target database instance.

Creating Stored Scripts

You can use the CREATE SCRIPT command to create a stored script. If GLOBAL is specified, then a global script with this name must not already exist in the recovery catalog. If GLOBAL is not specified, then a local script must not already exist with the same name for the same target database. Note that you can also use the REPLACE SCRIPT to create a new script or update an existing script.

To create a stored script:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Run the CREATE SCRIPT command.

   The following example illustrates creation of a local script:

   ```
   CREATE SCRIPT full_backup
   {
     BACKUP DATABASE PLUS ARCHIVELOG;
     DELETE OBSOLETE;
   }
   ```

   For a global script, the syntax is similar:

   ```
   CREATE GLOBAL SCRIPT global_full_backup
   {
     BACKUP DATABASE PLUS ARCHIVELOG;
     DELETE OBSOLETE;
   }
   ```

   Optionally, you can provide a COMMENT with descriptive information:

   ```
   CREATE GLOBAL SCRIPT global_full_backup
   COMMENT 'use only with ARCHIVELOG mode databases'
   {
   ```

```
  BACKUP DATABASE PLUS ARCHIVELOG;
  DELETE OBSOLETE;
}
```

You can also create a script by reading its contents from a text file. The file must begin with a left brace ({) character, contain a series of commands valid within a RUN block, and end with a right brace (}) character. Otherwise, a syntax error is signalled, just as if the commands were entered at the keyboard.

```
CREATE SCRIPT full_backup
  FROM FILE '/tmp/my_script_file.txt';
```

3. Examine the output.

   If no errors are displayed, then RMAN successfully created the script and stored in the recovery catalog.

   See Also: *Oracle Database Backup and Recovery Reference* for the list of RMAN reserved words

Replacing Stored Scripts

To update stored scripts, use the REPLACE SCRIPT command. If you are replacing a local script, then you must be connected to the target database that you connected to when you created the script. If the script does not already exist, then RMAN creates it.

To replace a stored script:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Execute REPLACE SCRIPT.

   This following example updates the script full_backup with new contents:

```
REPLACE SCRIPT full_backup
{
  BACKUP DATABASE PLUS ARCHIVELOG;
}
```

You can update global scripts by specifying the GLOBAL keyword as follows:

```
REPLACE GLOBAL SCRIPT global_full_backup
COMMENT 'A script for full backup to be used with any database'
{
  BACKUP AS BACKUPSET DATABASE PLUS ARCHIVELOG;
}
```

As with CREATE SCRIPT, you can update a local or global stored script from a text file with the following form of the command:

```
REPLACE GLOBAL SCRIPT global_full_backup
  FROM FILE '/tmp/my_script_file.txt';
```

See Also: *Oracle Database Backup and Recovery Reference* for REPLACE SCRIPT command syntax

Executing Stored Scripts

Use the EXECUTE SCRIPT command to run a stored script. If GLOBAL is specified, then a global script with this name must already exist in the recovery catalog; otherwise, RMAN returns error RMAN-06004. If GLOBAL is not specified, then RMAN searches for a local stored script defined for the current target database. If no local

script with this name is found, then RMAN searches for a global script by the same name and executes it if one is found.

To execute a stored script:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. If needed, use `SHOW` to examine your configured channels.

   Your script will use the automatic channels configured at the time you execute the script. Use `ALLOCATE CHANNEL` commands in the script if you need to override the configured channels. Because of the `RUN` block, if an RMAN command in the script fails, subsequent RMAN commands in the script will not execute.
3. Run `EXECUTE SCRIPT`. This command requires a `RUN` block, as shown in the following example:

```
RUN
{
  EXECUTE SCRIPT full_backup;
}
```

   The preceding command invokes a local script if one is with the name specified. If no local script is found, but there is a global script with the name specified, then RMAN executes the global script.

   You can also use `EXECUTE GLOBAL SCRIPT` to control which script is invoked if a local and a global script have the same name. If there is no local script called `global_full_backup`, the following two commands have the same effect:

```
RUN
{
  EXECUTE GLOBAL SCRIPT global_full_backup;
}

RUN
{
  EXECUTE SCRIPT global_full_backup;
}
```

> **See Also:** *Oracle Database Backup and Recovery Reference* for `EXECUTE SCRIPT` command syntax

Creating and Executing Dynamic Stored Scripts

You can specify substitution variables in the `CREATE SCRIPT` command. When you start RMAN on the command line, the `USING` clause specifies one or more values for use in substitution variables in a command file. As in SQL*Plus, `&1` indicates where to place the first value, `&2` indicates where to place the second value, and so on.

To create and use a dynamic stored script:

1. Create a command file that contains a `CREATE SCRIPT` statement with substitution variables for values that must be dynamically updated.

   The following example uses substitution variables for the name of the tape set, for a string in the `FORMAT` specification, and for the name of the restore point.

```
CREATE SCRIPT quarterly
{
  ALLOCATE CHANNEL c1
    DEVICE TYPE sbt
```

```
    PARMS 'ENV=(OB_MEDIA_FAMILY=&1)';
  BACKUP
    TAG &2
    FORMAT '/disk2/bck/&1%U.bck'
    KEEP FOREVER
    RESTORE POINT &3
    DATABASE;
}
```

2. Connect RMAN to a target database (which must be mounted or open) and recovery catalog, specifying the initial values for the recovery catalog script.

   For example, enter the following command:

   ```
   % rman TARGET / CATALOG rman@catdb USING arc_backup bck0906 FY06Q3
   ```

   A recovery catalog is required for `KEEP FOREVER`, but is not required for any other `KEEP` option.

3. Run the command file created in the first step to create the stored script.

   For example, run the `/tmp/catscript.rman` command file as follows:

   ```
   RMAN> @/tmp/catscript.rman
   ```

   Note that this step creates but does not execute the stored script.

4. Every quarter, execute the stored script, passing values for the substitution variables.

   The following example executes the recovery catalog script named `quarterly`. The example specifies `arc_backup` as the name of the media family (set of tapes), `bck1206` as part of the `FORMAT` string and `FY06Q4` as the name of the restore point.

   ```
   RUN
   {
     EXECUTE SCRIPT quarterly
       USING arc_backup
             bck1206
             FY06Q4;
   }
   ```

   See Also: "Making Database Backups for Long-Term Storage" on page 8-23

Printing Stored Scripts

The `PRINT SCRIPT` command displays a stored script or writes it out to a file.

To print stored scripts:

1. Start RMAN and connect to a target database and recovery catalog.
2. Run the `PRINT SCRIPT` command as follows:

   ```
   PRINT SCRIPT full_backup;
   ```

   To send the contents of a script to a file, use this form of the command:

   ```
   PRINT SCRIPT full_backup
     TO FILE '/tmp/my_script_file.txt';
   ```

   For global scripts, the analogous syntax would be as follows:

```
PRINT GLOBAL SCRIPT global_full_backup;
PRINT GLOBAL SCRIPT global_full_backup
  TO FILE '/tmp/my_script_file.txt';
```

See Also: *Oracle Database Backup and Recovery Reference* for `PRINT SCRIPT` command syntax

Listing Stored Script Names

Use the `LIST ... SCRIPT NAMES` command to display the names of scripts defined in the recovery catalog. `LIST GLOBAL SCRIPT NAMES` and `LIST ALL SCRIPT NAMES` are the only commands that work when RMAN is connected to a recovery catalog without connecting to a target instance; the other forms of the `LIST ... SCRIPT NAMES` command require a recovery catalog connection.

To list stored script names:

1. Start RMAN and connect to a target database and recovery catalog.
2. Run the `LIST ... SCRIPT NAMES` command.

   For example, run the following command to list the names of all global and local scripts that can be executed for the currently connected target database:

   ```
   LIST SCRIPT NAMES;
   ```

   The following example lists *only* global script names:

   ```
   LIST GLOBAL SCRIPT NAMES;
   ```

   To list the names of all scripts stored in the current recovery catalog, including global scripts and local scripts for all target databases registered in the recovery catalog, use the following form of the command:

   ```
   LIST ALL SCRIPT NAMES;
   ```

   The output will indicate for each script listed which target database the script is defined for (or whether a script is global).

   See Also: *Oracle Database Backup and Recovery Reference* for `LIST SCRIPT NAMES` command syntax and output format

Deleting Stored Scripts

Use the `DELETE GLOBAL SCRIPT` command to delete a stored script from the recovery catalog.

To delete a stored script:

1. Start RMAN and connect to a target database and recovery catalog.
2. Enter the `DELETE SCRIPT` command.

   If you use `DELETE SCRIPT` without `GLOBAL`, and there is no stored script for the target database with the specified name, then RMAN looks for a global stored script by the specified name and deletes the global script if it exists. For example, suppose you enter the following command:

   ```
   DELETE SCRIPT 'global_full_backup';
   ```

In this case, RMAN looks for a script `global_full_backup` defined for the connected target database, and if it did not find one, it searches the global scripts for a script called `global_full_backup` and delete that script.

To delete a global stored script, use `DELETE GLOBAL SCRIPT`:

```
DELETE GLOBAL SCRIPT 'global_full_backup';
```

> **See Also:** *Oracle Database Backup and Recovery Reference* for `DELETE SCRIPT` command syntax

Executing a Stored Script at RMAN Startup

To run the RMAN client and start a stored script in the recovery catalog on startup, use the `SCRIPT` argument when starting the RMAN client. For example, you could enter the following command to execute script `/tmp/fbkp.cmd`:

```
% rman TARGET / CATALOG rman@catdb SCRIPT '/tmp/fbkp.cmd';
```

You must connect to a recovery catalog, which contains the stored script, and target database, to which the script will apply, when starting the RMAN client.

If local and global stored scripts are defined with the same name, then RMAN always executes the local script.

> **See Also:** *Oracle Database Backup and Recovery Reference* for full RMAN client command line syntax

Maintaining a Recovery Catalog

This section describes various management and maintenance tasks. This section contains the following topics:

- About Recovery Catalog Maintenance
- Resynchronizing the Recovery Catalog
- Updating the Recovery Catalog After Changing a DB_UNIQUE_NAME
- Unregistering a Target Database from the Recovery Catalog
- Resetting the Database Incarnation in the Recovery Catalog
- Upgrading the Recovery Catalog
- Importing and Moving a Recovery Catalog

About Recovery Catalog Maintenance

After you have created a recovery catalog and registered your target databases, you need to maintain this catalog. For example, you need to run the **RMAN maintenance commands**, which are explained in Chapter 11, "Maintaining RMAN Backups and Repository Records", to update backup records as well as to delete backups that are no longer needed. You must perform this type of maintenance regardless of whether you use RMAN with a recovery catalog. Other types of maintenance, such as upgrading a recovery catalog schema, are specific to use of RMAN with a recovery catalog.

If you use a recovery catalog in a Data Guard environment, then special considerations apply for backups and database files recorded in the catalog. See "RMAN File Management in a Data Guard Environment" on page 3-8 for an explanation of when

backups are accessible to RMAN and how RMAN maintenance commands work with accessible backups.

Resynchronizing the Recovery Catalog

When RMAN performs a **resynchronization**, it compares the recovery catalog to either the current or backup control file of the target database and updates the catalog with metadata that is missing or changed. Most RMAN commands perform a resynchronization automatically when the target control file is mounted and the catalog is available. In a Data Guard environment, RMAN can perform a **reverse resynchronization** to update a database control file with metadata from the catalog.

About Resynchronization of the Recovery Catalog

Resynchronization of the recovery catalog ensures that the metadata that RMAN obtains from the control file stays current. Resynchronizations can be full or partial.

In a **partial resynchronization**, RMAN reads the current control file of the target database to update changed metadata about new backups, new archived redo logs, and so on. RMAN does not resynchronize metadata about the database physical schema.

In a **full resynchronization**, RMAN updates all changed records, including those for the database schema. RMAN performs a full resynchronization after structural changes to database (adding or dropping database files, creating new incarnation, and so on) or after changes to the RMAN persistent configuration.

RMAN creates a **snapshot control file**, which is a temporary backup control file, when it performs a full resynchronization. The database ensures that only one RMAN session accesses a snapshot control file at any point in time. RMAN creates the snapshot control file in an operating system-specific location on the target database host. You can specify the name and location of the snapshot control file, as explained in "Configuring the Snapshot Control File Location" on page 6-11.

This snapshot control file ensures that RMAN has a consistent view of the control file. Because the control file is intended for short-term use, it is not registered in the catalog. RMAN records the control file checkpoint in the recovery catalog to indicate the currency of the catalog.

> **See Also:** *Oracle Database Backup and Recovery Reference* for more information about the `RESYNC` command

Recovery Catalog Resynchronization in a Data Guard Environment RMAN only automatically resynchronizes the recovery catalog with a database when connected to this database as `TARGET`. Thus, RMAN does not automatically resynchronize every database in a Data Guard environment when connected as `TARGET` to one database in the environment. You can use the `RESYNC CATALOG FROM DB_UNIQUE_NAME` command to manually resynchronize the recovery catalog with a database in the Data Guard environment.

For an example of a manual resynchronization, assume that RMAN is connected as `TARGET` to production database `prod`, and that you have used `CONFIGURE` to create a configuration for `dgprod3`. If you run `RESYNC CATALOG FROM DB_UNIQUE_NAME dgprod3`, then RMAN resynchronizes the recovery catalog with the `dgprod3` control file. In this case RMAN performs both a normal resynchronization, in which metadata flows from the `dgprod3` control file to the catalog, and a **reverse resynchronization**. In a reverse resynchronization, RMAN uses the persistent configurations in the recovery catalog to update the `dgprod3` control file.

See Also: *Oracle Data Guard Concepts and Administration*

Deciding When to Resynchronize the Recovery Catalog

RMAN automatically resynchronizes the recovery catalog when RMAN is connected to a target database and recovery catalog and you have executed RMAN commands. Thus, you should not need to manually run the `RESYNC CATALOG` command very often. The following sections describe situations requiring a manual catalog resynchronization.

Resynchronizing After the Recovery Catalog is Unavailable If the recovery catalog is unavailable when you issue RMAN commands that cause a partial resynchronization, then open the catalog database later and resynchronize it manually with the `RESYNC CATALOG` command.

For example, the target database may be in New York while the recovery catalog database is in Japan. You may not want to make daily backups of the target database in `CATALOG` mode, to avoid depending on the availability of a geographically distant database. In such a case you could connect to the catalog as often as feasible and run the `RESYNC CATALOG` command.

Resynchronizing in ARCHIVELOG Mode When You Back Up Infrequently Assume that a target database runs in `ARCHIVELOG` mode. Also assume that you do the following:

- Back up the database infrequently (for example, hundreds of redo logs are archived between database backups)
- Generate a high number of log switches every day (for example, 1000 switches between catalog resynchronizations)

In this case, you may want to manually resynchronize the recovery catalog regularly because the recovery catalog is *not* updated automatically when a redo log switch occurs or when a redo log is archived. The database stores metadata about redo log switches and archived redo logs only in the control file. You must periodically resynchronize in order to propagate this information into the recovery catalog.

How frequently you need to resynchronize the recovery catalog depends on the rate at which the database archives redo logs. The cost of the operation is proportional to the number of records in the control file that have been inserted or changed since the previous resynchronization. If no records have been inserted or changed, then the cost of resynchronization is very low; if many records have been inserted or changed, then the resynchronization is more time-consuming.

Resynchronizing After Configuring a Standby Database You can create or change an RMAN configuration for a standby database even when not connected to this database as `TARGET`. You perform this task with the `CONFIGURE DB_UNIQUE_NAME` or `CONFIGURE ... FOR DB_UNIQUE_NAME` command. As explained in "Manually Resynchronizing the Recovery Catalog" on page 12-24, you can resynchronize the standby database manually to update the control file of the standby database.

Resynchronizing the Recovery Catalog Before Control File Records Age Out Your goal is to ensure that the metadata in the recovery catalog is current. Because the recovery catalog obtains its metadata from the target control file, the currency of the data in the catalog depends on the currency of the data in the control file. You need to make sure that the backup metadata in the control file is recorded in the catalog before it is overwritten with new records.

The `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter determines the minimum number of days that records are retained in the control file before they are

candidates for being overwritten. Thus, you must ensure that you resynchronize the recovery catalog with the control file records before these records are erased. You should perform either of the following actions at intervals less than the CONTROL_FILE_RECORD_KEEP_TIME setting:

- Make a backup, thereby performing an implicit resynchronization of the recovery catalog
- Manually resynchronize the recovery catalog with the RESYNC CATALOG command

Make sure that CONTROL_FILE_RECORD_KEEP_TIME is longer than the interval between backups or resynchronizations. Otherwise, control file records could be reused before they are propagated to the recovery catalog. An extra week is a safe margin in most circumstances.

> **Caution:** Never set CONTROL_FILE_RECORD_KEEP_TIME to 0. If you do, then backup records may be overwritten in the control file before RMAN is able to add them to the catalog.

One problem can arise if the control file becomes too large. The size of the target database control file grows depending on the number of:

- Backups that you perform
- Archived redo logs that the database generates
- Days that this information is stored in the control file

If the control file grows so large that it can no longer expand because it has reached either the maximum number of blocks or the maximum number of records, then the database may overwrite the oldest records even if their age is less than the CONTROL_FILE_RECORD_KEEP_TIME setting. In this case, the database writes a message to the alert log. If you discover that this situation occurs frequently, then reducing the value of CONTROL_FILE_RECORD_KEEP_TIME and increase the frequency of resynchronizations.

See Also:

- *Oracle Database Reference* for more information about the CONTROL_FILE_RECORD_KEEP_TIME parameter
- *Oracle Database Administrator's Guide* for more detailed information on other aspects of control file management
- "Preventing the Loss of Control File Records" on page 11-5 to learn how to monitor the overwriting of control file records

Manually Resynchronizing the Recovery Catalog

Use RESYNC CATALOG to force a full resynchronization of the recovery catalog with a target database control file. Note that you can specify a database unique name with RESYNC FROM DB_UNIQUE_NAME or ALL, depending on whether you want to resynchronize a specific database or all databases in the Data Guard environment. Typically, you would perform this operation after you have run the CONFIGURE command for a standby database, but have not yet connected to this standby database.

1. Start RMAN and connect to a target database and recovery catalog.
2. Mount or open the target database if it is not already mounted or open:

```
STARTUP MOUNT;
```

3. Resynchronize the recovery catalog.

   Run the RESYNC CATALOG command at the RMAN prompt as follows:

   ```
   RESYNC CATALOG;
   ```

   The following example resynchronizes the control file of standby1:

   ```
   RESYNC CATALOG FROM DB_UNIQUE_NAME standby1;
   ```

   The following variation resynchronizes the control files for all databases in the Data Guard environment:

   ```
   RESYNC CATALOG FROM DB_UNIQUE_NAME ALL;
   ```

   See Also:

   - *Oracle Database Backup and Recovery Reference* for RESYNC CATALOG command syntax
   - *Oracle Data Guard Concepts and Administration* to learn how to configure the RMAN environment for use with a standby database

Updating the Recovery Catalog After Changing a DB_UNIQUE_NAME

You may decide to change the DB_UNIQUE_NAME of a database in a Data Guard environment. In this case, you can run the CHANGE DB_UNIQUE_NAME command to associate the metadata stored in recovery catalog for the old DB_UNIQUE_NAME to the new DB_UNIQUE_NAME. Note that the CHANGE DB_UNIQUE_NAME command does not actually change the DB_UNIQUE_NAME of the database itself. Instead, it updates the catalog metadata for the database whose unique name has been or will be changed.

The following procedure assumes that the DB_UNIQUE_NAME of the primary database is prodny, and that you have changed the DB_UNIQUE_NAME of a standby database from prodsf1 to prodsf2. You can use the same procedure after changing the DB_UNIQUE_NAME of a *primary* database, except in step 1 connect RMAN as TARGET to a standby database instead of a primary database.

To update the recovery catalog after a DB_UNIQUE_NAME is changed:

1. Connect RMAN to the *primary* database as TARGET and also to the recovery catalog. For example, enter the following commands:

   ```
   % rman
   RMAN> CONNECT CATALOG catowner@catdb

   recovery catalog database Password: password
   connected to recovery catalog database

   RMAN> CONNECT TARGET SYS@prodny

   target database Password: password
   connected to target database: PRODNY (DBID=39525561)
   ```

2. List the DB_UNQUE_NAME values known to the recovery catalog.

   Run the following LIST command:

   ```
   RMAN> LIST DB_UNIQUE_NAME OF DATABASE;
   ```

3. Change the DB_UNIQUE_NAME in the RMAN metadata.

The following example changes the database unique name from standby database `prodsf1` to `prodsf2`:

```
RMAN> CHANGE DB_UNIQUE_NAME FROM prodsf1 TO prodsf2;
```

Unregistering a Target Database from the Recovery Catalog

You can use the `UNREGISTER DATABASE` command to unregister a database from the recovery catalog. When a database is unregistered from the recovery catalog, all RMAN repository records in the recovery catalog are lost. The database can be registered again, but the recovery catalog records for that database are then based on the contents of the control file at the time of reregistration. Records older than the `CONTROLFILE_RECORD_KEEP_TIME` setting in the target database control file are lost. Stored scripts, which are not stored in the control file, are also lost.

Unregistering a Target Database When Not in a Data Guard Environment

This scenario assumes that you are not using the recovery catalog to store metadata for primary and standby databases.

To unregister a database:

1. Start RMAN and connect as `TARGET` to the database that you want to unregister. Also connect to the recovery catalog.

   It is not necessary to connect to the target database, but if you do not, then you must specify the name of the target database in the `UNREGISTER` command. If more than one database has the same name in the recovery catalog, then you must create a `RUN` block around the command and use `SET DBID` to set the DBID for the database.

2. Make a note of the DBID as displayed by RMAN at startup.

   For example, RMAN outputs a line of the following form when it connects to a target database that is open:

   ```
   connected to target database: PROD (DBID=39525561)
   ```

3. As a precaution, it may be useful to list all of the backups recorded in the recovery catalog using `LIST BACKUP SUMMARY` and `LIST COPY SUMMARY`. This way, you can recatalog backups not known to the control file if you later decide to reregister the database.

4. If your intention is to actually delete all backups of the database completely, then run `DELETE` statements to delete all existing backups. Do not delete all backups if your intention is only to remove the database from the recovery catalog and rely on the control file to store the RMAN metadata for this database.

   The following commands illustrate how to delete backups:

   ```
   DELETE BACKUP DEVICE TYPE sbt;
   DELETE BACKUP DEVICE TYPE DISK;
   DELETE COPY;
   ```

   RMAN lists the backups that it intends to delete and prompts for confirmation before deleting them.

5. Run the `UNREGISTER DATABASE` command. For example:

   ```
   UNREGISTER DATABASE;
   ```

RMAN displays the database name and DBID, and prompts you for a confirmation:

```
database name is "RDBMS" and DBID is 931696259

Do you really want to unregister the database (enter YES or NO)? yes
```

When the process is complete, RMAN outputs the following message:

```
database unregistered from the recovery catalog
```

Unregistering a Standby Database

The UNREGISTER command supports a DB_UNIQUE_NAME clause for use in a Data Guard environment. You can use this clause to remove metadata for a specific database.

The recovery catalog associates a backup with a particular database. When you unregister a database, RMAN updates the database name for these backup files to null. Thus, the backups are still recorded but have no owner. You can execute the CHANGE ... RESET DB_UNIQUE_NAME command to associate ownership of the currently ownerless backups to a different database. If you specify INCLUDING BACKUPS on the UNREGISTER command, then RMAN removes the backup metadata for the unregistered database as well.

In this scenario, assume that primary database lnx3 has associated standby databases standby1. You want to unregister the standby database.

To unregister a standby database:

1. Start RMAN and connect as TARGET to the *primary* database. Also, connect RMAN to a recovery catalog.

   For example, enter the following commands:

   ```
   % rman
   RMAN> CONNECT TARGET SYS@lnx3

   target database Password: password
   connected to target database: LNX3 (DBID=781317675)

   RMAN> CONNECT CATALOG rman@catdb
   ```

2. List the database unique names.

   For example, execute the LIST DB_UNIQUE_NAME command as follows:

   ```
   RMAN> LIST DB_UNIQUE_NAME OF DATABASE;

   List of Databases
   DB Key  DB Name DB ID            Database Role   Db_unique_name
   ------- ------- ----------------- --------------- ------------------
   1       LNX3    781317675         STANDBY         STANDBY
   1       LNX3    781317675         PRIMARY         LNX3
   ```

3. Run the UNREGISTER DB_UNIQUE_NAME command.

   For example, execute the UNREGISTER command as follows to unregister database standby:

   ```
   RMAN> UNREGISTER DB_UNIQUE_NAME standby;
   ```

RMAN displays the database name and DBID, and prompts you for a confirmation:

```
database db_unique_name is "standby", db_name is "LNX3" and DBID is 781317675

Do you really want to unregister the database (enter YES or NO)? yes
```

When the process is complete, RMAN outputs the following message:

```
database with db_unique_name standby unregistered from the recovery catalog
```

Resetting the Database Incarnation in the Recovery Catalog

As explained in "Database Incarnations" on page 13-5, you create a new **incarnation** of the database when you open the database with the RESETLOGS option. You can access a record of the new incarnation in the V$DATABASE_INCARNATION view.

If you open the database with the RESETLOGS option, then a new database incarnation record is automatically created in the recovery catalog. The database also implicitly and automatically issues a RESET DATABASE command, which specifies that this new incarnation of the database is the current incarnation. All subsequent backups and log archiving done by the target database is associated with the new database incarnation.

Whenever RMAN returns the database to an SCN before the current RESETLOGS SCN, either by means of RESTORE and RECOVER or FLASHBACK DATABASE, the RESET DATABASE TO INCARNATION command is required. However, you do not need to execute RESET DATABASE TO INCARNATION explicitly in the following scenarios because RMAN runs the command implicitly with Flashback.

- You use FLASHBACK DATABASE to rewind the database to an SCN in the **direct ancestral path** (see "Database Incarnations" on page 13-5 for an explanation of the direct ancestral path).
- You use FLASHBACK DATABASE to rewind the database to a **restore point**.

The following procedure explains how to reset the database incarnation when recovering through a RESETLOGS.

To reset the recovery catalog to an older incarnation for media recovery:

1. Determine the incarnation key of the desired database incarnation. Obtain the incarnation key value by issuing a LIST command:

```
LIST INCARNATION OF DATABASE trgt;

List of Database Incarnations
DB Key  Inc Key  DB Name  DB ID       STATUS     Reset SCN    Reset Time
------- -------  -------  ------      -------    ----------   ----------
1       2        TRGT     1224038686  PARENT     1            02-JUL-02
1       582      TRGT     1224038686  CURRENT    59727        10-JUL-02
```

   The incarnation key is listed in the Inc Key column.

2. Reset the database to the old incarnation. For example, enter:

```
RESET DATABASE TO INCARNATION 2;
```

3. If the control file of the previous incarnation is available and mounted, then skip to step 6 of this procedure. Otherwise, shut down the database and start it without mounting. For example:

```
SHUTDOWN IMMEDIATE
```

```
STARTUP NOMOUNT
```

4. Restore a control file from the old incarnation. If you have a control file tagged, then specify the tag. Otherwise, you can run the `SET UNTIL` command, as in this example:

```
RUN
{
  SET UNTIL 'SYSDATE-45';
  RESTORE CONTROLFILE; # only if current control file is not available
}
```

5. Mount the restored control file:

```
ALTER DATABASE MOUNT;
```

6. Run `RESTORE` and `RECOVER` commands to restore and recover the database files from the prior incarnation, then open the database with the `RESETLOGS` option. For example, enter:

```
RESTORE DATABASE;
RECOVER DATABASE;
ALTER DATABASE OPEN RESETLOGS;
```

> **See Also:** *Oracle Database Backup and Recovery Reference* for `RESET DATABASE` syntax, *Oracle Database Backup and Recovery Reference* for `LIST` syntax

Upgrading the Recovery Catalog

This section explains what a recovery catalog upgrade is and when you need to do it.

About Recovery Catalog Upgrades

If you use a version of the recovery catalog schema that is older than that required by the RMAN client, then you must upgrade it. The compatibility matrix in *Oracle Database Backup and Recovery Reference* explains which schema versions are compatible with which versions of RMAN. For example, you must upgrade the catalog if you use an Oracle Database 11*g* RMAN client with a release 10.2 version of the recovery catalog schema.

Note that the Oracle Database 10*g*R1 version of the recovery catalog schema requires the `CREATE TYPE` privilege. If you created the recovery catalog owner in a release before 10*g*R1, and if you granted the `RECOVERY_CATALOG_OWNER` role when it did not include the `CREATE TYPE` privilege, then you must grant `CREATE TYPE` to this user explicitly before upgrading the catalog.

You receive an error when issuing `UPGRADE CATALOG` if the recovery catalog is already at a version greater than that required by the RMAN client. RMAN permits the `UPGRADE CATALOG` command to be run if the recovery catalog is current and does not require upgrading, however, so that you can re-create packages at any time if necessary. Check the message log for error messages generated during the upgrade.

Special Considerations in a Data Guard Environment Assume that you upgrade the recovery catalog schema to Oracle Database 11*g* in a Data Guard environment. When RMAN connects to a standby database, it automatically registers the new database information and resynchronizes to obtain the filenames from the control file.

During the resynchronization, RMAN associates the names with the target database name. Because the recovery catalog contains historical metadata, some records in the

catalog will not be known to the control file. For example, the `standby1` control file will not know about all backups made on `primary1`. The database unique names for these old records will be `null`. As explained in "About Recovery Catalog Maintenance" on page 12-21, you can use `CROSSCHECK` to fix these records.

Determining the Schema Version of the Recovery Catalog

The schema version of the recovery catalog is stored in the recovery catalog itself. The information is important in case you maintain multiple databases of different versions in your production system, and need to determine whether the catalog schema version is usable with a specific target database version.

To determine the schema version of the recovery catalog:

1. Start SQL*Plus and connect to the recovery catalog database as the catalog owner.
2. Query the `RCVER` table to obtain the schema version, as in the following example (sample output included):

```
SELECT *
FROM   rcver;

VERSION
------------
10.02.00
```

If the table displays multiple rows, then the highest version in the `RCVER` table is the current catalog schema version. The table stores only the major version numbers and not the patch numbers. For example, assume that the `rcver` table displays the following rows:

```
VERSION
------------
08.01.07
09.00.01
10.02.00
```

These rows indicate that the catalog was created with a release 8.1.7 executable, then upgraded to release 9.0.1, and finally upgraded to release 10.2.0. The current version of the catalog schema is 10.2.0.

> **See Also:** *Oracle Database Backup and Recovery Reference* for the complete set of compatibility rules governing the RMAN environment

Using the UPGRADE CATALOG Command

This scenario assumes that you are upgrading a recovery catalog schema to the current version.

To upgrade the recovery catalog:

1. If you created the recovery catalog owner in a release before 10*g*R1, and if the `RECOVERY_CATALOG_OWNER` role did not include the `CREATE TYPE` privilege, then grant it.

   For example, start SQL*Plus and connect to the recovery catalog database with administrator privileges. You can then execute the following `GRANT` statement:

```
SQL> GRANT CREATE TYPE TO rman;
SQL> EXIT;
```

2. Start RMAN and connect RMAN to the recovery catalog database.
3. Run the `UPGRADE CATALOG` command:

```
RMAN> UPGRADE CATALOG;

recovery catalog owner is rman
enter UPGRADE CATALOG command again to confirm catalog upgrade
```

4. Run the `UPDATE CATALOG` command again to confirm:

```
RMAN> UPGRADE CATALOG;

recovery catalog upgraded to version 11.01.00
DBMS_RCVMAN package upgraded to version 11.01.00
DBMS_RCVCAT package upgraded to version 11.01.00
```

See Also:

- *Oracle Database Backup and Recovery Reference* for `UPGRADE CATALOG` command syntax
- *Oracle Database Backup and Recovery Reference* for information about recovery catalog compatibility
- *Oracle Database Upgrade Guide* for complete compatibility and migration information

Importing and Moving a Recovery Catalog

You can use the `IMPORT CATALOG` command in RMAN to merge one recovery catalog schema into another. This command is useful in the following situations:

- You have multiple recovery catalog schemas for different versions of the database. You want to merge all existing schemas into one without losing backup metadata.
- You want to move a recovery catalog from one database to another database.

About Recovery Catalog Imports

When using `IMPORT CATALOG`, the **source catalog schema** is the catalog schema that you want to import into a different schema. The **destination catalog schema** is the catalog schema into which you intend to import the source catalog schema.

By default, RMAN imports metadata from all target databases registered in the source recovery catalog. Optionally, you can specify the list of database IDs to be imported from the source catalog schema.

By default, RMAN unregisters the imported databases from the source catalog schema after a successful import. To indicate whether the unregister was successful, RMAN prints messages before and after unregistering the merged databases. You can also specify the `NO UNREGISTER` option to specify that the databases should not be unregistered from the destination catalog.

A **stored script** is either global or local. It is possible for global scripts, but not local scripts, to have name conflicts during import because the destination schema already contains the script name. In this case, RMAN renames the global script name to `COPY OF` *`script_name`*. For example, RMAN renames `bp_cmd` to `COPY OF bp_cmd`.

If the renamed global script is still not unique, then RMAN renames it to `COPY(2) OF` *`script_name`*. If this script name also exists, then RMAN renames the script to `COPY(3) OF` *`script_name`*. RMAN continues the `COPY(`*`n`*`) OF` pattern until the script is uniquely named.

Prerequisites for Importing a Recovery Catalog

As shown in compatibility matrix in *Oracle Database Backup and Recovery Reference*, a target database, recovery catalog database, and recovery catalog schema can be at different database versions. The recommended practice is to import all existing recovery catalogs into a single recovery catalog at the latest version of the recovery catalog schema. "Determining the Schema Version of the Recovery Catalog" on page 12-30 explains how to determine the catalog version. Check the compatibility matrix to determine which schema versions are compatible in your environment.

When using `IMPORT CATALOG`, the version of the source recovery catalog schema must be equal to the current version of the RMAN executable with which you run the command. If the source catalog schema is a *lower* version, then upgrade it to the current version before importing the schema. "Upgrading the Recovery Catalog" on page 12-29 explains how to upgrade. If the source recovery catalog schema is a *higher* version, then retry the import with a higher version RMAN executable.

No database can be registered in both the source and destination catalog schema. If a database is currently registered in both catalog schemas, then unregister the database from source catalog schema before performing the import.

Importing a Recovery Catalog

When importing one recovery catalog into another, no connection to a target database is necessary. RMAN only needs connectivity to the source and destination catalogs.

In this example, database `srcdb` contains a 10.2 recovery catalog schema owned by user `102cat`, while database `destdb` contains an 11.1 recovery catalog schema owned by user `111cat`.

To import a recovery catalog:

1. Start RMAN and connect as `CATALOG` to the destination recovery catalog schema. For example:

   ```
   % rman
   RMAN> CONNECT CATALOG 111cat@destdb;
   ```

2. Import the source recovery catalog schema, specifying the connection string for the source catalog.

   For example, enter the following command to import the catalog owned by `102cat` on database `srcdb`:

   ```
   IMPORT CATALOG 102cat@srcdb;
   ```

   A variation is to import metadata for a subset of the target databases registered in the source catalog. You can specify the databases by DBID or database name, as shown in the following examples:

   ```
   IMPORT CATALOG 102cat@srcdb DBID=1423241, 1423242;
   IMPORT CATALOG 102cat@srcdb DB_NAME=prod3, prod4;
   ```

3. Optionally, connect to a target database to check that the metadata was successfully imported. For example, the following commands connect to database `prod1` as `TARGET` and list all backups for this database:

   ```
   LIST BACKUP;
   ```

Moving a Recovery Catalog

The procedure for moving a recovery catalog from one database to another is a variation of the procedure for importing a catalog. In this scenario, the source database is the database containing the existing recovery catalog, while the destination database will contain the moved recovery catalog.

To move a recovery catalog from the source database to the destination database:

1. Create a recovery catalog on the destination database, but do not register any databases in the new catalog.

   "Creating a Recovery Catalog" on page 12-4 explains how to perform this task.

2. Import the source catalog into the catalog created in the preceding step.

   "Importing a Recovery Catalog" on page 12-32 explains how to perform this task.

Dropping a Recovery Catalog

The `DROP CATALOG` command removes those objects that were created as a result of the `CREATE CATALOG` command. If the user who owns the recovery catalog also owns objects that were *not* created by `CREATE CATALOG`, then the `DROP CATALOG` command does not remove these objects.

If you drop a recovery catalog, and if you have no backups of the recovery catalog schema, then backups of all target databases registered in this catalog may become unusable. However, the control file of every target database will still retain a record of recent backups of this database.

The `DROP CATALOG` command is not appropriate for unregistering a single database from a recovery catalog that has multiple target databases registered. Dropping the recovery catalog deletes the recovery catalog record of backups for all target databases registered in the catalog.

To drop a recovery catalog schema:

1. Start RMAN and connect to a target database and recovery catalog. Connect to the recovery catalog as the owner of the catalog schema to be dropped.

   The following example connects to a recovery catalog as user `catowner`:

   ```
   % rman TARGET / CATALOG catowner@catdb
   ```

2. Run the `DROP CATALOG` command:

   ```
   DROP CATALOG;

   recovery catalog owner is catowner
   enter DROP CATALOG command again to confirm catalog removal
   ```

3. Run the `DROP CATALOG` command again to confirm:

   ```
   DROP CATALOG;
   ```

> **Note:** Even after you drop the recovery catalog, the control file still contains records about the backups. To purge RMAN repository records from the control file, re-create the control file.

See Also: *Oracle Database Backup and Recovery Reference* for `DROP CATALOG` command syntax, and "Unregistering a Target Database from the Recovery Catalog" on page 12-26 to learn how to unregister a database from the catalog

Part V

Diagnosing and Responding to Failures

The following chapters describe how to diagnose and respond to media failures and data corruptions. This part of the book contains the following chapters:

- Chapter 13, "RMAN Data Repair Concepts"
- Chapter 14, "Diagnosing and Repairing Failures with Data Recovery Advisor"
- Chapter 15, "Validating Database Files and Backups"
- Chapter 16, "Performing Flashback and Database Point-in-Time Recovery"
- Chapter 17, "Performing Complete Database Recovery"
- Chapter 18, "Performing Block Media Recovery"
- Chapter 19, "Performing RMAN Recovery: Advanced Scenarios"
- Chapter 20, "Performing RMAN Tablespace Point-in-Time Recovery (TSPITR)"

13

RMAN Data Repair Concepts

This chapter describes the general concepts that you need to understand to perform **data repair**. This chapter contains the following topics:

- Overview of RMAN Data Repair
- RMAN Restore Operations
- RMAN Media Recovery

Overview of RMAN Data Repair

As explained in "Data Protection" on page 1-2, a principal purpose of a backup and recovery strategy is data protection. The key to an effective, efficient strategy is to understand the basic options of data repair.

Problems Requiring Data Repair

While several problems can halt the normal operation of an Oracle database or affect database I/O operations, only the following typically require DBA intervention and data repair: user errors, application errors, and media failures.

User Errors

User errors occur when, either due to an error in application logic or a manual mistake, data in your database is changed or deleted incorrectly. For example, a user logs in to the wrong database and drops a database table. User errors are estimated to be the greatest single cause of database downtime.

Application Errors

Sometimes a software malfunction can corrupt data blocks. In a **physical corruption**, which is also called a media corruption, the database does not recognize the block.

Media Failures

A **media failure** occurs when a problem external to the database prevents it from reading from or writing to a file during normal operations. Typical media failures include disk failures and the deletion of database files. Media failures are less common than user or application errors, but your backup and recovery strategy should prepare for them.

RMAN Data Repair Techniques

Depending on the situations you anticipate, consider incorporating each of the following options into your strategy for responding to data loss, and then set up your database to make these options possible.

- **Data Recovery Advisor**

  This Oracle Database infrastructure can diagnose failures, advise you on how to respond to them, and repair the failures automatically.

  "Overview of Data Recovery Advisor" on page 14-1 explains the basic concepts of Data Recovery Advisor.

- **logical flashback features**

  This subset of **Oracle Flashback Technology** features enables you to view or rewind individual database objects or transactions to a past time. These features do not require use of RMAN.

  "Overview of Flashback Technology and Database Point-in-Time Recovery" on page 16-1 explains the basic concepts of the logical flashback features and provides pointers where appropriate.

- **Oracle Flashback Database**

  Flashback Database is a block-level recovery mechanism that is similar to media recovery, but is generally faster and does not require a backup to be restored. You can return your whole database to a previous state without restoring old copies of your datafiles from backup, as long as you have enabled flashback logging in advance. You must have a flash recovery area configured for logging for flashback database or guaranteed restore points.

  "Basic Concepts of Point-in-Time Recovery and Flashback Features" on page 16-1 explains the basic concepts of Flashback Database.

- **datafile media recovery**

  This form of **media recovery** enables you to restore datafile backups and apply archived redo logs or incremental backups to recover lost changes. You can either recover a whole database or a subset of the database. Datafile media recovery is the most general-purpose form of recovery and can protect against both physical and logical failures.

  The general concepts of datafile media recovery are explained in this chapter. The techniques are described in Chapter 17, "Performing Complete Database Recovery" and "Performing Database Point-in-Time Recovery" on page 16-14.

- **block media recovery**

  This form of **media recovery** enables you to recover individual blocks within a datafile rather than the whole datafile.

  "Overview of Block Media Recovery" on page 18-1 explains the basic concepts of block media recovery.

- **tablespace point-in-time recovery (TSPITR)**

  This is a specialized form of **point-in-time recovery** in which you recover one or more tablespaces to a time earlier than the rest of the database.

  "Overview of RMAN TSPITR" on page 20-1 explains the basic concepts of TSPITR.

In general, the concepts required to use the preceding repair techniques are explained along with the techniques. This chapter explains concepts that are common to several RMAN data repair solutions.

RMAN Restore Operations

In an RMAN **restore** operation, you select files to be restored and then run the `RESTORE` command. Typically, you restore files in preparation for media recovery. You can restore the following types of files:

- Database (all datafiles)
- Tablespaces
- Control files
- Archived redo logs
- Server parameter files

You can specify either the default location or a new location for restored datafiles and control files. If you restore to the default location, then RMAN overwrites any files with the same name that currently exist in this location. Alternatively, you can use the `SET NEWNAME` command to specify new locations for restored datafiles. You can then run a `SWITCH` command to update the control file to indicate that the restored files in their new locations are now the current datafiles.

> **See Also:** *Oracle Database Backup and Recovery Reference* for `RESTORE` syntax and prerequisites, *Oracle Database Backup and Recovery Reference* for `SET NEWNAME` syntax, and *Oracle Database Backup and Recovery Reference* for `SWITCH` syntax

Backup Selection

RMAN uses the records of available backup sets or image copies in the RMAN repository to select the best available backups for use in the restore operation. The most recent backup available, or the most recent backup satisfying any `UNTIL` clause specified in the `RESTORE` command, is the preferred choice. If two backups are from the same point in time, then RMAN prefers image copies over backup sets because RMAN can restore more quickly from image copies than from backup sets (especially those stored on tape).

All specifications of the `RESTORE` command must be satisfied before RMAN restores a backup. Unless limited by the `DEVICE TYPE` clause, the `RESTORE` command searches for backups on all device types of configured channels. If no available backup in the repository satisfies all the specified criteria, then RMAN returns an error indicating that the file cannot be restored.

If you use only manually allocated channels, then a backup job may fail if there is no usable backup on the media for which you allocated channels. Configuring automatic channels makes it more likely that RMAN can find and restore a backup that satisfies the specified criteria.

If backup sets are protected with **backup encryption**, then RMAN automatically decrypts them when their contents are restored. Transparently encrypted backups require no intervention to restore, as long as the Oracle wallet is open and available. Password-encrypted backups require the correct password to be entered before they can be restored.

> **See Also:** "Configuring Advanced Channel Options" on page 6-1

Restore Failover

RMAN automatically uses **restore failover** to skip corrupted or inaccessible backups and look for usable backups. When a backup is not found, or contains corrupt data, RMAN automatically looks for another backup from which to restore the desired files.

RMAN generates messages that indicate the type of failover that it is performing. For example, when RMAN fails over to another backup of the same file, it generates a message similar to the following:

```
failover to piece handle=/u01/backup/db_1 tag=BACKUP_031009
```

If no usable copies are available, then RMAN searches for previous backups. The message generated is similar to the following example:

```
ORA-19624: operation failed, retry possible
ORA-19505: failed to identify file "/u01/backup/db_1"
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
failover to previous backup
```

RMAN will perform restore failover repeatedly until it has exhausted all possible backups. If all of the backups are unusable or no backups exists, then RMAN attempts to re-create the datafile. Restore failover is also used when there are errors restoring archived redo logs during `RECOVER`, `RECOVER ... BLOCK`, and `FLASHBACK DATABASE` commands.

Restore Optimization

RMAN uses **restore optimization** to avoid restoring datafiles from backup when possible. If a datafile is already present in the correct location and its header contains the expected information, then RMAN does not restore the datafile from backup.

> **Note:** Restore optimization only checks the datafile header. It does not the scan the datafile body for corrupted blocks.

You can use the `FORCE` option of the `RESTORE` command to override this behavior and restore the requested files unconditionally.

Restore optimization is particularly useful when an operation that restores several datafiles is interrupted. For example, assume that a full database restore encounters a power failure after all except one of the datafiles has been restored. If you run the same `RESTORE` command again, then RMAN only restores the single datafile that was not restored during the previous attempt.

Restore optimization is also used when duplicating a database. If a datafile at the duplicate is in the correct place with the correct header contents, then the datafile is not duplicated. Unlike `RESTORE`, `DUPLICATE` does not support a `FORCE` option. To force RMAN to duplicate a datafile that is skipped due to restore optimization, delete the datafile from the duplicate before running the `DUPLICATE` command.

> **See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* for description of `RESTORE` behavior in an Oracle RAC configuration

RMAN Media Recovery

In **media recovery**, RMAN applies changes to restored data to roll forward this data in time. RMAN can perform either **datafile media recovery** or **block media recovery**.

Datafile media recovery is the application of redo logs or incremental backups to a restored datafile in order to update it to the current time or some other specified time. As explained in *Oracle Database Concepts*, you can use RMAN to perform **complete recovery**, **database point-in-time recovery (DBPITR)**, or **tablespace point-in-time recovery (TSPITR)**. You can use the `RESTORE` command to restore backups of lost and damaged datafiles or control files and the `RECOVER` command to perform media recovery.

Block media recovery is the recovery of individual data blocks rather than entire datafiles. This section explains datafile media recovery only. Block media recovery, which is a specialized form of media recovery, is explained in "Overview of Block Media Recovery" on page 18-1.

Selection of Incremental Backups and Archived Redo Logs

RMAN automates media recovery. RMAN automatically restores and applies both incremental backups and archived redo logs in whatever combination is most efficient.

If the RMAN repository indicates that no copies of a required log sequence number exist on disk, then will automatically restore the required log from backup. By default, RMAN restores the archived logs to the flash recovery area, if one of the archiving destinations is set to `USE_DB_RECOVERY_FILE_DEST`. Otherwise, RMAN restores the logs to the first local archiving destination specified in the initialization parameter file.

> **See Also:** *Oracle Database Backup and Recovery Reference* for `CROSSCHECK` syntax

Database Incarnations

A database **incarnation** is created whenever you open the database with the `RESETLOGS` option. After complete recovery, you can resume normal operations without an `OPEN RESETLOGS`. After a DBPITR or recovery with a backup control file, however, you must open the database with the `RESETLOGS` option, thereby creating a new incarnation of the database. The database requires a new incarnation to avoid confusion when two different redo streams have the same SCNs, but occurred at different times. If you apply the wrong redo to your database, then you will corrupt it.

The existence of multiple incarnations of a single database determines how RMAN treats backups that are not in the current incarnation path. In almost all cases, the current database incarnation is the correct one to use. Nevertheless, in some cases resetting the database to a previous incarnation is the best approach. For example, you may be dissatisfied with the results of a point-in-time recovery that you have already performed and want to return the database to a time before the `RESETLOGS`. An understanding of database incarnations is helpful to prepare for such situations.

OPEN RESETLOGS Operations

When you open the database with the `RESETLOGS` option, the database performs the following actions:

- Archives the current online redo logs (if they are accessible) and then erases the contents of the online redo logs and resets the log sequence number to 1.

For example, if the current online redo logs are sequence 1000 and 1001 when you open RESETLOGS, then the database archives logs 1000 and 1001 and then resets the online redo logs to sequence 1 and 2.

- Creates the online redo log files if they do not currently exist.
- Initializes redo thread records and online redo log records in the control file to the beginning of the new database incarnation.

  More specifically, the database sets the redo thread status to closed, sets the current thread sequence in the redo thread records to 1, sets the thread checkpoint of each redo thread to the beginning of log sequence 1, chooses one redo log from each thread and initialize its sequence to 1, and so on.
- Updates all current datafiles and online redo logs and all subsequent archived redo logs with a new RESETLOGS SCN and time stamp.

Because the database will not apply an archived redo log to a datafile unless the RESETLOGS SCN and time stamps match, the RESETLOGS requirement prevents you from corrupting datafiles with archived logs that are not from direct parent incarnations of the current incarnation. The relationship among incarnations is explained more fully in the following section.

In previous releases, it was recommended that you back up the database immediately after the OPEN RESETLOGS. Because you can now easily recover a pre-RESETLOGS backup like any other backup, making a new database backup is optional. To perform recovery through RESETLOGS you must have all archived logs generated after the most recent backup and at least one control file (current, backup, or created).

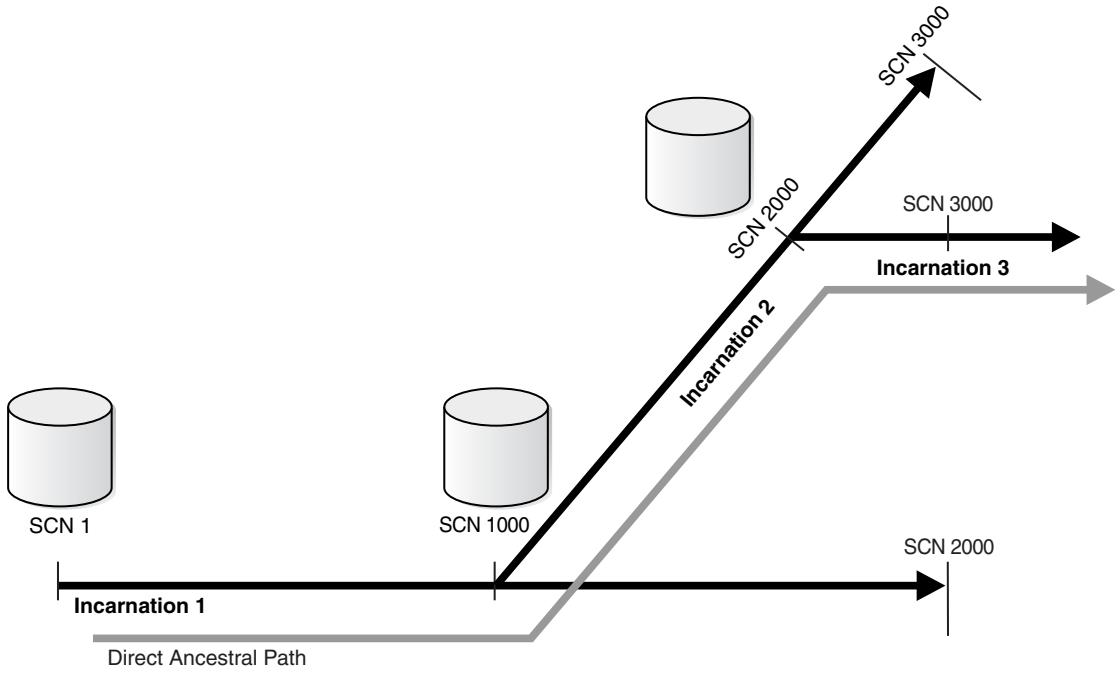
Relationship Among Database Incarnations

Database incarnations can stand in the following relationships to each other:

- The **current incarnation** is the one in which the database is currently operating.
- The incarnation from which the current incarnation branched following an OPEN RESETLOGS operation is the **parent incarnation** of the current incarnation.
- The parent of the parent incarnation is an **ancestor incarnation**. Any parent of an ancestor incarnation is also an ancestor of the current incarnation.
- The **direct ancestral path** of the current incarnation begins with the earliest incarnation and includes only the branches to an ancestor of the current incarnation, the parent incarnation, or the current incarnation.

An incarnation number is used to uniquely tag and identify a stream of redo. Figure 13–1 illustrates a database that goes through several incarnations, each with a different incarnation number.

Figure 13–1 Database Incarnation History



Incarnation 1 of the database starts at SCN 1 and continues through SCN 1000 to SCN 2000. Suppose that at SCN 2000 in incarnation 1, you perform a point-in-time recovery back to SCN 1000, and then open the database with the `RESETLOGS` option. Incarnation 2 now begins at SCN 1000 and continues to SCN 3000. In this example, incarnation 1 is the parent of incarnation 2.

Suppose that at SCN 3000 in incarnation 2, you perform a point-in-time recovery to SCN 2000 and open the database with the `RESETLOGS` option. In this case, incarnation 2 is the parent of incarnation 3. Incarnation 1 is an ancestor of incarnation 3.

When DBPITR or Flashback Database has occurred in database, an SCN can refer to more than one point in time, depending on which incarnation is current. For example, SCN 1500 in Figure 13–1 could refer to an SCN in incarnation 1 or 2.

You can use the `RESET DATABASE TO INCARNATION` command to specify that SCNs are to be interpreted in the frame of reference of a specific database incarnation. The `RESET DATABASE TO INCARNATION` command is required when you use `FLASHBACK`, `RESTORE`, or `RECOVER` to return to an SCN in a noncurrent database incarnation. However, RMAN executes the `RESET DATABASE TO INCARNATION` command implicitly with Flashback, as explained in "Resetting the Database Incarnation in the Recovery Catalog" on page 12-28.

See Also:

- "Recovering the Database to an Ancestor Incarnation" on page 16-20
- *Oracle Database Backup and Recovery Reference* for details about the `RESET DATABASE` command

Orphaned Backups

When a database goes through multiple incarnations, some backups can become **orphaned backups**. Orphaned backups are backups created during incarnations of the database that are not in the direct ancestral path.

Assume the scenario shown in Figure 13–1 on page 13-7. If incarnation 3 is the current incarnation, then the following backups are orphaned:

- All backups from incarnation 1 after SCN 1000
- All backups from incarnation 2 after SCN 2000

In contrast, the following backups are not orphaned because they are in the direct ancestral path:

- All backups from incarnation 1 prior to SCN 1000
- All backups from incarnation 2 prior to SCN 2000
- All backups from incarnation 3

You can use orphaned backups when you intend to restore the database to an SCN not in the direct ancestral path. RMAN can restore backups from parent and ancestor incarnations and recover to the current time, even across `OPEN RESETLOGS` operations, as long as a continuous path of archived logs exists from the earliest backups to the point to which you want to recover. If you restore a control file from an incarnation in which the changes represented in the backups had not been abandoned, then RMAN can also restore and recover orphaned backups.

14

Diagnosing and Repairing Failures with Data Recovery Advisor

This chapter explains how to use the Data Recovery Advisor tool in RMAN to diagnose and repair database failures. This chapter contains the following topics:

- Overview of Data Recovery Advisor
- Listing Failures
- Checking for Block Corruptions by Validating the Database
- Determining Repair Options
- Repairing Failures
- Changing Failure Status and Priority

Overview of Data Recovery Advisor

This section explains the purpose and basic concepts of the **Data Recovery Advisor**.

Purpose of Data Recovery Advisor

Data Recovery Advisor is an Oracle Database tool that automatically diagnoses data failures, determines and presents appropriate repair options, and executes repairs at the user's request. In this context, a data failure is a corruption or loss of persistent data on disk. By providing a centralized tool for automated data repair, Data Recovery Advisor improves the manageability and reliability of an Oracle database and thus helps reduce the **MTTR**.

Diagnosing a data failure and devising an optimal strategy for repair requires a high degree of training and experience. Data Recovery Advisor provides the following advantages over traditional repair techniques:

- Data Recovery Advisor can potentially detect, analyze, and repair data failures *before* a database process discovers the corruption and signals an error. Early warnings help limit damage caused by corruption.
- Manually assessing symptoms of data failures and correlating them into a problem statement can be complex, error-prone, and time-consuming. Data Recovery Advisor automatically diagnoses failures, assesses their impact, and reports these findings to the user.
- Traditionally, users must manually determine repair options along with the repair impact. If multiple failures are present, then users must determine the right sequence of repair execution and try to consolidate repairs. In contrast, Data

Recovery Advisor automatically determines the best repair options and runs checks to make sure that these options are feasible in your environment.

- Execution of a data repair can be complex and error-prone. If you choose an automated repair option, then Data Recovery Advisor executes the repair and verifies its success.

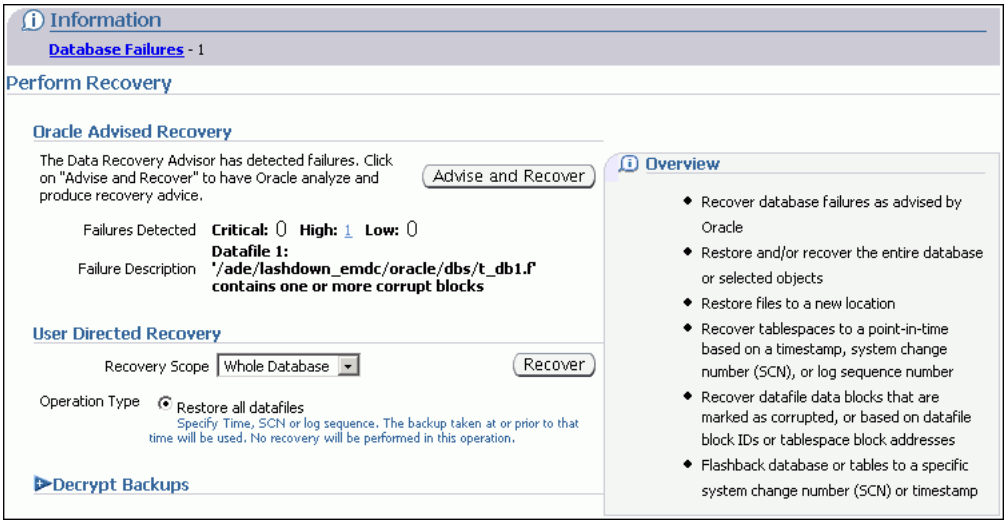
Basic Concepts of Data Recovery Advisor

This section explains the concepts that you need to familiarize yourself with before using Data Recovery Advisor.

User Interfaces to Data Recovery Advisor

Data Recovery Advisor has both a command-line and GUI interface. The GUI interface is available in Oracle Enterprise Manager Database Control and Grid Control. You can click **Perform Recovery** in the Availability tab of the Database Home page to navigate to the page shown in Figure 14–1.

Figure 14–1 Data Recovery Advisor



In the RMAN command-line interface, the Data Recovery Advisor commands are `LIST FAILURE`, `ADVISE FAILURE`, `REPAIR FAILURE`, and `CHANGE FAILURE`.

A **failure** is detected either automatically by the database or through a manual check such as the `VALIDATE` command. You can use the `LIST FAILURE` command to view problem statements for failures and the effect of these failures on database operations. Each failure is uniquely identified by a failure number. In the same RMAN session, you can then use the `ADVISE FAILURE` command to view repair options, which typically include both automated and manual options.

After executing `ADVISE FAILURE`, you can either repair failures manually or run the `REPAIR FAILURE` command to repair the failures automatically. A **repair** is an action that fixes one or more failures. Examples of repairs include **block media recovery**, **datafile media recovery**, and **Oracle Flashback Database**. When you choose an automated **repair option**, Data Recovery Advisor verifies the repair success and closes the relevant repaired failures.

> **See Also:** *Oracle Database 2 Day DBA* to learn how to perform backup and recovery with Enterprise Manager

Data Integrity Checks

A checker is a diagnostic operation or procedure registered with the Health Monitor to assess the health of the database or its components. The health assessment is known as a **data integrity check** and can be invoked reactively or proactively.

Failures are normally detected reactively. A database operation involving corrupted data results in an error, which automatically invokes a data integrity check that searches the database for failures related to the error. If failures are diagnosed, then they are recorded in the **Automatic Diagnostic Repository (ADR)**, which is a directory structure stored outside of the database. You can use Data Recovery Advisor to generate repair advice and repair failures only after failures have been detected by the database and stored in ADR.

You can also invoke a data integrity check proactively. You can execute the check through the Health Monitor, which detects and stores failures in the same way as when the checks are invoked reactively. You can also check for block corruption with the `VALIDATE` and `BACKUP VALIDATE` commands, as explained in "Checking for Block Corruptions by Validating the Database" on page 14-8.

> **See Also:** *Oracle Database Administrator's Guide* to learn how to use the Health Monitor

Failures

A **failure** is a persistent data corruption that is detected by a data integrity check. A failure can manifest itself as observable symptoms such as error messages and alerts, but a failure is different from a symptom because it represents a diagnosed **problem**. After a problem is diagnosed by the database as a failure, you can obtain information about the failure and potentially repair it by means of Data Recovery Advisor.

Because failure information is not stored in the database itself, the database does not need to be open or mounted for you to access it. You can view failures when the database is started in `NOMOUNT` mode. Thus, the availability of the control file and **recovery catalog** does not affect the ability to view detected failures, although it may affect the feasibility of some repairs.

Data Recovery Advisor can diagnose failures such as the following:

- Components such as datafiles and control files that are not accessible because they do not exist, do not have the correct access permissions, have been taken offline, and so on
- Physical corruptions such as block checksum failures and invalid block header field values
- Inconsistencies such as a datafile that is older than other database files
- I/O failures such as hardware errors, operating system driver failures, and exceeding operating system resource limits (for example, the number of open files)

Note that Data Recovery Advisor may detect or handle some logical corruptions. But in general, corruptions of this type require help from Oracle Support Services.

Failure Status Every failure has a **failure status**: `OPEN` or `CLOSED`. The status of a failure is `OPEN` until the appropriate repair action is invoked. The status changes to `CLOSED` after the failure is repaired.

Every time you execute `LIST FAILURE`, Data Recovery Advisor revalidates all open failures and closes failures that no longer exist. Thus, if you fixed some failures as part of a separate procedure, or if the failures were transient problems that disappeared by themselves, running `LIST FAILURE` will automatically close them.

You can use CHANGE FAILURE to change the status of an open failure to CLOSED if you have fixed it manually. However, it makes sense to use CHANGE FAILURE ... CLOSED only if for some reason the failure was not closed automatically. If a failure still exists when you use CHANGE to close it manually, then Data Recover Advisor re-creates it with a different failure ID when the appropriate data integrity check is executed.

Failure Priority Every failure has a **failure priority**: CRITICAL, HIGH, or LOW. Data Recovery Advisor only assigns CRITICAL or HIGH priority to diagnosed failures.

Failures with CRITICAL priority require immediate attention because they make the whole database unavailable. For example, a disk containing a current control file may fail. Failures with HIGH priority make a database partly unavailable or unrecoverable and usually have to be repaired quickly. Examples include block corruptions and missing archived redo logs.

If a failure was assigned a HIGH priority, but the failure has little impact on database availability and recoverability, then you can downgrade the priority to LOW. A LOW priority indicates that a failure can be ignored until more important failures are fixed.

By default LIST FAILURE displays only failures with CRITICAL and HIGH priority. You can use the CHANGE command to change the status for LOW and HIGH failures, but you cannot change the status of CRITICAL failures. The main reason for changing a priority to LOW is to reduce the LIST FAILURE output. If a failure cannot be revalidated at this time (for example, because of another failure), then LIST FAILURE shows the failure as open.

Failure Grouping For clarity, Data Recovery Advisor groups related failures together. For example, if 20 different blocks in a file are corrupted, then these failures will be grouped under a single parent failure. By default, Data Recovery Advisor lists information about the group of failures, although you can specify the DETAIL option to list information about the individual subfailures.

A subfailure has the same format as a failure. You can get advice on a subfailure and repair it separately or in a combination with any other failure.

Manual Actions and Automatic Repair Options

The ADVISE FAILURE command can present both manual and automatic repair options. Data Recovery Advisor categorizes manual actions as either mandatory or optional.

In some cases, the only possible actions are manual. Suppose that no backups exist for a lost control file. In this case, the manual action is to re-create the control file with the CREATE CONTROLFILE statement. Data Recovery Advisor presents this manual action as mandatory because no automatic repair is available. In contrast, suppose that RMAN backups exist for a missing datafile. In this case, the REPAIR FAILURE command can perform the repair automatically by restoring and recovering the datafile. An optional manual action would be to restore the datafile if it was unintentionally renamed or moved. Data Recovery Advisor suggests optional manual actions if they might prevent a more extreme form of repair such as datafile restore and recovery.

In contrast to manual actions, automated repairs can be performed by Data Recovery Advisor. The ADVISE FAILURE command presents an option ID for each automated repair option and summarizes the action.

Data Recovery Advisor performs feasibility checks before recommending an automated repair. For example, Data Recovery Advisor checks that all backups and

archived redo logs needed for media recovery are present and consistent. Data Recovery Advisor may need specific backups and archived redo logs. If the files needed for recovery are not available, then recovery will not possible.

> **Note:** For performance reasons, Data Recovery Advisor does not exhaustively check every byte in every file. Thus, it is possible that a feasible repair may still fail because of a corrupted backup or archived redo log.

Consolidated Repairs When possible, Data Recovery Advisor consolidates repairs to fix multiple failures into a single repair. A consolidated repair may contain multiple steps.

Sometimes a consolidated repair is not possible, as when one failure prevents the creation of repairs for other failures. For example, the feasibility of a datafile repair cannot be determined when the control file is missing. In such cases, Data Recovery Advisor generates a repair option for failures that can be repaired and prints a message stating that some selected failures cannot be repaired at this time. After executing the proposed repair, you can repeat the `LIST`, `ADVISE`, and `REPAIR` sequence to repair remaining failures.

Repair Scripts Whenever Data Recovery Advisor generates an automated repair option, it creates a script that explains which commands RMAN intends to use to repair the failure. Data Recovery Advisor prints the location of this script, which is a text file residing on the operating system. Example 14–1 shows a sample repair script, which shows how Data Recovery Advisor plans to repair the loss of datafile 27.

Example 14–1 Sample Repair Script

```
# restore and recover datafile
sql 'alter database datafile 27 offline';
restore datafile 27;
recover datafile 27;
sql 'alter database datafile 27 online';
```

If you do not want Data Recovery Advisor to automatically repair the failure, then you can copy the script, edit it, and execute it manually.

Supported Database Configurations

In the current release, Data Recovery Advisor only supports single-instance databases. Oracle Real Application Clusters (Oracle RAC) databases are not supported.

If a data failure occurs that brings down all Oracle RAC instances, then you can mount the database in single-instance mode and use Data Recovery Advisor to detect and repair control file, `SYSTEM` datafile, and data dictionary failures. You can also invoke data recovery checks proactively to test other database components for data failures. This approach will not detect data failures that are local to other cluster instances, for example, an inaccessible datafile.

Data Recovery Advisor cannot use blocks or files transferred from a **physical standby database** to repair failures on a primary database. Also, you cannot use Data Recovery Advisor to diagnose and repair failures on a standby database. If you are using Enterprise Manager Grid Control in a Data Guard configuration, however, then you can fail over to a standby database and perform repairs on the old primary database.

> **See Also:** *Oracle Data Guard Concepts and Administration* to learn how to use RMAN in a Data Guard configuration

Basic Steps of Diagnosing and Repairing Failures

The Data Recovery Advisor workflow begins when you either suspect or discover a failure. You can discover failures in many ways, including error messages, alerts, trace files, and failed data integrity checks. As explained in "Data Integrity Checks" on page 14-3, the database can automatically diagnose failures when errors occur.

The basic process for responding to failures is to start an RMAN session and perform all of the following steps in the same session:

1. List failures by running the `LIST FAILURE` command.

   This task is explained in "Listing Failures" on page 14-6.

2. If you suspect that failures exist that have not been automatically diagnosed by the database, then run `VALIDATE DATABASE` to check for corrupt blocks and missing files.

   If `VALIDATE` detects a problem, then RMAN triggers execution of a failure assessment. If a failure is detected, then RMAN logs it into the Automated Diagnostic Repository, where is can be accessed by Data Recovery Advisor.

   This task is explained in "Checking for Block Corruptions by Validating the Database" on page 14-8.

3. Determine repair options by running the `ADVISE FAILURE` command.

   This task is explained in "Determining Repair Options" on page 14-10.

4. Choose a repair option. You can repair the failures manually or run the `REPAIR FAILURE` command to fix them automatically.

   This task is explained in "Repairing Failures" on page 14-12.

5. Return to the first step to confirm that all failures were repaired or determine which failures remain.

If appropriate, you can use `CHANGE FAILURE` command at any time in the Data Recovery Advisor workflow to change the priority of a failure from `LOW` to `HIGH` or `HIGH` to `LOW`, or close a failure that has been fixed manually. This task is explained in "Changing Failure Status and Priority" on page 14-14.

Listing Failures

If you suspect or know that one or more database failures have occurred, then use `LIST FAILURE` to obtain information about them. You can list all or a subset of failures and restrict output in various ways. Failures are uniquely identified by failure numbers. Note that these numbers are not consecutive, so gaps between failure numbers have no significance.

The `LIST FAILURE` command does not execute data integrity checks to diagnose new failures; rather, it lists the results of previously executed assessments. Thus, repeatedly executing `LIST FAILURE` reveals new failures only if the database automatically diagnosed them in response to errors that occurred in between command executions. However, executing `LIST FAILURE` causes Data Recovery Advisor to revalidate all existing failures. If a user fixed failures manually, or if transient failures disappeared, then Data Recovery Advisor removes these failures from the `LIST FAILURE` output. If a failure cannot be revalidated at this moment (for example, because of another failure), `LIST FAILURE` shows the failure as open.

Listing All Failures

The easiest way to determine problems that your database is encountering is to use the `LIST FAILURE` command.

To list all failures:

1. Start RMAN and connect to a target database. The target database instance must be started.
2. Execute the `LIST FAILURE` command.

   The following example reports all failures known to Data Recovery Advisor (the output has been reformatted to fit on the page).

```
RMAN> LIST FAILURE;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
142        HIGH     OPEN      23-APR-07     One or more non-system datafiles
are missing
101        HIGH     OPEN      23-APR-07     Datafile 1:
'/disk1/oradata/prod/system01.dbf' contains one or more corrupt blocks
```

   In this example, RMAN reports two different failures: a group of missing datafiles and a datafile with corrupt blocks. The output indicates the unique identifier for each failure (12 and 5), the priority, status, and detection time.

3. Optionally, execute `LIST FAILURE ... DETAIL` to list failures individually.

   As explained in "Failure Grouping" on page 14-4, Data Recovery Advisor consolidates failures when possible. Specify the `DETAIL` option to list failures individually. For example, if multiple block corruptions exist in a file, then specifying the `DETAIL` option would list each of the block corruptions. The following example lists detailed information about failure 101.

```
RMAN> LIST FAILURE 101 DETAIL;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
101        HIGH     OPEN      23-APR-07     Datafile 1:
'/disk1/oradata/prod/system01.dbf' contains one or more corrupt blocks
  List of child failures for parent failure ID 101
  Failure ID Priority Status    Time Detected Summary
  ---------- -------- --------- ------------- -------
  104        HIGH     OPEN      23-APR-07     Block 56416 in datafile 1:
'/disk1/oradata/prod/system01.dbf' is media corrupt
    Impact: Object BLKTEST owned by SYS might be unavailable
```

4. Proceed to "Determining Repair Options" on page 14-10 to determine how to repair the failures displayed by the `LIST FAILURE` command.

Listing a Subset of Failures

Besides providing more verbose output, `LIST FAILURE` also enables you to restrict output. For example, you can execute `LIST FAILURE` with the `CRITICAL`, `HIGH`,

LOW, or CLOSED options to list only failures with a particular status or priority. You can also exclude specified failures from the output by specifying EXCLUDE FAILURE.

To list a subset of failures:

1. Start RMAN and connect to a target database. The target database instance must be started.
2. Execute LIST FAILURE with the desired options.

   The following examples illustrate some LIST FAILURE commands:

```
LIST FAILURE LOW;
LIST FAILURE CLOSED;
LIST FAILURE EXCLUDE FAILURE 234234;
```

> **See Also:** *Oracle Database Backup and Recovery Reference* to learn about the LIST FAILURE command

Checking for Block Corruptions by Validating the Database

As explained in "Data Integrity Checks" on page 14-3, the database invokes data integrity checks reactively when a user transaction is trying to access corrupted data. In some cases, latent failures can go undetected. For example, when a data block corruption error occurs, the database reactively execute a data integrity check that validates the block on which the error occurred and other blocks in its immediate vicinity. However, blocks outside of the vicinity may be corrupted. Also, corrupted blocks that are never read by the database will never be detected by a reactive data integrity check.

One effective way to execute a data integrity check proactively is to run the VALIDATE or BACKUP VALIDATE commands in RMAN. These commands can check datafiles and control files for physical and logical corruption. If RMAN discovers block corruptions, then it logs them into the Automatic Diagnostic Repository and creates one or more failures. You can then use Data Recovery Advisor to list information about the failures and repair them.

To validate the database:

1. Start RMAN and connect to a target database. The target database must be mounted.
2. Validate the desired database files.

   The following example uses VALIDATE DATABASE to check for physical and logical corruption in the whole database (partial sample output included). Because "Listing Failures" on page 14-6 indicates that some datafiles are missing, the SKIP INACCESSIBLE clause is specified. The output shows that the system01.dbf database file has one newly corrupt block (Blocks Failing) and no blocks previously marked corrupt by the database (Marked Corrupt).

```
RMAN> VALIDATE CHECK LOGICAL SKIP INACCESSIBLE DATABASE;

Starting validate at 23-APR-07
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=103 device type=DISK
could not access datafile 28
skipping inaccessible file 28
RMAN-06060: WARNING: skipping datafile compromises tablespace USERS
recoverability
RMAN-06060: WARNING: skipping datafile compromises tablespace USERS
```

```
recoverability
channel ORA_DISK_1: starting validation of datafile
channel ORA_DISK_1: specifying datafile(s) for validation
input datafile file number=00001 name=/disk1/oradata/prod/system01.dbf
input datafile file number=00002 name=/disk1/oradata/prod/sysaux01.dbf
input datafile file number=00022 name=/disk1/oradata/prod/undotbs01.dbf
input datafile file number=00023 name=/disk1/oradata/prod/cwmlite01.dbf
input datafile file number=00024 name=/disk1/oradata/prod/drsys01.dbf
input datafile file number=00025 name=/disk1/oradata/prod/example01.dbf
input datafile file number=00026 name=/disk1/oradata/prod/indx01.dbf
input datafile file number=00027 name=/disk1/oradata/prod/tools01.dbf
channel ORA_DISK_1: validation complete, elapsed time: 00:00:25
List of Datafiles
=================
File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
1    FAILED 0              3536         57600           637711
  File Name: /disk1/oradata/prod/system01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       1               41876
  Index      0               7721
  Other      0               4467
.
.
.
File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
27   OK     0              1272         1280            400914
  File Name: /disk1/oradata/prod/tools01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0               0
  Index      0               0
  Other      0               8

validate found one or more corrupt blocks
See trace file /disk1/oracle/log/diag/rdbms/prod/prod/trace/prod_ora_2596.trc
 for details
channel ORA_DISK_1: starting validation of datafile
channel ORA_DISK_1: specifying datafile(s) for validation
including current control file for validation
including current SPFILE in backup set
channel ORA_DISK_1: validation complete, elapsed time: 00:00:01
List of Control File and SPFILE
===============================
File Type    Status Blocks Failing Blocks Examined
------------ ------ -------------- ---------------
SPFILE       OK     0              2
Control File OK     0              512
Finished validate at 23-APR-07
```

See Also:

- Chapter 15, "Validating Database Files and Backups"
- *Oracle Database Backup and Recovery Reference* to learn about the `VALIDATE` command
- *Oracle Database Administrator's Guide* to learn about how Oracle Database manages diagnostic data

Determining Repair Options

Use the ADVISE FAILURE command to display repair options after running LIST FAILURE in an RMAN session. This command prints a summary of the failures and implicitly closes all open failures that are already repaired.

Where appropriate, the ADVISE FAILURE command presents a list of manual and automated repair options. Manual options, which are categorized as either mandatory or optional, appear first. In some cases, an optional manual fix can avoid more extreme actions such as restoring and recovering datafiles. As a rule, use the repair technique that has the least effect on the database and the least possibility for error.

Determining Repair Options for All Failures

If one or more failures exist, then you should typically use LIST FAILURE to show information about the failures and then use ADVISE FAILURE in the same RMAN session to obtain a report of your repair options.

To determine repair options for all failures:

1. List failures as described in "Listing All Failures" on page 14-7.
2. In the same RMAN session, execute ADVISE FAILURE.

   The following example requests repair options for all failures known to Data Recovery Advisor and includes sample output (reformatted to fit the page).

```
RMAN> ADVISE FAILURE;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
142        HIGH     OPEN      23-APR-07     One or more non-system datafiles
                                            are missing
101        HIGH     OPEN      23-APR-07     Datafile 1:
'/disk1/oradata/prod/system01.dbf' contains one or more corrupt blocks

analyzing automatic repair options; this may take some time
using channel ORA_DISK_1
analyzing automatic repair options complete

Mandatory Manual Actions
========================
no manual actions available

Optional Manual Actions
=======================
1. If file /disk1/oradata/prod/users01.dbf was unintentionally renamed or
moved, restore it

Automated Repair Options
========================
Option Repair Description
------ ------------------
1      Restore and recover datafile 28; Perform block media recovery of
       block 56416 in file 1
  Strategy: The repair includes complete media recovery with no data loss
  Repair script: /disk1/oracle/log/diag/rdbms/prod/prod/hm/reco_660500184.hm
```

In the preceding example, ADVISE FAILURE reports two failures: a missing datafile and a datafile with corrupt blocks. The command does not list mandatory manual actions, but it suggests making sure that the missing datafile was not accidentally renamed or removed. The automated repair option involves **block media recovery** and restoring and recovering the missing datafile. ADVISE FAILURE lists the location of the repair script.

The following variation of the same example shows the output when the RMAN backups or archived redo logs needed for the automated repair are not available. Note that ADVISE FAILURE now shows mandatory manual actions.

```
RMAN> ADVISE FAILURE;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
142        HIGH     OPEN      23-APR-07     One or more non-system datafiles
                                            are missing
101        HIGH     OPEN      23-APR-07     Datafile 1:
'/disk1/oradata/prod/system01.dbf' contains one or more corrupt blocks

analyzing automatic repair options; this may take some time
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=103 device type=DISK
analyzing automatic repair options complete

Mandatory Manual Actions
========================
1. If file /disk1/oradata/prod/users01.dbf was unintentionally renamed or
moved, restore it
2. Contact Oracle Support Services if the preceding recommendations cannot be
used, or if they do not fix the failures selected for repair

Optional Manual Actions
=======================
no manual actions available

Automated Repair Options
========================
Option Repair Description
------ ------------------
1      Perform block media recovery of block 56416 in file 1
  Strategy: The repair includes complete media recovery with no data loss
  Repair script: /disk1/oracle/log/diag/rdbms/prod/prod/hm/reco_1863891774.hm
```

3. Proceed to "Repairing Failures" on page 14-12 to determine how to repair the failures shown in the LIST FAILURE output.

Determining Repair Options for a Subset of Failures

You can also request repair options for specific failures. You can specify failures by status (CRITICAL, HIGH, or LOW) or by failure number. You can also use EXCLUDE FAILURE to exclude one or more failures from the report.

To determine repair options for a subset of failures:

1. List failures as described in "Listing All Failures" on page 14-7.

2. In the same RMAN session, execute `ADVISE FAILURE` with the desired options.

   The following example requests repair options for failure 101 only.

```
RMAN> ADVISE FAILURE 101;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
101        HIGH     OPEN      23-APR-07     Datafile 1:
'/disk1/oradata/prod/system01.dbf' contains one or more corrupt blocks

analyzing automatic repair options; this may take some time
using channel ORA_DISK_1
analyzing automatic repair options complete

Mandatory Manual Actions
========================
no manual actions available

Optional Manual Actions
=======================
no manual actions available

Automated Repair Options
========================
Option Repair Description
------ ------------------
1      Perform block media recovery of block 56416 in file 1
  Strategy: The repair includes complete media recovery with no data loss
  Repair script: /disk1/oracle/log/diag/rdbms/prod/prod/hm/reco_708819503.hm
```

3. Proceed to "Repairing Failures" on page 14-12 to determine how to repair the failures displayed by the `LIST FAILURE` command.

   See Also: *Oracle Database Backup and Recovery Reference* to learn about the `ADVISE FAILURE` command

Repairing Failures

This section explains how to use Data Recovery Advisor to repair failures automatically.

About Repairing Failures

If `ADVISE FAILURE` suggests manual repairs, then try these first. If manual repairs are not possible, or if they do not repair all failures, then you can use `REPAIR FAILURE` to automatically fix failures suggested in the most recent `ADVISE FAILURE` command in your current RMAN session.

By default, `REPAIR FAILURE` prompts for confirmation before it begins executing. You can suppress the confirmation prompt by specifying the `NOPROMPT` option. After it starts executing, the command indicates the current phase of repair. Depending on the circumstances, RMAN may prompt for a response. After executing a repair, RMAN reevaluates all existing failures on the chance that they may have been fixed during this repair.

Before performing a repair, it is typically advisable to preview it by specifying the PREVIEW option. RMAN does not make any repairs and generates a script with all repair actions and comments. If you do not specify a particular repair option, then RMAN uses the first repair option of the most recent ADVISE FAILURE command in the current session. By default the repair script is displayed to standard output. You can use the SPOOL command to write the script to an editable file.

See Also:

- *Oracle Database Backup and Recovery Reference* to learn about the REPAIR FAILURE command
- *Oracle Database Backup and Recovery Reference* to learn about the SPOOL command

Repairing a Failure

By default the script is displayed to standard output. You can use the SPOOL command to write the script to an editable file.

To repair a failure:

1. List failures as described in "Listing All Failures" on page 14-7.
2. Display repair options as described in "Determining Repair Options" on page 14-10.
3. Optionally, execute REPAIR FAILURE PREVIEW.

   The following example previews the first repair options displayed by the previous ADVISE FAILURE command in the RMAN session.

```
RMAN> REPAIR FAILURE PREVIEW;

Strategy: The repair includes complete media recovery with no data loss
Repair script: /disk1/oracle/log/diag/rdbms/prod/prod/hm/reco_475549922.hm
contents of repair script:
   # restore and recover datafile
   sql 'alter database datafile 28 offline';
   restore datafile 28;
   recover datafile 28;
   sql 'alter database datafile 28 online';
   # block media recovery
   recover datafile 1 block 56416;
```

4. Execute REPAIR FAILURE.

   The following repair restores and recovers one datafile and performs block media recovery on one corrupt block. RMAN prompts for confirmation that it should perform the repair. The user-entered text is in bold.

```
RMAN> REPAIR FAILURE;

Strategy: The repair includes complete media recovery with no data loss
Repair script: /disk1/oracle/log/diag/rdbms/prod/prod/hm/reco_475549922.hm
contents of repair script:
   # restore and recover datafile
   sql 'alter database datafile 28 offline';
   restore datafile 28;
   recover datafile 28;
   sql 'alter database datafile 28 online';
   # block media recovery
```

```
   recover datafile 1 block 56416;

Do you really want to execute the above repair (enter YES or NO)? YES
executing repair script

sql statement: alter database datafile 28 offline

Starting restore at 23-APR-07
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00028 to /disk1/oradata/prod/users01.dbf
channel ORA_DISK_1: reading from backup piece /disk2/PROD/backupset/2007_04_
18/o1_mf_nnndf_TAG20070418T182042_32fjzd3z_.bkp
channel ORA_DISK_1: piece handle=/disk2/PROD/backupset/2007_04_18/o1_mf_nnndf_
TAG20070418T182042_32fjzd3z_.bkp tag=TAG20070418T182042
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
Finished restore at 23-APR-07

Starting recover at 23-APR-07
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:01

Finished recover at 23-APR-07

sql statement: alter database datafile 28 online

Starting recover at 23-APR-07
using channel ORA_DISK_1
searching flashback logs for block images until SCN 429690
finished flashback log search, restored 1 blocks

starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished recover at 23-APR-07
repair failure complete
```

5. Optionally, execute `LIST FAILURE` to confirm

Changing Failure Status and Priority

In some situations, you may want to use the `CHANGE FAILURE` command to alter the status or priority of a failure. For example, if a block corruption has `HIGH` priority, you may want to change it to `LOW` temporarily if the block is in a little-used tablespace.

If you repair a failure by a means other than the `REPAIR FAILURE` command, then Data Recovery Advisor closes it implicitly the next time you execute `LIST FAILURE`. For this reason, you do not normally need to execute the `CHANGE FAILURE ... CLOSED` command. You should need to use this command only if the automatic failure revalidation fails, but you believe the failure no longer exists. Note that if you use `CHANGE FAILURE` to close a failure that still exists, then Data Recovery Advisor re-creates it with a different failure ID when the appropriate data integrity check is executed.

Typically, you specify the failures that you want to change by failure number. You can also change failures in bulk by specifying ALL, CRITICAL, HIGH, or LOW. You can change a failure to CLOSED or to PRIORITY HIGH or PRIORITY LOW.

To change the status or priority of a failure:

1. List failures as described in "Listing All Failures" on page 14-7.

   The following example lists one failure involving corrupt data blocks.

```
RMAN> LIST FAILURE;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
142        HIGH     OPEN      23-APR-07     One or more non-system datafiles
                                            are missing
101        HIGH     OPEN      23-APR-07     Datafile 25:
'/disk1/oradata/prod/example01.dbf' contains one or more corrupt blocks
```

2. Execute CHANGE FAILURE with the desired options.

   The following example changes the priority of a block corruption failure from HIGH to LOW.

```
RMAN> CHANGE FAILURE 101 PRIORITY LOW;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
101        HIGH     OPEN      23-APR-07     Datafile 25:
'/disk1/oradata/prod/example01.dbf' contains one or more corrupt blocks

Do you really want to change the above failures (enter YES or NO)? YES
changed 1 failures to LOW priority
```

3. Optionally, execute LIST FAILURE ALL to view the change.

   Note that if you execute LIST FAILURE without ALL, then the command lists failures with LOW priority only if no CRITICAL or HIGH priority failures exist.

```
RMAN> LIST FAILURE ALL;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
142        HIGH     OPEN      23-APR-07     One or more non-system datafiles
                                            are missing
101        LOW      OPEN      23-APR-07     Datafile 25:
'/disk1/oradata/prod/example01.dbf' contains one or more corrupt blocks
```

See Also: *Oracle Database Backup and Recovery Reference* to learn about the CHANGE command

15

Validating Database Files and Backups

This chapter explains how to check the integrity of database files and backups. This chapter includes the following topics:

- Overview of RMAN Validation
- Checking for Block Corruption with the VALIDATE Command
- Validating Database Files with BACKUP VALIDATE
- Validating Backups Before Restoring Them

Overview of RMAN Validation

This section explains the basic concepts and tasks involved in RMAN validation.

Purpose of RMAN Validation

The main purpose of RMAN **validation** is to check for corrupt blocks and missing files. You can also use RMAN to determine whether backups can be restored. You can use the following RMAN commands to perform validation:

- `VALIDATE`
- `BACKUP ... VALIDATE`
- `RESTORE ... VALIDATE`

See Also:

- *Oracle Database Backup and Recovery Reference* for `VALIDATE` syntax
- *Oracle Database Backup and Recovery Reference* for `RESTORE ... VALIDATE` syntax

Basic Concepts of RMAN Validation

The database prevents operations that result in unusable backup files or corrupted restored datafiles. The database automatically does the following:

- Blocks access to datafiles while they are being restored or recovered
- Permits only one restore operation for each datafile at a time
- Ensures that incremental backups are applied in the correct order

- Stores information in backup files to allow detection of corruption
- Checks a block every time it is read or written in an attempt to report a corruption as soon as it has been detected

Checksums and Corrupt Blocks

A **corrupt block** is a block that has been changed so that it differs from what Oracle Database expects to find. Block corruptions can be caused by a number of different failures including, but not limited to the following:

- Faulty disks and disk controllers
- Faulty memory
- Oracle Database software defects

`DB_BLOCK_CHECKSUM` is a database initialization parameter that controls the writing of checksums for the blocks in datafiles and online redo log files in the database (not backups). If `DB_BLOCK_CHECKSUM` is `typical`, then the database computes a checksum for each block during normal operations and stores it in the header of the block before writing it to disk. When the database reads the block from disk later, it recomputes the checksum and compares it to the stored value. If the values do not match, then the block is corrupt.

By default, the `BACKUP` command computes a checksum for each block and stores it in the backup. The `BACKUP` command ignores the values of `DB_BLOCK_CHECKSUM` because this initialization parameter applies to datafiles in the database, not backups.

Physical and Logical Block Corruption

In a **physical corruption**, which is also called a media corruption, the database does not recognize the block at all: the **checksum** is invalid, the block contains all zeros, or the header and footer of the block do not match.

> **Note:** By default, the `BACKUP` command computes a checksum for each block and stores it in the backup. If you specify the `NOCHECKSUM` option, then RMAN does not perform a checksum of the blocks when creating the backup.

In a **logical corruption**, the contents of the block are logically inconsistent. Examples of logical corruption include corruption of a row piece or index entry. If RMAN detects logical corruption, then it logs the block in the alert log and server session trace file.

By default, RMAN does not check for logical corruption. If you specify `CHECK LOGICAL` on the `BACKUP` command, however, then RMAN tests data and index blocks for logical corruption, such as corruption of a row piece or index entry, and log them in the alert log located in the **Automatic Diagnostic Repository (ADR)**. If you use RMAN with the following configuration when backing up or restoring files, then it detects all types of block corruption that are possible to detect:

- In the initialization parameter file of a database, set `DB_BLOCK_CHECKSUM=typical` so that the database calculates datafile checksums automatically (not for backups, but for datafiles in use by the database)
- Do not precede the `BACKUP` or `RESTORE` command with `SET MAXCORRUPT` so that RMAN does not tolerate any block corruptions
- In a `BACKUP` command, do *not* specify the `NOCHECKSUM` option so that RMAN calculates a checksum when writing backups

- In BACKUP and RESTORE commands, specify the CHECK LOGICAL option so that RMAN checks for logical as well as physical corruption

Limits for Corrupt Blocks in RMAN Backups

You can use the SET MAXCORRUPT command to set the total number of corruptions permitted in a file for RMAN backups. The default is zero, meaning that RMAN tolerates no corrupt blocks of any kind.

If the MAXCORRUPT limit is exceeded when RMAN encounters a corrupt block during a backup, then RMAN terminates the backup. Otherwise, RMAN writes the corrupt block to the backup with a special header indicating that the block is marked corrupt. You can use the VALIDATE command to determine which blocks are marked corrupt.

Because RMAN can permit block corruptions in a backup, it is possible to restore a datafile that RMAN knows to contain block corruptions. If you back up this restored datafile, then RMAN does not consider blocks already marked corrupt when it calculates whether MAXCORRUPT has been exceeded.

> **See Also:** *Oracle Database Backup and Recovery Reference* for SET MAXCORRUPT syntax

Detection of Block Corruption

Oracle Database supports different techniques for detecting, repairing, and monitoring block corruption. The technique depends on whether the corruption is **interblock corruption** or **intrablock corruption**. In intrablock corruption, the corruption occurs within the block itself. This corruption can be either physical or logical. In an interblock corruption, the corruption occurs between blocks and can only be logical.

For example, the V$DATABASE_BLOCK_CORRUPTION view records intrablock corruptions, while the **Automatic Diagnostic Repository (ADR)** tracks all types of corruptions. Table 15–1 summarizes how the database treats different types of block corruption.

Table 15–1 Detection, Repair, and Monitoring of Block Corruption

Response	Intrablock Corruption	Interblock Corruption
Detection	All database utilities detect intrablock corruption, including RMAN (for example, the BACKUP command) and the DBVERIFY utility. If a database process can encounter the ORA-1578 error, then it can detect the corruption and monitor it.	Only DBVERIFY and the ANALYZE statement detect interblock corruption.
Tracking	The V$DATABASE_BLOCK_CORRUPTION view displays blocks marked corrupt by Oracle Database components such as RMAN commands, ANALYZE, dbv, SQL queries, and so on. Any process that encounters an intrablock corruption records the block corruption in this view and in ADR.	The database monitors this type of block corruption in ADR.
Repair	Repair techniques include **block media recovery**, restoring datafiles, recovering by means of incremental backups, and block newing. Block media recovery can repair physical corruptions, but not logical corruptions. Any RMAN command that fixes or detects that a block is repaired updates V$DATABASE_BLOCK_CORRUPTION. For example, RMAN updates the repository at end of successful block media recovery. If a BACKUP, RESTORE, or VALIDATE command detects that a block is no longer corrupted, then it removes the repaired block from the view.	You must fix interblock corruption by means of manual techniques such as dropping an object, rebuilding an index, and so on.

See Also:

- Chapter 17, "Performing Complete Database Recovery"
- Chapter 18, "Performing Block Media Recovery"
- *Oracle Database Administrator's Guide* to learn about ADR

Checking for Block Corruption with the VALIDATE Command

You can use the `VALIDATE` command to manually check for physical and logical corruptions in database files. This command performs the same types of checks as `BACKUP VALIDATE`, but `VALIDATE` can check a larger selection of objects. For example, you can validate individual blocks with the `VALIDATE DATAFILE ... BLOCK` command.

When validating whole files, RMAN checks every block of the input files. If the backup validation discovers corrupt blocks, then RMAN updates the `V$DATABASE_BLOCK_CORRUPTION` view with rows describing the corruptions.

Use `VALIDATE BACKUPSET` when you suspect that one or more backup pieces in a backup set are missing or have been damaged. This command checks every block in a backup set to ensure that the backup can be restored. If RMAN finds block corruption, then it issues an error and terminates the validation. Note that `VALIDATE BACKUPSET` enables you to choose which backups to check, whereas the `VALIDATE` option of the `RESTORE` command lets RMAN choose.

To use VALIDATE to check database files and backups:

1. Start RMAN and connect to a target database.
2. Execute the `VALIDATE` command with the desired options.

   For example, to validate all datafiles and control files (and the server parameter file if one is in use), execute the following command at the RMAN prompt:

   ```
   RMAN> VALIDATE DATABASE;
   ```

   Alternatively, you can validate a particular backup set by using the form of the command shown in the following example (sample output included).

   ```
   RMAN> VALIDATE BACKUPSET 22;

   Starting validate at 17-AUG-06
   using channel ORA_DISK_1
   allocated channel: ORA_SBT_TAPE_1
   channel ORA_SBT_TAPE_1: SID=89 device type=SBT_TAPE
   channel ORA_SBT_TAPE_1: Oracle Secure Backup
   channel ORA_DISK_1: starting validation of datafile backup set
   channel ORA_DISK_1: reading from backup piece
   /disk1/oracle/work/orcva/RDBMS/backupset/2007_08_16/o1_mf_nnndf_TAG20070816T153
   034_2g774bt2_.bkp
   channel ORA_DISK_1: piece
   handle=/disk1/oracle/work/orcva/RDBMS/backupset/2007_08_16/o1_mf_nnndf_TAG20070
   816T153034_2g774bt2_.bkp tag=TAG20070816T153034
   channel ORA_DISK_1: restored backup piece 1
   channel ORA_DISK_1: validation complete, elapsed time: 00:00:01
   Finished validate at 17-AUG-06
   ```

   The following example illustrates how you can check individual data blocks within a datafile for corruption.

```
RMAN> VALIDATE DATAFILE 1 BLOCK 10;

Starting validate at 17-AUG-06
using channel ORA_DISK_1
channel ORA_DISK_1: starting validation of datafile
channel ORA_DISK_1: specifying datafile(s) for validation
input datafile file number=00001 name=/disk1/oracle/dbs/tbs_01.f
channel ORA_DISK_1: validation complete, elapsed time: 00:00:01
List of Datafiles
=================
File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
1    OK     0              2            127             481907
  File Name: /disk1/oracle/dbs/tbs_01.f
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              36
  Index      0              31
  Other      0              58

Finished validate at 17-AUG-06
```

Parallelizing the Validation of a Datafile

If you need to validate a large datafile, then RMAN can parallelize the work by dividing the file into sections and processing each **file section** in parallel. If multiple channels are configured or allocated, and if you want the channels to parallelize the validation, then specify the `SECTION SIZE` parameter of the `VALIDATE` command.

If you specify a section size that is larger than the size of the file, then RMAN does not create file sections. If you specify a small section size that would produce more than 256 sections, then RMAN increases the section size to a value that results in exactly 256 sections.

To parallelize the validation of a datafile:

1. Start RMAN and connect to a target database. The target database must be mounted or open.
2. Run `VALIDATE` with the `SECTION SIZE` parameter.

   The following example allocates two channels and validates a large datafile. The section size is 1200 MB.

   ```
   RUN
   {
     ALLOCATE CHANNEL c1 DEVICE TYPE DISK;
     ALLOCATE CHANNEL c2 DEVICE TYPE DISK;
     VALIDATE DATAFILE 1 SECTION SIZE 1200M;
   }
   ```

See Also:

- "Dividing the Backup of a Large Datafile into Sections" on page 9-2
- *Oracle Database Backup and Recovery Reference* to learn about the `VALIDATE` command

Validating Database Files with BACKUP VALIDATE

You can use the BACKUP VALIDATE command to do the following:

- Check datafiles for physical and logical block corruption
- Confirm that all database files exist and are in the correct locations

When you run BACKUP VALIDATE, RMAN reads the files to be backed up in their entirety, as it would during a real backup. RMAN does not, however, actually produce any backup sets or image copies.

You cannot use the BACKUPSET, MAXCORRUPT, or PROXY parameters with BACKUP VALIDATE. To validate specific backup sets, run the VALIDATE command.

To validate files with the BACKUP VALIDATE command:

1. Start RMAN and connect to a target database and recovery catalog (if used).
2. Run the BACKUP VALIDATE command.

   For example, you can validate that all database files and archived logs can be backed up by running a command as shown in the following example. This command checks for physical corruptions only.

   ```
   BACKUP VALIDATE
     DATABASE
     ARCHIVELOG ALL;
   ```

   To check for logical corruptions in addition to physical corruptions, run the following variation of the preceding command:

   ```
   BACKUP VALIDATE
     CHECK LOGICAL
     DATABASE
     ARCHIVELOG ALL;
   ```

   In the preceding examples, the RMAN client displays the same output that it would if it were really backing up the files. If RMAN cannot back up one or more of the files, then it issues an error message. For example, RMAN may show output similar to the following:

   ```
   RMAN-00571: ===========================================================
   RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
   RMAN-00571: ===========================================================
   RMAN-03002: failure of backup command at 08/29/2007 14:33:47
   ORA-19625: error identifying file /oracle/oradata/trgt/arch/archive1_6.dbf
   ORA-27037: unable to obtain file status
   SVR4 Error: 2: No such file or directory
   Additional information: 3
   ```

See Also:

- *Oracle Database Backup and Recovery Reference* for BACKUP syntax
- Chapter 18, "Performing Block Media Recovery"to learn how to repair corrupt blocks discovered by BACKUP VALIDATE

Validating Backups Before Restoring Them

You can run RESTORE . . . VALIDATE to test whether RMAN can restore a specific file or set of files from a backup. RMAN chooses which backups to use.

The database must be mounted or open for this command. You do not have to take datafiles offline when validating the restore of datafiles, because validation of backups of the datafiles only reads the backups and does not affect the production datafiles.

When validating files on disk or tape, RMAN reads all blocks in the backup piece or image copy. RMAN also validates offsite backups. The validation is identical to a real restore operation except that RMAN does not write output files.

> **Note:** As an additional test measure, you can perform a **trial recovery** with the `RECOVER ... TEST` command. A trial recovery applies redo in a way similar to normal recovery, but it is in memory only and it rolls back its changes after the trial.

To validate backups with the `RESTORE` command:

1. Run the `RESTORE` command with the `VALIDATE` option.

   This following example illustrates validating the restore of the database and all archived redo logs:

   ```
   RESTORE DATABASE VALIDATE;
   RESTORE ARCHIVELOG ALL VALIDATE;
   ```

   If you do *not* see an RMAN error stack, then skip the subsequent steps. The lack of error messages means that RMAN had confirmed that it can use these backups successfully during a real restore and recovery.

2. If you see error messages in the output and the `RMAN-06026` message, then investigate the cause of the problem. If possible, correct the problem that is preventing RMAN from validating the backups and retry the validation.

   The following error means that RMAN cannot restore one or more of the specified files from your available backups:

   ```
   RMAN-06026: some targets not found - aborting restore
   ```

   The following sample output shows that RMAN encountered a problem reading the specified backup:

   ```
   RMAN-03009: failure of restore command on c1 channel at 12-DEC-06 23:22:30
   ORA-19505: failed to identify file "oracle/dbs/1fafv9gl_1_1"
   ORA-27037: unable to obtain file status
   SVR4 Error: 2: No such file or directory
   Additional information: 3
   ```

   > **See Also:** *Oracle Database Backup and Recovery Reference* to learn about the `RESTORE ... VALIDATE` command

16

Performing Flashback and Database Point-in-Time Recovery

This chapter explains how to investigate unwanted database changes, and select and carry out an appropriate recovery strategy based upon Oracle Flashback Technology and database backups. It includes the following topics:

- Overview of Flashback Technology and Database Point-in-Time Recovery
- Rewinding a Table with Flashback Table
- Rewinding a DROP TABLE Operation with Flashback Drop
- Rewinding a Database with Flashback Database
- Performing Database Point-in-Time Recovery
- Flashback and Database Point-in-Time Recovery Scenarios

Overview of Flashback Technology and Database Point-in-Time Recovery

This section explains the purpose and basic concepts of Flashback Technology and database point-in-time recovery.

Purpose of Flashback and Database Point-in-Time-Recovery

Typically, the following situations call for flashback features or point-in-time recovery:

- A user error or corruption removes needed data or introduces corrupted data. For example, a user or DBA might erroneously delete or update the contents of one or more tables, drop database objects that are still needed during an update to an application, or run a large batch update that fails midway.
- A database upgrade fails or an upgrade script goes awry.
- A complete database recovery after a media failure cannot succeed because you do not have all of the needed redo logs or incremental backups.

In either situation, you can use point-in-time recovery or flashback features to return the database or database object to its state at a previous point in time.

Basic Concepts of Point-in-Time Recovery and Flashback Features

The most basic solution to unwanted database changes is RMAN **database point-in-time recovery (DBPITR)**. DBPITR is sometimes called **incomplete recovery** because it does not use all of the available redo or completely recover all changes to your database. In this case, you restore a whole database backup and then apply redo

logs or incremental backups to re-create all changes up to a point in time before the unwanted change.

If unwanted database changes are extensive but confined to specific tablespaces, then you can use **tablespace point-in-time recovery (TSPITR)** to return these tablespaces to an earlier SCN while the unaffected tablespaces remain available. RMAN TSPITR is an advanced technique described in Chapter 20, "Performing RMAN Tablespace Point-in-Time Recovery (TSPITR)".

Oracle Database also provides a set of features collectively known as Flashback Technology that supports viewing past states of data, and winding and rewinding data back and forth in time, without requiring the restore of the database from backup. Depending on the changes to your database, Flashback Technology can often reverse the unwanted changes more quickly and with less impact on database availability.

Basic Concepts of Database Point-in-Time Recovery

DBPITR works at the physical level to return the datafiles to their state at a target time in the past. In an RMAN DBPITR operation, you specify a target SCN, log sequence, restore point, or time. RMAN restores the database from backups created before the target time, and then applies incremental backups and logs to re-create all changes between the time of the datafile backups and the end point of recovery. When the end point is specified as an SCN, the database applies the redo logs and stops at the end of each redo thread or the specified SCN, whichever occurs first. When the end point is specified as a time, the database internally determines a suitable SCN for the specified time and then recovers to this SCN.

If your backup strategy is properly designed and your database is running in `ARCHIVELOG` mode, then DBPITR is an option in nearly all circumstances. RMAN simplifies DBPITR in comparison to the user-managed DBPITR described in "Performing Incomplete Database Recovery" on page 28-13. Given a target SCN, datafiles are restored from backup and recovered efficiently with no intervention from the user. Nevertheless, RMAN DBPITR has the following disadvantages:

- You cannot return selected objects to their earlier state, only the entire database.
- Your entire database is unavailable during the DBPITR.
- DBPITR can be time-consuming because RMAN must restore all datafiles. Also, RMAN may need to restore redo logs and incremental backups to recover the datafiles. If backups are on tape, then this process can take even longer.

Basic Concepts of Oracle Flashback Technology

The flashback features of Oracle are more efficient than media recovery in most circumstances in which they are available. You can use them to investigate past states of the database.

Physical Flashback Features Useful in Backup and Recovery **Oracle Flashback Database**, which is explained in "Rewinding a Database with Flashback Database" on page 16-11, is the most efficient alternative to DBPITR. Unlike the other flashback features, it operates at a physical level and reverts the current datafiles to their contents at a past time. The result is like the result of a DBPITR, including the `OPEN RESETLOGS`, but Flashback Database is typically faster because it does not require you to restore datafiles and requires only limited application of redo compared to media recovery.

As explained in "Configuring the Flash Recovery Area" on page 5-13, a **flash recovery area** is required for Flashback Database. To enable logging for Flashback Database, you must set the `DB_FLASHBACK_RETENTION_TARGET` initialization parameter and issue the `ALTER DATABASE FLASHBACK ON` statement.

During normal operation, the database periodically writes old images of datafile blocks to the **flashback logs**. Flashback logs are written sequentially and often in bulk. In some respects, flashback logging is like a continuous backup. The database automatically creates, deletes, and resizes flashback logs in the recovery area. Flashback logs are not archived. You need only be aware of flashback logs for monitoring performance and determining disk space allocation for the recovery area.

When you perform a Flashback Database operation, the database uses flashback logs to access past versions of data blocks and also uses some data in the archived redo logs. Consequently, you cannot enable Flashback Database *after* a failure is discovered and then use Flashback Database to rewind through this failure. Note that you can use the related capability of guaranteed restore points to protect the contents of your database at a fixed point in time, such as immediately before a risky database change.

Logical Flashback Features Useful in Backup and Recovery The remaining flashback features operate at the logical level. The logical features documented in this chapter are as follows:

- Oracle Flashback Table

  You can recover a table or set of tables to a specified point in time in the past without taking any part of the database offline. In many cases, Flashback Table eliminates the need to perform more complicated point-in-time recovery operations. Flashback Table restores tables while automatically maintaining associated attributes such as current indexes, triggers and constraints, and not requiring you to find and restore application-specific properties.

  "Rewinding a Table with Flashback Table" on page 16-4 explains how to use this feature.

- Oracle Flashback Drop

  You can reverse the effects of a `DROP TABLE` statement.

  "Rewinding a DROP TABLE Operation with Flashback Drop" on page 16-7 explains how to use this feature.

> **Note:** Because the logical flashback features have uses not specific to backup and recovery, some of the documentation for them is located elsewhere in the documentation set.

All logical flashback features except Flashback Drop rely on **undo data**. Used primarily for providing read consistency for SQL queries and rolling back transactions, undo records contain the information required to reconstruct data as it existed at a past time and examine the record of changes since that past time.

Flashback Drop relies on a mechanism called the **recycle bin**, which the database uses to manage dropped database objects until the space they occupied is needed for new data. There is no fixed amount of space allocated to the recycle bin, and no guarantee as to how long dropped objects remain in the recycle bin. Depending on system activity, a dropped object may remain in the recycle bin for seconds or for months.

See Also:

- *Oracle Database Concepts* and *Oracle Database Administrator's Guide* for more information about undo data and automatic undo management
- *Oracle Database Advanced Application Developer's Guide* to learn how to use the logical flashback features
- "Configuring Oracle Flashback Database and Restore Points" *on page 5-27* for more information on setting up your database to use Flashback Database, and on the related restore points feature

Rewinding a Table with Flashback Table

Flashback Table uses information in the undo tablespace rather than restored backups to retrieve the table. When a Flashback Table operation occurs, new rows are deleted and old rows are reinserted. The rest of your database remains available while the flashback of the table is being performed.

See Also: *Oracle Database Administrator's Guide* for more information on Automatic Undo Management

Prerequisites for Flashback Table

To use the Flashback Table feature on one or more tables, use the `FLASHBACK TABLE` SQL statement with a target time or SCN.

You must have the following privileges to use the Flashback Table feature:

- You must have been granted the `FLASHBACK ANY TABLE` system privilege or you must have the `FLASHBACK` object privilege on the table.
- You must have `SELECT`, `INSERT`, `DELETE`, and `ALTER` privileges on the table.
- To flash back a table to a restore point, you must have the `SELECT ANY DICTIONARY` or `FLASHBACK ANY TABLE` system privilege or the `SELECT_CATALOG_ROLE` role.

For an object to be eligible to be flashed back, the following prerequisites must be met:

- The object must **not** be included the following categories: tables that are part of a cluster, materialized views, Advanced Queuing (AQ) tables, static data dictionary tables, system tables, remote tables, object tables, nested tables, or individual table partitions or subpartitions.
- The structure of the table must not have been changed between the current time and the target flash back time.

  The following DDL operations change the structure of a table: upgrading, moving, or truncating a table; adding a constraint to a table, adding a table to a cluster; modifying or dropping a column; adding, dropping, merging, splitting, coalescing, or truncating a partition or subpartition (with the exception of adding a range partition).
- Row movement must be enabled on the table, which indicates that rowids will change after the flashback occurs.

  This restriction exists because if rowids before the flashback were stored by the application, then there is no guarantee that the rowids will correspond to the same

rows after the flashback. If your application depends on rowids, then you cannot use Flashback Table.

- The undo data in the **undo tablespace** must extend far enough back in time to satisfy the flashback target time or SCN.

  The point to which you can perform Flashback Table is determined by the **undo retention period**, which is the minimal time for which undo data will be kept before being recycled, and tablespace characteristics. The undo data contains information about data blocks before they were changed. The flashback operation uses undo to re-create the original data.

  To ensure that the undo information is retained for Flashback Table operations, Oracle suggests setting the `UNDO_RETENTION` parameter to 86400 seconds (24 hours) or greater for the undo tablespace.

> **Note:** `FLASHBACK TABLE ... TO BEFORE DROP` is a use of the Flashback Drop feature, not Flashback Table, and therefore is not subject to these prerequisites. See "Rewinding a DROP TABLE Operation with Flashback Drop" on page 16-7 for more information.

Performing a Flashback Table Operation

In this scenario, assume that you want to perform a flashback of the `hr.temp_employees` table after a user made a number of incorrect updates.

The perform a flashback of `temp_employees`:

1. Connect SQL*Plus to the target database and identify the current SCN.

   You cannot roll back a `FLASHBACK TABLE` statement, but you can issue another `FLASHBACK TABLE` statement and specify a time just prior to the current time. Therefore, it is advisable to record the current SCN. You can obtain it by querying `V$DATABASE` as follows:

   ```
   SELECT CURRENT_SCN
   FROM   V$DATABASE;
   ```

2. Identify the time, SCN, or restore point to which you want to return the table.

   If you have created restore points, then you can list available restore points by executing the following query:

   ```
   SELECT NAME, SCN, TIME
   FROM   V$RESTORE_POINT;
   ```

3. Ensure that enough undo data exists to rewind the table to the specified target.

   If the `UNDO_RETENTION` intialization parameter is set, and the undo retention guarantee is on, then you can use the following query to determine how long undo data is being retained:

   ```
   SELECT NAME, VALUE/60 MINUTES_RETAINED
   FROM   V$PARAMETER
   WHERE  NAME = 'undo_retention';
   ```

4. Ensure that row movement is enabled for all objects that you are rewinding with Flashback Table.

   You can enable row movement for a table with the following SQL statement, where *table* is the name of the table that you are rewinding:

```
ALTER TABLE table ENABLE ROW MOVEMENT;
```

5. Determine whether the table that you intend to flash back has dependencies on other tables. If dependencies exist, then decide whether to flash back these tables as well.

   You can issue the following SQL query to determine the dependencies, where *schema_name* is the schema for the table to be flashed back and *table_name* is the name of the table:

```
SELECT other.owner, other.table_name
FROM   sys.all_constraints this, sys.all_constraints other
WHERE  this.owner = schema_name
AND    this.table_name = table_name
AND    this.r_owner = other.owner
AND    this.r_constraint_name = other.constraint_name
AND    this.constraint_type='R';
```

6. Execute a `FLASHBACK TABLE` statement for the objects that you want to flash back.

   The following SQL statement returns the `hr.temp_employees` table to the restore point named `temp_employees_update`:

```
FLASHBACK TABLE hr.temp_employees
  TO RESTORE POINT temp_employees_update;
```

   The following SQL statement rewinds the `hr.temp_employees` table to its state when the database was at the time specified by the SCN:

```
FLASHBACK TABLE hr.temp_employees
  TO SCN 123456;
```

   As shown in the following example, you can also specify the target point in time with `TO_TIMESTAMP`:

```
FLASHBACK TABLE hr.temp_employees
  TO TIMESTAMP TO_TIMESTAMP('2007-10-17 09:30:00', 'YYYY-MM-DD HH:MI:SS');
```

   Note: The mapping of timestamps to SCNs is not always exact. When using timestamps with the `FLASHBACK TABLE` statement, the time to which the table is flashed back can vary by up to approximately three seconds of the time specified for `TO_TIMESTAMP`. If an exact point in time is required, then use an SCN rather than a time.

7. Optionally, query the table to check the data.

Keeping Triggers Enabled During Flashback Table

By default, the database disables triggers on the affected table before performing a `FLASHBACK TABLE` operation. After the operation, the database returns the triggers to the state they were in before the operation (enabled or disabled). To keep triggers enabled during the flashback of the table, add an `ENABLE TRIGGERS` clause to the `FLASHBACK TABLE` statement in step 6 on page 16-6.

For example, assume that at 17:00 an HR administrator discovers that an employee is missing from the `hr.temp_employees` table. This employee was included in the table at 14:00, the last time the report was run. Therefore, someone accidentally deleted

the record for this employee between 14:00 and 17:00. She uses Flashback Table to return the table to its state at 14:00, respecting any triggers set on the `hr.temp_employees` table, by using the SQL statement in the following example:

```
FLASHBACK TABLE temp_employees
  TO TIMESTAMP TO_TIMESTAMP('2005-03-03 14:00:00' , 'YYYY-MM-DD HH:MI:SS')
  ENABLE TRIGGERS;
```

See Also:

- *Oracle Database Administrator's Guide* to learn how to recover tables with the Flashback Table feature
- *Oracle Database SQL Language Reference* for a simple Flashback Table scenario

Rewinding a DROP TABLE Operation with Flashback Drop

This section explains how to retrieve objects from the recycle bin by means of the `FLASHBACK TABLE ... TO BEFORE DROP` statement.

About Flashback Drop

Flashback Drop reverses the effects of a `DROP TABLE` operation. Flashback Drop is faster than other recovery mechanisms that can be used in this situation, such as point-in-time recovery, and does not lead to downtime or loss of recent transactions.

When you drop a table, the database does not immediately remove the space associated with the table. Instead, the table is renamed and, along with any associated objects, placed in the recycle bin. System-generated recycle bin object names are unique. You can query objects in the recycle bin, just as you can query other objects.

A flashback operation retrieves the table from the recycle bin. When retrieving dropped tables, you can specify either the original user-specified name of the table or the system-generated name.

When you drop a table, the table and all of its dependent objects go into the recycle bin together. Likewise, when you perform Flashback Drop, the objects are generally all retrieved together. When you restore a table from the recycle bin, dependent objects such as indexes do not get their original names back; they retain their system-generated recycle bin names. Oracle Database retrieves all indexes defined on the table except for bitmap join indexes, and all triggers and constraints defined on the table except for referential integrity constraints that reference other tables.

It is possible that some dependent objects such as indexes may have been reclaimed because of space pressure. In such cases, the reclaimed dependent objects are not retrievable from the recycle bin.

Prerequisites of Flashback Drop

The following list summarizes the user privileges required for the operations related to Flashback Drop and the recycle bin:

- `DROP`

  Any user with drop privileges over an object can drop the object, placing it in the recycle bin.

- `FLASHBACK TABLE ... TO BEFORE DROP`

Privileges for this statement are tied to the privileges for DROP. That is, any user who can drop an object can perform Flashback Drop to retrieve the dropped object from the recycle bin.

- PURGE

  Privileges for a purge of the recycle bin are tied to the DROP privileges. Any user having DROP TABLE or DROP ANY TABLE privileges can purge the objects from the recycle bin.

- SELECT for objects in the Recycle Bin

  Users must have SELECT and FLASHBACK privileges over an object in the recycle bin to query the object in the recycle bin. Any users who had the SELECT privilege over an object before it was dropped continue to have the SELECT privilege over the object in the recycle bin. Users must have FLASHBACK privilege to query any object in the recycle bin because these are objects from a past state of the database.

Objects must meet the following prerequisites to be eligible for retrieval from the recycle bin:

- The recycle bin is only available for non-system, locally managed tablespaces. If a table is in a non-system, locally managed tablespace, but one or more of its dependent segments (objects) is in a dictionary-managed tablespace, then these objects are protected by the recycle bin.
- Tables that have Fine-Grained Auditing (FGA) and Virtual Private Database (VPD) policies defined over them are not protected by the recycle bin.
- Partitioned index-organized tables are not protected by the recycle bin.
- The table must not have been purged, either by a user or by Oracle Database as a result of a space reclamation operation.

Performing a Flashback Drop Operation

Use the FLASHBACK TABLE . . . TO BEFORE DROP statement to recover objects from the recycle bin. You can specify either the name of the table in the recycle bin or the original table name.

This section assumes a scenario in which you drop the wrong table. Many times you have been asked to drop tables in the test databases, but in this case you accidentally connect to the production database instead and drop hr.employee_demo. You decide to use FLASHBACK TABLE to retrieve the dropped object.

To retrieve a dropped table:

1. Connect SQL*Plus to the target database and obtain the name of the dropped table in the recycle bin.

   You can use the SQL*Plus command SHOW RECYCLEBIN as follows:

```
SHOW RECYCLEBIN;

ORIGINAL NAME    RECYCLEBIN NAME                 TYPE         DROP TIME
---------------- ------------------------------- ------------ -------------
EMPLOYEE_DEMO    BIN$gk3lsj/3akk5hg3j2lkl5j3d==$0  TABLE    2005-04-11:17:08:54
```

   The ORIGINAL NAME column shows the original name of the object, while the RECYCLEBIN NAME column shows the name of the object as it exists in the bin.

Alternatively, you can query USER_RECYCLEBIN or DBA_RECYCLEBIN to obtain the table name. The following example queries the views to determine the original names of dropped objects:

```
SELECT object_name AS recycle_name, original_name, type
FROM   recyclebin;

RECYCLE_NAME                     ORIGINAL_NAME          TYPE
-------------------------------  ---------------------  ----------
BIN$gk3lsj/3akk5hg3j2lkl5j3d==$0  EMPLOYEE_DEMO          TABLE
BIN$JKS983293M1dsab4gsz/I249==$0  I_EMP_DEMO            INDEX
```

If you plan to manually restore original names for dependent objects, then ensure that you make note of each dependent object's system-generated recycle bin name before you restore the table.

Note: Object views such as DBA_TABLES do not display the recycle bin objects.

---

2. Optionally, query the table in the recycle bin.

   You must use the recycle bin name of the object in your query rather than the object's original name. The following example queries the table with the recycle bin name of BIN$KSD8DB9L345KLA==$0:

   ```
   SELECT *
   FROM   "BIN$gk3lsj/3akk5hg3j2lkl5j3d==$0";
   ```

   Quotes are required because of the special characters in the recycle bin name.

   Note: If you have the necessary privileges, then you can also use Oracle Flashback Query on tables in the recycle bin, but only by using the recycle bin name rather than the original table name. You cannot use DML or DDL statements on objects in the recycle bin.

   ---

3. Retrieve the dropped table.

   Use the FLASHBACK TABLE ... TO BEFORE DROP statement. The following example restores the BIN$gk3lsj/3akk5hg3j2lkl5j3d==$0 table, changes its name back to hr.employee_demo, and purges its entry from the recycle bin:

   ```
   FLASHBACK TABLE "BIN$gk3lsj/3akk5hg3j2lkl5j3d==$0" TO BEFORE DROP;
   ```

   Note that the table name is enclosed in quotes because of the possibility of special characters appearing in the recycle bin object names.

   Alternatively, you can use the original name of the table:

   ```
   FLASHBACK TABLE HR.EMPLOYEE_DEMO TO BEFORE DROP;
   ```

   You can also assign a new name to the restored table by specifying the RENAME TO clause. For example:

   ```
   FLASHBACK TABLE "BIN$KSD8DB9L345KLA==$0" TO BEFORE DROP
     RENAME TO hr.emp_demo;
   ```

4. Optionally, verify that all dependent objects retained their system-generated recycle bin names.

The following query determines the names of the indexes of the retrieved `hr.employee_demo` table:

```
SELECT INDEX_NAME
FROM   USER_INDEXES
WHERE  TABLE_NAME = 'EMPLOYEE_DEMO';

INDEX_NAME
------------------------------
BIN$JKS983293M1dsab4gsz/I249==$0
```

5. Optionally, rename the retrieved indexes to their original names.

   The following statement renames the index to its original name of `i_emp_demo`:

   ```
   ALTER INDEX "BIN$JKS983293M1dsab4gsz/I249==$0" RENAME TO I_EMP_DEMO;
   ```

6. If the retrieved table had referential constraints before it was placed in the recycle bin, then re-create them.

   This step must be performed manually because the recycle bin does not preserve referential constraints on a table.

Retrieving Objects When Multiple Objects Share the Same Original Name

You can create, and then drop, several objects with the same original name. All the dropped objects will be stored in the recycle bin. For example, consider the SQL statements in the following example.

Example 16–1 Dropping Multiple Objects with the Same Name

```
CREATE TABLE temp_employees ( ...columns ); # temp_employees version 1
DROP TABLE temp_employees;

CREATE TABLE temp_employees ( ...columns ); # temp_employees version 2
DROP TABLE temp_employees;

CREATE TABLE temp_employees ( ...columns ); # temp_employees version 3
DROP TABLE temp_employees;
```

In Example 16–1, each table `temp_employees` is assigned a unique name in the recycle bin when it is dropped. You can use a `FLASHBACK TABLE ... TO BEFORE DROP` statement with the original name of the table, as shown in this example:

```
FLASHBACK TABLE temp_employees TO BEFORE DROP;
```

The most recently dropped table with this original name is retrieved from the recycle bin, with its original name. Example 16–2 shows the retrieval from the recycle bin of all three dropped `temp_employees` tables from the previous example, with each assigned a new name.

Example 16–2 Renaming Dropped Tables

```
FLASHBACK TABLE temp_employees TO BEFORE DROP
  RENAME TO temp_employees_VERSION_3;
FLASHBACK TABLE temp_employees TO BEFORE DROP
  RENAME TO temp_employees_VERSION_2;
FLASHBACK TABLE temp_employees TO BEFORE DROP
  RENAME TO temp_employees_VERSION_1;
```

Because the original name in FLASHBACK TABLE refers to the most recently dropped table with this name, the last table dropped is the first retrieved.

You can also retrieve any table from the recycle bin, regardless of any collisions among original names, by using the unique recycle bin name of the table. For example, assume that you query the recycle bin as follows (sample output included):

```
SELECT object_name, original_name, createtime
FROM   recyclebin;

OBJECT_NAME                    ORIGINAL_NAME   CREATETIME
------------------------------ --------------- -------------------
BIN$yrMKlZaLMhfgNAgAIMenRA==$0 TEMP_EMPLOYEES  2007-02-05:21:05:52
BIN$yrMKlZaVMhfgNAgAIMenRA==$0 TEMP_EMPLOYEES  2007-02-05:21:25:13
BIN$yrMKlZaQMhfgNAgAIMenRA==$0 TEMP_EMPLOYEES  2007-02-05:22:05:53
```

You can use the following command to retrieve the middle table:

```
FLASHBACK TABLE BIN$yrMKlZaVMhfgNAgAIMenRA==$0 TO BEFORE DROP;
```

See Also:

- *Oracle Database Administrator's Guide* to learn how to use Flashback Drop and manage the recycle bin
- *Oracle Database SQL Language Reference* for information about the FLASHBACK TABLE statement

Rewinding a Database with Flashback Database

This section explains the most common scenario for using Flashback Database to reverse unwanted changes to your database.

Prerequisites of Flashback Database

To use the FLASHBACK DATABASE command to return your database contents to points in time within the flashback window, your database must have been previously configured for flashback logging as described in "Configuring Oracle Flashback Database and Restore Points". To return the database to a guaranteed restore point, you must have previously defined a guaranteed restore point as described in "Creating Normal and Guaranteed Restore Points" on page 5-33.

Flashback Database works by undoing changes to the datafiles that exist at the moment that you run the command. Note the following important prerequisites:

- No current datafiles are lost or damaged. You can only use FLASHBACK DATABASE to rewind changes to a datafile made by an Oracle database, not to repair media failures or recover from accidental deletion of datafiles.
- You are not trying to use FLASHBACK DATABASE to return to a point in time before the restore or re-creation of a control file. If the database control file is restored from backup or re-created, then all accumulated flashback log information is discarded.
- You are not trying to use FLASHBACK DATABASE to undo a resize datafile operation. Note that shrinking a database object such as a table does not affect your ability to use FLASHBACK DATABASE.

See Also: *Oracle Database Backup and Recovery Reference* for a complete list of command prerequisites and usage notes for FLASHBACK DATABASE

Performing a Flashback Database Operation

This section presents a basic technique for performing a flashback of the database in almost all cases, specifying the desired target point in time with a time expression, the name of a normal or guaranteed restore point, or an SCN.

This scenario assumes that you are rewinding the database to a point in time within the current database **incarnation**. To return the database to the point in time immediately before the most recent `OPEN RESETLOGS`, see "Rewinding an OPEN RESETLOGS Operation with Flashback Database" on page 16-17.

By default, an SCN used in a `FLASHBACK DATABASE` command refers to an SCN in the **direct ancestral path** of the database incarnations. As explained in "Database Incarnations" on page 13-5, an incarnation is in this path if it was not abandoned after the database was previously opened with the `RESETLOGS` option. To retrieve changes in abandoned incarnations, see "Rewinding the Database to an SCN in an Abandoned Incarnation Branch" on page 16-18.

To perform a Flashback Database operation:

1. Connect SQL*Plus to the target database and determine the desired SCN, restore point, or point in time for the `FLASHBACK DATABASE` command.

   Obtain the earliest SCN in the flashback database window as follows:

   ```
   SELECT OLDEST_FLASHBACK_SCN, OLDEST_FLASHBACK_TIME
   FROM   V$FLASHBACK_DATABASE_LOG;
   ```

   The most recent SCN that can be reached with Flashback Database is the current SCN of the database. The following query returns the current SCN:

   ```
   SELECT CURRENT_SCN
   FROM   V$DATABASE;
   ```

   You can query available guaranteed restore points as follows (sample output included):

   ```
   SELECT NAME, SCN, TIME, DATABASE_INCARNATION#,
          GUARANTEE_FLASHBACK_DATABASE
   FROM   V$RESTORE_POINT
   WHERE  GUARANTEE_FLASHBACK_DATABASE='YES';

   NAME                 SCN TIME                 DATABASE_INCARNATION# GUA
   --------------- ---------- -------------------- --------------------- ---
   BEFORE_CHANGES     5753126 04-MAR-05 12.39.45 AM                     2 YES
   ```

   > **Note:** If the flashback window does not extend far enough back into the past to reach the desired target time, and if you do not have a guaranteed restore point at the desired time, then you can achieve similar results by using database point-in-time recovery, as described in "Performing Database Point-in-Time Recovery" on page 16-14.

2. Shut down the database consistently, ensure that it is not opened by any instance, and then mount it:

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT;
   ```

3. Repeat the query in the first step in this procedure.

Some flashback logging data is generated when the database is shut down. If flashback logs were deleted due to space pressure in the flash recovery area, then it is possible that your target SCN is no longer reachable.

> **Note:** If you run `FLASHBACK DATABASE` when your target SCN is outside the flashback window, then `FLASHBACK DATABASE` fails with an `ORA-38729` error. In this case your database will not be changed.

4. Start RMAN and connect to the target database.
5. Run the `SHOW` command to see which channels are preconfigured.

   During the flashback operation, RMAN may need to restore archived redo logs from backup. Enter the following command to see whether channels are configured (sample output is included):

```
SHOW ALL;

RMAN configuration parameters for database with db_unique_name PROD1 are:
.
.
.
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DEVICE TYPE SBT_TAPE PARALLELISM 1 BACKUP TYPE TO BACKUPSET; #
default
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS  "SBT_
LIBRARY=/usr/local/oracle/backup/lib/libobk.so";
```

   If the necessary devices and channels are already configured, then no action is necessary. Otherwise, use the `CONFIGURE` command to configure automatic channels, or include `ALLOCATE CHANNEL` commands within a `RUN` block.
6. Run the RMAN `FLASHBACK DATABASE` command.

   You can specify the target time by using one of the forms of the command shown in the following examples:

```
FLASHBACK DATABASE TO SCN 46963;

FLASHBACK DATABASE
  TO RESTORE POINT BEFORE_CHANGES;

FLASHBACK DATABASE TO TIME
  "TO_DATE('09/20/05','MM/DD/YY')";
```

   When the `FLASHBACK DATABASE` command completes, the database is left mounted and recovered to the specified target time.
7. Open the database read-only in SQL*Plus and run some queries to verify the database contents.

   Open the database read-only as follows:

```
ALTER DATABASE OPEN READ ONLY;
```

   If you are satisfied with the state of the database, then end the procedure with step 8. If you are *not* satisfied with the state of the database, skip to step 9.
8. If satisfied with the results, then perform either of the following mutually exclusive actions:

- Make the database available for updates by opening the database with the RESETLOGS option. If the database is currently open read-only, then execute the following commands in SQL*Plus:

```
SHUTDOWN IMMEDIATE
STARTUP MOUNT
ALTER DATABASE OPEN RESETLOGS;
```

> **Note:** After you perform this OPEN RESETLOGS operation, all changes to the database after the target SCN for FLASHBACK DATABASE are abandoned. Nevertheless, you can use the technique in "Rewinding the Database to an SCN in an Abandoned Incarnation Branch" on page 16-18 to return the database to that range of SCNs while they remain in the flashback window.

- Use Oracle Data Pump Export to make a logical backup of the objects whose state was corrupted. Afterward, use RMAN to recover the database to the present time:

```
RECOVER DATABASE;
```

  This step undoes the effect of the Flashback Database by re-applying all changes in the redo logs to the database, returning it to the most recent SCN.

  After re-opening the database read/write, you can import the exported objects with the Data Pump Import utility. See *Oracle Database Utilities* to learn how to use Data Pump.

9. If you find that you used the wrong restore point, time, or SCN for the flashback, then mount the database and perform one of the following mutually exclusive options:

   - If your chosen target time was not far enough in the past, then use another FLASHBACK DATABASE command to rewind the database further back in time:

```
FLASHBACK DATABASE TO SCN 42963;  #earlier than current SCN
```

   - If you chose a target SCN that is too far in the past, then use RECOVER DATABASE UNTIL to wind the database forward in time to the desired SCN:

```
RECOVER DATABASE UNTIL SCN 56963; #later than current SCN
```

   - If you want to completely undo the effect of the FLASHBACK DATABASE command, then you can perform complete recovery of the database by using the RECOVER DATABASE command without an UNTIL clause or SET UNTIL command:

```
RECOVER DATABASE;
```

     The RECOVER DATABASE command reapplies all changes to the database, returning it to the most recent SCN.

Performing Database Point-in-Time Recovery

RMAN DBPITR restores the database from backups prior to the target time for recovery, then uses incremental backups and redo to roll the database forward to the target time. You can recover to an SCN, time, log sequence number, or restore point. Oracle recommends that you create restore points at important times to make point-in-time recovery more manageable if it ever becomes necessary.

Oracle recommends that you perform Flashback Database rather than database point-in-time recovery if possible. Media recovery with backups should be the last option when flashback technologies cannot be used to undo the most recent changes.

Prerequisites of Database Point-in-Time Recovery

The prerequisites for database point-in-time recovery are as follows:

- Your database must be running in `ARCHIVELOG` mode.
- You must have backups of all datafiles from before the target SCN for DBPITR and archived logs for the period between the SCN of the backups and the target SCN.

For a complete account of command prerequisites and usage notes, refer to the `RECOVER` entry in *Oracle Database Backup and Recovery Reference*.

Performing Database Point-in-Time Recovery

This section explains the basic steps of DBPITR. The procedure makes the following assumptions:

- You are performing DBPITR within the current database **incarnation**. If your target time is not in the current incarnation, then see "Recovering the Database to an Ancestor Incarnation" on page 16-20 for more information on DBPITR to ancestor incarnations.
- The control file is current. If you need to restore a backup control file, then see "Performing Recovery with a Backup Control File" on page 19-4.
- Your database is using the current server parameter file. If you need to restore a backup server parameter file, then see "Restoring the Server Parameter File" on page 19-2.

When performing DBPITR, you can avoid errors by using the `SET UNTIL` command to set the target time at the beginning of the procedure, rather than specifying the `UNTIL` clause on the `RESTORE` and `RECOVER` commands individually. This ensures that the datafiles restored from backup will have timestamps early enough to be used in the subsequent `RECOVER` operation.

To perform DBPITR:

1. Determine the time, SCN, restore point, or log sequence that should end recovery.

   You can use the Flashback Query features to help you identify when the logical corruption occurred. Note that if you have a **flashback data archive** enabled for a table, then you can query data that existed far in the past.

   You can also use the alert log to try to determine the time of the event from which you need to recover.

   Alternatively, you can use a SQL query to determine the log sequence number that contains the target SCN and then recover through this log. For example, run the following query to list the logs in the current database incarnation (sample output included):

```
SELECT RECID, STAMP, THREAD#, SEQUENCE#, FIRST_CHANGE#
       FIRST_TIME, NEXT_CHANGE#
FROM   V$ARCHIVED_LOG
WHERE  RESETLOGS_CHANGE# =
       ( SELECT RESETLOGS_CHANGE#
         FROM   V$DATABASE_INCARNATION
         WHERE  STATUS = 'CURRENT');
```

```
RECID      STAMP      THREAD#    SEQUENCE#  FIRST_CHAN FIRST_TIM NEXT_CHANG
---------- ---------- ---------- ---------- ---------- --------- ----------
         1  344890611          1          1      20037 24-SEP-05      20043
         2  344890615          1          2      20043 24-SEP-05      20045
         3  344890618          1          3      20045 24-SEP-05      20046
```

For example, if you discover that a user accidentally dropped a tablespace at 9:02 a.m., then you can recover to 9 a.m., just before the drop occurred. You lose all changes to the database made after this time.

2. If you are using a target time expression instead of a target SCN, then make sure the time format environment variables are appropriate before invoking RMAN.

   The following are sample Globalization Support settings:

   ```
   NLS_LANG = american_america.us7ascii
   NLS_DATE_FORMAT="Mon DD YYYY HH24:MI:SS"
   ```

3. Connect RMAN to the target database and, if applicable, the recovery catalog database. Bring the database to a mounted state:

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT;
   ```

4. Perform the following operations within a `RUN` block:

   a. Use `SET UNTIL` to specify the target time, restore point, SCN, or log sequence number for DBPITR. If specifying a time, then use the date format specified in the `NLS_LANG` and `NLS_DATE_FORMAT` environment variables.

   b. If automatic channels are not configured, then manually allocate disk and tape channels as needed.

   c. Restore and recover the database.

   The following example performs DBPITR on the target database until SCN 1000:

   ```
   RUN
   {
     SET UNTIL SCN 1000;
     RESTORE DATABASE;
     RECOVER DATABASE;
   }
   ```

   As shown in the following examples, you can also use time expressions, restore points, or log sequence numbers to specify the `SET UNTIL` time:

   ```
   SET UNTIL TIME 'Nov 15 2004 09:00:00';
   SET UNTIL SEQUENCE 9923;
   SET UNTIL RESTORE POINT before_update;
   ```

   If the operation completes without errors, then DBPITR has succeeded.

5. Open the database read-only in SQL*Plus and perform queries as needed to ensure that the effects of the logical corruption have been reversed.

   Open the database read-only as follows:

   ```
   ALTER DATABASE OPEN READ ONLY;
   ```

   If you are satisfied with the state of the database, then end the procedure with step 6. If not satisfied, then you may have chosen the wrong target SCN. In this case,

investigate the unwanted change further and determine a new target SCN, then repeat the DBPITR procedure.

6. If satisfied with the results, then perform either of the following mutually exclusive actions:

   - Open your database for read/write, abandoning all changes after the target SCN. In this case, you must shut down the database, mount it, and then execute the following command:

     ```
     ALTER DATABASE OPEN RESETLOGS;
     ```

     Note that the OPEN RESETLOGS operation will fail if a datafile is offline unless the datafile went offline normally or is read-only. You can bring files in read-only or offline normal tablespaces online after the RESETLOGS because they do not need any redo.

   - Export one or more objects from your database with Data Pump Export. You can then recover the database to the current point in time and re-import the exported objects, thus returning these objects to their state before the unwanted change without abandoning all other changes.

Flashback and Database Point-in-Time Recovery Scenarios

This section describes variations on the basic scenarios described in "Rewinding a Database with Flashback Database" on page 16-11 and "Performing Database Point-in-Time Recovery" on page 16-14.

Rewinding an OPEN RESETLOGS Operation with Flashback Database

The procedure for using Flashback Database to reverse an unwanted ALTER DATABASE OPEN RESETLOGS statement is similar to the general case described in "Performing a Flashback Database Operation" on page 16-12. Rather than specifying a particular SCN or point in time for the FLASHBACK DATABASE command, however, you use FLASHBACK DATABASE TO BEFORE RESETLOGS.

To undo an OPEN RESETLOGS operation:

1. Connect SQL*Plus to the target database and verify that the beginning of the flashback window is earlier than the time of the most recent OPEN RESETLOGS.

   Run the following queries:

   ```
   SELECT RESETLOGS_CHANGE#
   FROM   V$DATABASE;

   SELECT OLDEST_FLASHBACK_SCN
   FROM   V$FLASHBACK_DATABASE_LOG;
   ```

   If V$DATABASE.RESETLOGS_CHANGE# is greater than V$FLASHBACK_DATABASE_LOG.OLDEST_FLASHBACK_SCN, then you can use Flashback Database to reverse the OPEN RESETLOGS.

2. Shut down the database, mount it, and recheck the flashback window. If the resetlogs SCN is still within the flashback window, then proceed to the next step.

3. Connect RMAN to the target database and perform a flashback to the SCN immediately before the RESETLOGS.

   Use the following form of the FLASHBACK DATABASE command:

```
FLASHBACK DATABASE TO BEFORE RESETLOGS;
```

As with other uses of FLASHBACK DATABASE, if the target SCN is before the beginning of the flashback database window, an error is returned and the database is not modified. If the command completes successfully, then the database is left mounted and recovered to the most recent SCN before the OPEN RESETLOGS in the previous incarnation.

4. Open the database read-only in SQL*Plus and perform queries as needed to ensure that the effects of the logical corruption have been reversed.

   Open the database read-only as follows:

   ```
   ALTER DATABASE OPEN READ ONLY;
   ```

5. To make the database available for updates again, shut down the database, mount it, and then execute the following command:

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

Undoing an OPEN RESETLOGS on Standby Databases with Flashback Database

Flashback Database across OPEN RESETLOGS may be used to perform the following functions in a Dataguard environment:

- Flashback to undo logical standby switchovers

  In this case, the database reverts to its role (primary or standby) at the target time for the Flashback Database operation.

- Undo of a physical standby activation

  You can temporarily activate a physical standby database, use it for testing or reporting purposes, and then use Flashback Database to return it to its role as a physical standby.

- Ongoing use of a standby database for testing

  The use of Flashback Database means that you do are not require the use of storage snapshots.

  > **See Also:** *Oracle Data Guard Concepts and Administration* for details on these advanced applications of Flashback Database with Data Guard

Rewinding the Database to an SCN in an Abandoned Incarnation Branch

The effect of Flashback Database or DBPITR followed by an OPEN RESETLOGS is to return the database to a previous SCN, and to abandon changes after this point. Therefore, some SCNs after that point can refer either to changes that were abandoned or changes in the current history of the database. In this way, a target SCN specified in FLASHBACK DATABASE can be ambiguous.

Unlike SCNs, time expressions and restore points are not ambiguous. A time expression is always associated with the incarnation that was current at that time. A restore point is always associated with the current incarnation when it was created. This is true even for times and restore points that correspond to abandoned database incarnations. The database incarnation is automatically reset to the incarnation that was current at the specified time or when the restore point was created.

You may want to use Flashback Database to rewind the database to an SCN in the parent incarnation that is later than the SCN of the OPEN RESETLOGS at which the

current incarnation path branched from the old incarnation. Figure 13–1, "Database Incarnation History" on page 13-7 shows how SCNs can be generated in an incarnation branch even after an OPEN RESETLOGS creates a new incarnation. As shown in the diagram, the database could be at SCN 3000 in incarnation 3 when you need to return to the abandoned SCN 1500 in incarnation 1.

If the SCN to which you are rewinding is in the **direct ancestral path**, or if you are rewinding the database to a **restore point**, then an explicit RESET DATABASE is not necessary for Flashback Database. However, an explicit RESET DATABASE TO INCARNATION command is required when using FLASHBACK DATABASE to rewind the database to an SCN in an abandoned database incarnation.

To rewind the database to an SCN in an abandoned incarnation branch:

1. Use SQL*Plus to connect to the target database and verify that the flashback logs contain enough information to flash back to the SCN.

   For example, execute the following query:

   ```
   SELECT OLDEST_FLASHBACK_SCN
   FROM   V$FLASHBACK_DATABASE_LOG;
   ```

2. Determine the target incarnation number for the Flashback Database operation, that is, the incarnation key for the parent incarnation.

   For example, execute the following query:

   ```
   SELECT PRIOR_INCARNATION#
   FROM   V$DATABASE_INCARNATION
   WHERE  STATUS = 'CURRENT';
   ```

3. Start RMAN and connect to the target database.

4. Shut down the database, and then mount it as follows:

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT;
   ```

5. Set the database incarnation to the parent incarnation.

   For example, use the following command to return to incarnation 1:

   ```
   RESET DATABASE TO INCARNATION 1;
   ```

6. Run the FLASHBACK DATABASE command, specifying the target SCN.

   For example, use the following command to rewind the database to SCN 1500:

   ```
   FLASHBACK DATABASE TO SCN 1500;
   ```

7. Open the database read-only in SQL*Plus and perform queries as needed to ensure that the effects of the logical corruption have been reversed.

   Open the database read-only as follows:

   ```
   ALTER DATABASE OPEN READ ONLY;
   ```

8. To make the database available for updates again, shut down the database, mount it, and then execute the following command:

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

See Also:

- "Database Incarnations" on page 13-5 for useful background information about database incarnations, abandoned changes, and the effects of `ALTER DATABASE OPEN RESETLOGS`
- *Oracle Database Backup and Recovery Reference* for details about the `RESET DATABASE` command

Recovering the Database to an Ancestor Incarnation

The procedure for DBPITR within the current incarnation is different from DBPITR to an SCN in a noncurrent incarnation. In the latter case, you must explicitly execute the `RESET DATABASE` to reset the database to the incarnation that was current at the target SCN. Also, you must restore a control file from the database incarnation containing the target SCN.

When RMAN is connected to a recovery catalog, a `RESTORE CONTROLFILE` command only searches the current database incarnation for the closest time specified in the `UNTIL` clause. To restore a control file from a noncurrent incarnation, you must execute `LIST INCARNATION` to identify the target database incarnation and specify this incarnation in the `RESET DATABASE TO INCARNATION` command.

When RMAN is connected to a recovery catalog, you cannot execute the `RESET DATABASE TO INCARNATION` command before the database is mounted. Thus, you must execute `SET UNTIL`, restore the control file from autobackup, and then mount it.

Assume the following situation:

- RMAN is connected to a recovery catalog.
- You have a backup of target database `trgt` from October 2, 2007.
- DBPITR was performed on this database on October 10, 2007 to correct an earlier error. The `OPEN RESETLOGS` operation at the end of that DBPITR started a new incarnation.

On October 25, you discover that you need crucial data that was dropped from the database at 8:00 a.m. on October 8, 2007. This time is prior to the beginning of the current incarnation.

To perform DBPITR to a noncurrent incarnation:

1. Start RMAN and connect to a target database and recovery catalog.
2. Determine which database incarnation was current at the time of the backup.

   Use `LIST INCARNATION` to find the primary key of the incarnation that was current at the target time:

   ```
   LIST INCARNATION OF DATABASE trgt;

   List of Database Incarnations
   DB Key  Inc Key   DB Name   DB ID       STATUS     Reset SCN    Reset Time
   ------- -------   -------   ------      -------    ----------   ----------
   1       2         TRGT      1224038686  PARENT     1            02-OCT-06
   1       582       TRGT      1224038686  CURRENT    59727        10-OCT-06
   ```

   Look at the `Reset SCN` and `Reset Time` columns to identify the correct incarnation, and note the incarnation key in the `Inc Key` column. In this example, the backup was made 2 October 2007. In this case, the incarnation key value is 2.
3. Make sure the database is started but not mounted.

```
STARTUP FORCE NOMOUNT
```

4. Reset the target database to the incarnation obtained in step 2.

   In this example, specify the incarnation current at the time of the backup of 2 October. Use the value from the `Inc Key` column to identify the incarnation.

   ```
   RESET DATABASE TO INCARNATION 2;
   ```

5. Restore and recover the database, performing the following actions in the `RUN` command:

   - Set the end time for recovery to the time just before the loss of the data.
   - Allocate any channels required that are not already configured.
   - Restore the control file from the October 2 backup and mount it.
   - Restore the datafiles and recover the database. Use the `RECOVER DATABASE ... UNTIL` command to perform DBPITR, bringing the database to the target time of 7:55 a.m. on October 8, just before the data was lost.

   The following example shows all of the steps required in this case:

   ```
   RUN
   {
     SET UNTIL TIME 'Oct 8 2007 07:55:00';
     RESTORE CONTROLFILE;
     # without recovery catalog, use RESTORE CONTROLFILE FROM AUTOBACKUP
     ALTER DATABASE MOUNT;
     RESTORE DATABASE;
     RECOVER DATABASE;
   }
   ALTER DATABASE OPEN RESETLOGS;
   ```

   See Also: *Oracle Database Backup and Recovery Reference* for details about the `RESET DATABASE` command

17

Performing Complete Database Recovery

This chapter explains how to use RMAN to return your database to normal operation after the loss of one or more datafiles. This chapter includes the following topics:

- Overview of Complete Database Recovery
- Preparing for Complete Database Recovery
- Performing Complete Database Recovery

Overview of Complete Database Recovery

This section explains the purpose of complete restore and recovery of the database and specifies the scope of the chapter.

Purpose of Complete Database Recovery

This chapter assumes that some or all of your datafiles are lost or damaged. Typically, this situation is caused by a media failure or accidental deletion. Your goal is to return the database to normal operation by restoring the damaged files from RMAN backups and recovering all database changes.

Scope of This Chapter

This chapter explain how to use complete recovery to fix the most common database problems. This chapter makes the following assumptions:

- You have lost some or all datafiles and your goal is to recover all changes, but you have not lost all current control files or an entire online redo log group.

  Chapter 16, "Performing Flashback and Database Point-in-Time Recovery" explains how to recover some but not all database changes. Chapter 29, "Performing User-Managed Recovery: Advanced Scenarios" explains how to respond when some but not all current control files or members of an online redo log group are lost. "Performing Recovery with a Backup Control File" on page 19-4 explains how to recover the database when all control files are lost.

- Your database is using the current server parameter file.

  To restore a backup server parameter file, see "Restoring the Server Parameter File" on page 19-2.

- You have the complete set of archived redo logs and incremental backups needed for recovery of your datafile backups. Every datafile either has a backup, or a complete set of online and archived redo logs goes back to the creation of a datafile with no backup.

RMAN can handle lost datafiles without user intervention during restore and recovery. When a datafile is lost, the possible cases can be classified as follows:

- The control file knows about the datafile, that is, you backed up the control file after datafile creation, but the datafile itself is not backed up. If the datafile record is in the control file, then `RESTORE` creates the datafile in the original location or in a user-specified location. The `RECOVER` command can then apply the necessary logs to the datafile.
- The control file does not have the datafile record, that is, you did not back up the control file after datafile creation. During recovery, the database will detect the missing datafile and report it to RMAN, which will create a new datafile and continue recovery by applying the remaining logs. If the datafile was created in a parent incarnation, then it will be created during the restore or recovery phase as appropriate.

- You are not restoring and recovering an encrypted tablespace.

  If you perform media recovery on an encrypted tablespace, then the Oracle wallet must be open when performing media recovery of this tablespace. See *Oracle Database Administrator's Guide* to learn about encrypted tablespaces.

- Your database runs in a single-instance configuration.

  While RMAN can restore and recover databases in Oracle RAC and Data Guard configurations, these scenarios are beyond the scope of this manual.

- You are using the RMAN client rather than Oracle Enterprise Manager.

  Enterprise Manager provides access to RMAN through a set of wizards. These wizards lead you through a variety of recovery procedures based on an analysis of your database, your available backups, and your data recovery objectives.

  By using Enterprise Manager, you can perform the simpler restore and recovery scenarios outlined in this chapter. You can also use more sophisticated restore and recovery techniques such as point-in-time recovery and database flashback, which allow for efficient repair of media failures and user errors. In most cases, using Enterprise Manager is simpler than using the RMAN command-line client directly.

  See Also:

  - *Oracle Real Application Clusters Administration and Deployment Guide* for more information about using RMAN with Oracle RAC
  - *Oracle Data Guard Concepts and Administration* for more information about using RMAN with Data Guard
  - *Oracle Database 2 Day DBA* for more details on the restore and recovery features of Enterprise Manager

Preparing for Complete Database Recovery

While RMAN simplifies most database restore and recovery tasks, you must still plan your database restore and recovery strategy based on which database files have been lost and your recovery goal. This section contains the following topics:

- Identifying the Database Files to Restore or Recover
- Determining the DBID of the Database
- Previewing Backups Used in Restore Operations

- Validating Backups Before Restoring Them
- Restoring Archived Redo Logs Needed for Recovery

Identifying the Database Files to Restore or Recover

The techniques for determining which files require restore or recovery depend upon the type of file that is lost.

Identifying a Lost Control File

It is usually obvious when the control file of your database is lost. The database shuts down immediately when any of the multiplexed control files becomes inaccessible. Also, the database reports an error if you try to start it without a valid control file at each location specified in the `CONTROL_FILES` initialization parameter.

Loss of some but not all copies of your control file does not require you to restore a control file from backup. If at least one control file remains intact, then you can either copy an intact copy of the control file over the damaged or missing control file, or update the initialization parameter file so that it does not refer to the damaged or missing control file. After the `CONTROL_FILES` parameter references only present, intact copies of the control file, you can restart your database.

If you restore the control file from backup, then you must perform media recovery of the whole database and then open it with the `OPEN RESETLOGS` option, even if no datafiles need to be restored. This technique is described in "Performing Recovery with a Backup Control File" on page 19-4.

Identifying Datafiles Requiring Media Recovery

When and how to recover depends on the state of the database and the location of its datafiles.

Identifying Datafiles with RMAN An easy technique for determining which datafiles are missing is to run a `VALIDATE DATABASE` command, which attempts to read all specified datafiles. For example, start the RMAN client and run the following commands to validate the database (sample output included).

Example 17–1 BACKUP VALIDATE DATABASE

```
RMAN> VALIDATE DATABASE;

Starting validate at 20-OCT-06
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=90 device type=DISK
could not read file header for datafile 7 error reason 4
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of backup command at 10/20/2007 13:05:43
RMAN-06056: could not access datafile 7
```

The output in Example 17–1 indicates that datafile 7 is inaccessible. You can then run the `REPORT SCHEMA` command to obtain the tablespace name and filename for datafile 7 as follows (sample output included):

```
RMAN> REPORT SCHEMA;

Report of database schema for database with db_unique_name RDBMS
```

```
List of Permanent Datafiles
===========================
File Size(MB) Tablespace           RB segs Datafile Name
---- -------- -------------------- ------- ------------------------
1    450      SYSTEM               ***     +DATAFILE/tbs_01.f
2    86       SYSAUX               ***     +DATAFILE/tbs_ax1.f
3    15       UD1                  ***     +DATAFILE/tbs_undo1.f
4    2        SYSTEM               ***     +DATAFILE/tbs_02.f
5    2        TBS_1                ***     +DATAFILE/tbs_11.f
6    2        TBS_1                ***     +DATAFILE/tbs_12.f
7    2        TBS_2                ***     +DATAFILE/tbs_21.f

List of Temporary Files
=======================
File Size(MB) Tablespace           Maxsize(MB) Tempfile Name
---- -------- -------------------- ----------- --------------------
1    40       TEMP                 32767       +DATAFILE/tbs_tmp1.f
```

Identifying Datafiles with SQL Although `VALIDATE DATABASE` is a good technique for determining whether files are inaccessible, you may want to use SQL queries to obtain more detailed information.

To determine whether datafiles require media recovery:

1. Start SQL*Plus and connect to the target database instance with administrator privileges.
2. Determine the status of the database by executing the following SQL query:

   ```
   SELECT STATUS FROM V$INSTANCE;
   ```

   If the status is `OPEN`, then the database is open. Nevertheless, some datafiles may require media recovery.
3. Query `V$DATAFILE_HEADER` to determine the status of your datafiles. Run the following SQL statements to check the datafile headers:

   ```
   SELECT FILE#, STATUS, ERROR, RECOVER, TABLESPACE_NAME, NAME
   FROM   V$DATAFILE_HEADER
   WHERE  RECOVER = 'YES'
   OR     (RECOVER IS NULL AND ERROR IS NOT NULL);
   ```

   Each row returned represents a datafile that either requires media recovery or has an error requiring a restore. Check the `RECOVER` and `ERROR` columns. `RECOVER` indicates whether a file needs media recovery, and `ERROR` indicates whether there was an error reading and validating the datafile header.

   If `ERROR` is not `NULL`, then the datafile header cannot be read and validated. Check for a temporary hardware or operating system problem causing the error. If there is no such problem, you must restore the file or switch to a copy.

   If the `ERROR` column is `NULL` and the `RECOVER` column is `YES`, then the file requires media recovery (and may also require a restore from backup).

> **Note:** Because `V$DATAFILE_HEADER` only reads the header block of each datafile, it does not detect all problems that require the datafile to be restored. For example, this view cannot tell whether a datafile contains corrupt data blocks.

4. Optionally, query V$RECOVER_FILE to list datafiles requiring recovery by datafile number with their status and error information. For example, execute the following query:

```
SELECT FILE#, ERROR, ONLINE_STATUS, CHANGE#, TIME
FROM   V$RECOVER_FILE;
```

> **Note:** You cannot use V$RECOVER_FILE with a control file restored from backup or a control file that was re-created after the time of the media failure affecting the datafiles. A restored or re-created control file does not contain the information needed to update V$RECOVER_FILE accurately.

To find datafile and tablespace names, you can also perform useful joins using the datafile number and the V$DATAFILE and V$TABLESPACE views. For example:

```
SELECT r.FILE# AS df#, d.NAME AS df_name, t.NAME AS tbsp_name,
       d.STATUS, r.ERROR, r.CHANGE#, r.TIME
FROM V$RECOVER_FILE r, V$DATAFILE d, V$TABLESPACE t
WHERE t.TS# = d.TS#
AND d.FILE# = r.FILE#;
```

The ERROR column identifies the problem for each file requiring recovery.

> **See Also:** *Oracle Database Reference* for information about the V$ views

Determining the DBID of the Database

In situations requiring the recovery of your server parameter file or control file from autobackup, you need to know the DBID. You should record the DBID along with other basic information about your database.

If you do not have a record of the DBID of your database, then you can find it in the following places without opening your database:

- The DBID is used in forming the filename for the control file autobackup. Locate this file, and then refer to "Configuring the Control File Autobackup Format" on page 5-7 to determine where the DBID appears in the filename.
- If you have any text files that preserve the output from an RMAN session, then the DBID is displayed by the RMAN client when it starts up and connects to your database. Typical output follows:

```
% rman TARGET /

Recovery Manager: Release 11.1.0.6.0 - Production on Wed Jul 11 17:51:30 2007

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

connected to target database: PROD (DBID=36508508)
```

Previewing Backups Used in Restore Operations

You can apply RESTORE . . . PREVIEW to any RESTORE operation to create a detailed list of every backup to be used in the requested RESTORE operation, as well as the necessary target SCN for recovery after the RESTORE operation is complete. This

command accesses the RMAN repository to query the backup metadata, but does not actually read the backup files to ensure that they can be restored.

As an alternative to `RESTORE ... PREVIEW`, you can use the `RESTORE ... VALIDATE HEADER` command. In addition to listing the files needed for restore and recovery, the `RESTORE ... VALIDATE HEADER` command validates the backup file headers to determine whether the files on disk or in the media management catalog correspond to the metadata in the RMAN repository.

When planning your restore and recovery operation, use `RESTORE ... PREVIEW` or `RESTORE ... VALIDATE HEADER` to ensure that all required backups are available or to identify situations in which you may want to direct RMAN to use or avoid specific backups.

To preview backups to be used in a restore operation:

1. Run a `RESTORE` command with the `PREVIEW` option.

   For example, run one of the following commands:

   ```
   RESTORE DATABASE PREVIEW;
   RESTORE ARCHIVELOG FROM TIME 'SYSDATE-7' PREVIEW;
   ```

   If the report produced by `RESTORE ... PREVIEW` provides too much information, then specify the `SUMMARY` option as shown in the following example:

   ```
   RESTORE DATABASE PREVIEW SUMMARY;
   ```

   If satisfied with the output, then stop here. If the output indicates that RMAN will request a backup from a tape that you know is temporarily unavailable, then continue with this procedure. If the output indicates that a backup is stored offsite, then skip to "Recalling Offsite Backups" on page 17-6.

2. If needed, use the `CHANGE` command to set the backup status of any temporarily unavailable backups to `UNAVAILABLE`.

   "Updating a Backup to Status AVAILABLE or UNAVAILABLE" on page 11-14 explains how to perform this task.

3. Optionally, run `RESTORE ... PREVIEW` again to confirm that the restore will not attempt to use unavailable backups.

> **See Also:** *Oracle Database Backup and Recovery Reference* for details on interpreting `RESTORE ... PREVIEW` output, which is in the same format as the output of the `LIST` command

Recalling Offsite Backups

Some media managers provide status information to RMAN about which backups are offsite. An **offsite backup** is stored in a remote location, such as a secure storage facility, and cannot be restored unless the media manager retrieves the media.

Offsite backups are marked as `AVAILABLE` in the RMAN repository even though the media must be retrieved from storage before the backup can be restored. If RMAN attempts to restore a offsite backup, then the restore job fails.

You can use `RESTORE ... PREVIEW` to identify offsite backups. The command output indicates whether backups are stored offsite, as shown by the text at the end of the sample output in Example 17–2.

Example 17–2 RESTORE ... PREVIEW Output

```
List of Backup Sets
===================


BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
9       2.25M      SBT_TAPE    00:00:00     21-MAY-07
        BP Key: 9   Status: AVAILABLE  Compressed: NO  Tag: TAG20070521T144258
        Handle: 0aii9k7i_1_1   Media: 0aii9k7i_1_1

  List of Archived Logs in backup set 9
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    1       392314     21-MAY-07 392541     21-MAY-07
  1    2       392541     21-MAY-07 392545     21-MAY-07
  1    3       392545     21-MAY-07 392548     21-MAY-07
  1    4       392548     21-MAY-07 395066     21-MAY-07
  1    5       395066     21-MAY-07 395095     21-MAY-07
  1    6       395095     21-MAY-07 395355     21-MAY-07

List of remote backup files
===========================
        Handle: aii9k7i_1_1   Media: 0aii9k7i_1_1
validation succeeded for backup piece
Finished restore at 21-MAY-07
released channel: dev1
```

You can use `RESTORE . . . PREVIEW RECALL` to instruct the media manager to make offsite backups available.

To recall offsite backups:

1. If backups are stored offsite, then execute a `RESTORE . . . PREVIEW` command with the `RECALL` option.

   The following example initiates recall for the offsite archived log backups shown in Example 17–2 (sample output included):

   ```
   RESTORE ARCHIVELOG ALL PREVIEW RECALL;
   ```

   The following sample output indicates that RMAN initiated a recall:

   ```
   List of Backup Sets
   ===================


   BS Key  Size       Device Type Elapsed Time Completion Time
   ------- ---------- ----------- ------------ ---------------
   9       2.25M      SBT_TAPE    00:00:00     21-MAY-07
           BP Key: 9   Status: AVAILABLE  Compressed: NO  Tag: TAG20070521T144258
           Handle: VAULT0aii9k7i_1_1   Media: /tmp,VAULT0aii9k7i_1_1

     List of Archived Logs in backup set 9
     Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
     ---- ------- ---------- --------- ---------- ---------
     1    1       392314     21-MAY-07 392541     21-MAY-07
     1    2       392541     21-MAY-07 392545     21-MAY-07
     1    3       392545     21-MAY-07 392548     21-MAY-07
     1    4       392548     21-MAY-07 395066     21-MAY-07
     1    5       395066     21-MAY-07 395095     21-MAY-07
   ```

```
  1    6       395095     21-MAY-07 395355     21-MAY-07

Initiated recall for the following list of remote backup files
============================================================
        Handle: VAULT0aii9k7i_1_1   Media: /tmp,VAULT0aii9k7i_1_1
validation succeeded for backup piece
Finished restore at 21-MAY-07
released channel: dev1
```

2. Run the `RESTORE ... PREVIEW` command. If necessary, return to the previous step until no backups needed for the restore are reported as offsite.

Validating Backups Before Restoring Them

While the procedures in "Previewing Backups Used in Restore Operations" on page 17-5 indicate which backups will be restored, they do not verify that the backups are actually usable. You can run RMAN commands to test the availability of usable backups for any `RESTORE` operation, or test the contents of a specific backup for use in `RESTORE` operations. The contents of the backups are actually read and checked for corruption. You have the following validation options:

- `RESTORE ... VALIDATE` tests whether RMAN can restore a specific object from a backup. RMAN chooses which backups to use.
- `VALIDATE BACKUPSET` tests the validity of a backup set that you specify.

See Also: Chapter 15, "Validating Database Files and Backups"

Restoring Archived Redo Logs Needed for Recovery

RMAN restores archived redo log files from backup automatically as needed to perform recovery. You can also restore archived redo logs manually to save the time needed to restore these files later during the `RECOVER` command, or if you want to store the restored archived redo log files in some new location.

By default, RMAN restores archived redo logs with names constructed using the `LOG_ARCHIVE_FORMAT` and the `LOG_ARCHIVE_DEST_1` parameters of the target database. These parameters are combined in a platform-specific fashion to form the name of the restored archived log.

Restoring Archived Redo Logs to a New Location

You can override the default location for restored archived redo logs with the `SET ARCHIVELOG DESTINATION` command. This command manually stages archived logs to different locations while a database restore is occurring. During recovery, RMAN knows where to find the newly restored archived logs; it does not require them to be in the location specified in the initialization parameter file.

To restore archived redo logs to a new location:

1. Start RMAN and connect to a target database.
2. Ensure that the database is mounted or open.
3. Perform the following operations within a `RUN` command:
   a. Specify the new location for the restored archived redo logs using `SET ARCHIVELOG DESTINATION`.
   b. Either explicitly restore the archived redo logs or execute commands that automatically restore the logs.

The following sample RUN command explicitly restores all backup archived logs to a new location:

```
RUN
{
  SET ARCHIVELOG DESTINATION TO '/oracle/temp_restore';
  RESTORE ARCHIVELOG ALL;
  # restore and recover datafiles as needed
  .
  .
  .
}
```

The following example sets the archived log destination and then uses RECOVER DATABASE to restore archived logs from this destination automatically:

```
RUN
{
  SET ARCHIVELOG DESTINATION TO '/oracle/temp_restore';
  RESTORE DATABASE;
  RECOVER DATABASE; # restores and recovers logs automatically
}
```

Restoring Archived Redo Logs to Multiple Locations

You can specify restore destinations for archived logs multiple times in one **RUN block**, in order to distribute restored logs among several destinations. (You cannot, however specify multiple destinations simultaneously to produce multiple copies of the same log during the restore operation.) You can use this feature to manage disk space used to contain the restored logs.

This example restores 300 archived redo logs from backup, distributing them across the directories /fs1/tmp, /fs2/tmp, and /fs3/tmp:

```
RUN
{
  # Set a new location for logs 1 through 100.
  SET ARCHIVELOG DESTINATION TO '/fs1/tmp';
  RESTORE ARCHIVELOG FROM SEQUENCE 1 UNTIL SEQUENCE 100;
  # Set a new location for logs 101 through 200.
  SET ARCHIVELOG DESTINATION TO '/fs2/tmp';
  RESTORE ARCHIVELOG FROM SEQUENCE 101 UNTIL SEQUENCE 200;
  # Set a new location for logs 201 through 300.
  SET ARCHIVELOG DESTINATION TO '/fs3/tmp';
  RESTORE ARCHIVELOG FROM SEQUENCE 201 UNTIL SEQUENCE 300;
  # restore and recover datafiles as needed
  .
  .
  .
}
```

When you issue a RECOVER command, RMAN finds the needed restored archived logs automatically across the destinations to which they were restored, and applies them to the datafiles.

Performing Complete Database Recovery

This section assumes that you have already performed the tasks in "Preparing for Complete Database Recovery" on page 17-2. This section describes the basic outline of

complete database recovery, which is intended to encompass a wide range of different scenarios.

About Complete Database Recovery

You use the RESTORE and RECOVER commands to restore and recover the database. During the recovery, RMAN automatically restores backups of any needed archived redo logs. If backups are stored on a media manager, then channels must be configured in advance or a RUN block with ALLOCATE CHANNEL commands must be used to enable access to backups stored there.

If RMAN restores archived redo logs to the flash recovery area during a recovery, then it automatically deletes the restored logs after applying them to the datafiles. Otherwise, you can use the DELETE ARCHIVELOG command to delete restored archived redo logs from disk when they are no longer needed for recovery. For example, you can enter the following command:

```
RECOVER DATABASE DELETE ARCHIVELOG;
```

Restoring Datafiles to a Nondefault Location

If you cannot restore datafiles to their default locations, then you must update the control file to reflect the new locations of the datafiles. Use the RMAN SET NEWNAME command within a RUN command to specify the new filename. Afterward, use a SWITCH command, which is equivalent to using the SQL statement ALTER DATABASE RENAME FILE, to update the names of the datafiles in the control file. SWITCH DATAFILE ALL updates the control file to reflect the new names for all datafiles for which a SET NEWNAME has been issued in a RUN command.

> **See Also:** *Oracle Database Backup and Recovery Reference* for SWITCH syntax

Decryption of Backups

If RMAN is restoring encrypted backups, then RMAN automatically decrypts backup sets when their contents are restored. Transparently encrypted backups require no intervention to restore, as long as the Oracle wallet is open and available.

Password-encrypted backups require the correct password to be entered before they can be restored. You must enter the encryption password with the SET DECRYPTION command. If restoring from a group of backups that were created with different passwords, then specify all of the required passwords on the SET DECRYPTION command. RMAN will automatically use the correct password with each backup set.

> **See Also:**
>
> - "Configuring Backup Encryption" on page 6-7 to learn how to configure transparent backup encryption
> - "Encrypting RMAN Backups" on page 9-10 to learn how to create encrypted backups

Performing Complete Recovery of the Whole Database

This scenario assumes that database trgt has lost most or all of its datafiles. It also assumes that the database uses a flash recovery area.

Note that after restore and recovery of a whole database, when the database is opened, any missing temporary tablespaces recorded in the control file are re-created with their previous creation size, AUTOEXTEND, and MAXSIZE attributes. Only temporary

tablespaces that are missing are re-created. If a tempfile exists at the location recorded in the RMAN repository but has an invalid header, then RMAN does not re-create the tempfile.

If the tempfiles were originally created as Oracle-managed files, then they are re-created in the current DB_CREATE_FILE_DEST location. Otherwise, they are re-created at their previous locations. If RMAN is unable to re-create the file due to an I/O error or some other cause, then the error is reported in the alert log and the database open operation continues.

To restore and recover the whole database:

1. Start RMAN and connect to a target database.

   For example, enter the following command:

   ```
   % rman
   RMAN> CONNECT TARGET /
   ```

   RMAN displays the database status when it connects: not started, not mounted, not open (when the database is mounted but not open), or none (when the database is open).

2. If the database is not mounted, then mount but do not open the database.

   For example, enter the following command:

   ```
   STARTUP MOUNT;
   ```

3. Use the SHOW command to see which channels are preconfigured.

   For example, enter the following command (sample output is included):

   ```
   SHOW ALL;


   RMAN configuration parameters for database with db_unique_name PROD1 are:
   .
   .
   .
   CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
   CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
   CONFIGURE DEVICE TYPE SBT_TAPE PARALLELISM 1 BACKUP TYPE TO BACKUPSET; #
   default
   CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS "SBT_
   LIBRARY=/usr/local/oracle/backup/lib/libobk.so";
   ```

   If the necessary devices and channels are already configured, then no action is necessary. Otherwise, you can use the CONFIGURE command to configure automatic channels, or include ALLOCATE CHANNEL commands within a RUN block.

4. If restoring password-protected encrypted backups, then specify the password.

   Use the SET DECRYPTION IDENTIFIED BY command to specify a password for password-protected backups, as shown in the following example (where *password* represents the actual password that you enter):

   ```
   SET DECRYPTION IDENTIFIED BY password;
   ```

   If you created backups with different passwords, then you can run the SET DECRYPTION IDENTIFIED BY *password* command multiple times, specifying all of the possible passwords that might be required to restore the backups.

5. Restore and recover the database. Do one of the following:

   - If you are restoring all datafiles to their original locations, then execute `RESTORE DATABASE` and `RECOVER DATABASE` sequentially at the RMAN prompt.

     For example, enter the following commands if automatic channels are configured (sample output included):

```
RMAN> RESTORE DATABASE;

Starting restore at 20-JUN-06
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=35 device type=DISK
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=34 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00001 to /disk1/oracle/dbs/tbs_01.f
channel ORA_DISK_1: restoring datafile 00002 to /disk1/oracle/dbs/tbs_ax1.f
.
.
.
Finished restore at 20-JUN-06

RMAN> RECOVER DATABASE;

Starting recover at 20-JUN-06
using channel ORA_DISK_1
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=34 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup

starting media recovery

channel ORA_DISK_1: starting archived log restore to default destination
channel ORA_DISK_1: restoring archived log
archived log thread=1 sequence=5
channel ORA_DISK_1: restoring archived log
archived log thread=1 sequence=6
.
.
.
channel ORA_DISK_1: reading from backup piece
/disk1/oracle/work/orcva/TKRM/backupset/2007_06_20/o1_mf_annnn_
TAG20070620T113128_29jhr197_.bkp
channel ORA_DISK_1: piece
handle=/disk1/oracle/work/orcva/TKRM/backupset/2007_06_20/o1_mf_annnn_
TAG20070620T113128_29jhr197_.bkp tag=TAG20070620T113128
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:02
archived log file name=/disk1/oracle/work/orcva/TKRM/archivelog/2007_06_
20/o1_mf_1_5_29jhv47k_.arc thread=1 sequence=5
channel default: deleting archived log(s)
.
.
.
media recovery complete, elapsed time: 00:00:15
Finished recover at 20-JUN-06
```

If you manually allocate channels, then you must issue the RESTORE and RECOVER commands together within a RUN block as shown in the following example:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

- If you are restoring some datafiles to new locations, then execute RESTORE DATABASE and RECOVER DATABASE sequentially in a RUN command. Use the SET NEWNAME to rename datafiles, as described in "Restoring Datafiles to a Nondefault Location" on page 17-10.

  The following example restores the database, specifying new names for three of the datafiles, and then recovers the database:

```
RUN
{
  SET NEWNAME FOR DATAFILE 2 TO '/disk2/df2.dbf';
  SET NEWNAME FOR DATAFILE 3 TO '/disk2/df3.dbf';
  SET NEWNAME FOR DATAFILE 4 TO '/disk2/df4.dbf';
  RESTORE DATABASE;
  SWITCH DATAFILE ALL;
  RECOVER DATABASE;
}
```

6. Examine the output to see if media recovery was successful. If so, open the database.

   For example, enter the following command:

```
ALTER DATABASE OPEN;
```

Performing Complete Recovery of a Tablespace

In the basic scenario, the database is open, and some but not all of the datafiles are damaged. You want to restore and recover the damaged tablespace while leaving the database open so that the rest of the database remains available. This scenario assumes that database trgt has lost tablespace users.

To restore and recover a tablespace:

1. Start RMAN and connect to a target database.
2. If the database is open, then take the datafile requiring recovery offline.

   For example, enter the following command to take users offline:

```
SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
```

3. Use the SHOW command to see which channels are preconfigured.

   For example, enter the following command (sample output is included):

```
SHOW ALL;

RMAN configuration parameters for database with db_unique_name PROD1 are:
.
.
```

```
.
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DEVICE TYPE SBT_TAPE PARALLELISM 1 BACKUP TYPE TO BACKUPSET; #
default
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS  "SBT_
LIBRARY=/usr/local/oracle/backup/lib/libobk.so";
```

If the necessary devices and channels are already configured, then no action is necessary. Otherwise, you can use the `CONFIGURE` command to configure automatic channels, or include `ALLOCATE CHANNEL` commands within a `RUN` block.

4. If restoring password-protected encrypted backups, then specify the password.

   Use the `SET DECRYPTION IDENTIFIED BY` command to specify a password for password-protected backups, as shown in the following example (where *password* represents the actual password that you enter):

   ```
   SET DECRYPTION IDENTIFIED BY password;
   ```

5. Restore and recover the tablespace. Do one of the following:

   - If you are restoring datafiles to their original locations, then run the `RESTORE TABLESPACE` and `RECOVER TABLESPACE` commands at the RMAN prompt.

     For example, enter the following command if automatic channels are configured (sample output included):

     ```
     RMAN> RESTORE TABLESPACE users;

     Starting restore at 20-JUN-06
     allocated channel: ORA_DISK_1
     channel ORA_DISK_1: SID=37 device type=DISK
     allocated channel: ORA_SBT_TAPE_1
     channel ORA_SBT_TAPE_1: SID=38 device type=SBT_TAPE
     channel ORA_SBT_TAPE_1: Oracle Secure Backup

     channel ORA_DISK_1: starting datafile backup set restore
     channel ORA_DISK_1: specifying datafile(s) to restore from backup set
     channel ORA_DISK_1: restoring datafile 00012 to /disk1/oracle/dbs/users01.f
     channel ORA_DISK_1: restoring datafile 00013 to /disk1/oracle/dbs/users02.f
     channel ORA_DISK_1: restoring datafile 00021 to /disk1/oracle/dbs/users03.f
     channel ORA_DISK_1: reading from backup piece
     /disk1/oracle/work/orcva/TKRM/backupset/2007_06_20/o1_mf_nnndf_
     TAG20070620T105435_29jflwor_.bkp
     channel ORA_DISK_1: piece
     handle=/disk1/oracle/work/orcva/TKRM/backupset/2007_06_20/o1_mf_nnndf_
     TAG20070620T105435_29jflwor_.bkp tag=TAG20070620T105435
     channel ORA_DISK_1: restored backup piece 1
     channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
     Finished restore at 20-JUN-06

     RMAN> RECOVER TABLESPACE users;

     Starting recover at 20-JUN-06
     using channel ORA_DISK_1
     using channel ORA_SBT_TAPE_1

     starting media recovery

     archived log for thread 1 with sequence 27 is already on disk as file
     ```

```
/disk1/oracle/work/orcva/TKRM/archivelog/2007_06_20/o1_mf_1_27_29jjmtc9_
.arc
archived log for thread 1 with sequence 28 is already on disk as file
/disk1/oracle/work/orcva/TKRM/archivelog/2007_06_20/o1_mf_1_28_29jjnc5x_
.arc
.
.
.
channel ORA_DISK_1: starting archived log restore to default destination
channel ORA_DISK_1: restoring archived log
archived log thread=1 sequence=5
channel ORA_DISK_1: restoring archived log
archived log thread=1 sequence=6
channel ORA_DISK_1: restoring archived log
archived log thread=1 sequence=7
.
.
.
channel ORA_DISK_1: reading from backup piece
/disk1/oracle/work/orcva/TKRM/backupset/2007_06_20/o1_mf_annnn_
TAG20070620T113128_29jhr197_.bkp
channel ORA_DISK_1: piece
handle=/disk1/oracle/work/orcva/TKRM/backupset/2007_06_20/o1_mf_annnn_
TAG20070620T113128_29jhr197_.bkp tag=TAG20070620T113128
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:02
archived log file name=/disk1/oracle/work/orcva/TKRM/archivelog/2007_06_
20/o1_mf_1_5_29jkdvjq_.arc thread=1 sequence=5
channel default: deleting archived log(s)
archived log file name=/disk1/oracle/work/orcva/TKRM/archivelog/2007_06_
20/o1_mf_1_5_29jkdvjq_.arc RECID=91 STAMP=593611179
archived log file name=/disk1/oracle/work/orcva/TKRM/archivelog/2007_06_
20/o1_mf_1_6_29jkdvbz_.arc thread=1 sequence=6
channel default: deleting archived log(s)
.
.
.
media recovery complete, elapsed time: 00:00:01
Finished recover at 20-JUN-06
```

- If you are restoring some datafiles to new locations, then execute `RESTORE TABLESPACE` and `RECOVER TABLESPACE` in a `RUN` command. Use the `SET NEWNAME` to rename datafiles, as described in "Restoring Datafiles to a Nondefault Location" on page 17-10.

  The following example restores the datafiles in tablespaces `users` to a new location, then performs recovery. Assume that the old datafiles were stored in the `/disk1` path and the new ones will be stored in the `/disk2` path.

```
RUN
{
  # specify the new location for each datafile
  SET NEWNAME FOR DATAFILE '/disk1/oracle/dbs/users01.f' TO
                           '/disk2/users01.f';
  SET NEWNAME FOR DATAFILE '/disk1/oracle/dbs/users02.f' TO
                           '/disk2/users02.f';
  SET NEWNAME FOR DATAFILE '/disk1/oracle/dbs/users03.f' TO
                           '/disk2/users03.f';
  RESTORE TABLESPACE users;
  SWITCH DATAFILE ALL;   # update control file with new filenames
  RECOVER TABLESPACE users;
```

```
}
```

6. Examine the output to see if recovery was successful. If so, bring the recovered tablespace back online.

   For example, enter the following command:

   ```
   SQL "ALTER TABLESPACE users ONLINE";
   ```

Performing Complete Recovery After Switching to a Copy

If you have image copies of the inaccessible datafiles in the flash recovery area, then you can use the `SWITCH DATAFILE ... TO COPY` command to point the control file at the datafile copy and then use `RECOVER` to recover lost changes. You can also use the SWITCH DATABASE TO COPY command to point the control file at a copy of the whole database. Because you do not need to restore backups, this recovery technique takes less time than traditional restore and recovery.

> **Note:** A `SWITCH TABLESPACE ... TO COPY` command is also supported for cases when all datafiles in a tablespace are lost and copies of all datafiles exist. The same restriction exists for `SWITCH DATABASE TO COPY`.

Switching to a Datafile Copy

In the basic scenario, the database is open, and some but not all of the datafiles are damaged. During the course of the day, a datafile goes missing due to storage failure. You need to repair this file, but cannot afford the time to do a restore and recovery from a backup. You decide to use a recent image copy backup as the new file, thus eliminating restore time. This scenario assumes that database `trgt` has lost datafile 4.

To switch to a datafile copy and perform recovery:

1. Start RMAN and connect to a target database.
2. If the database is open, then take the tablespace requiring recovery offline.

   Enter the following command to take datafile 4 offline:

   ```
   SQL "ALTER DATABASE DATAFILE 4 OFFLINE";
   ```

3. Switch the offline datafile to the latest copy.

   Enter the following command to point the control file to the latest image copy of datafile 4:

   ```
   SWITCH DATAFILE 4 TO COPY;
   ```

4. Recover the datafile with the `RECOVER DATAFILE` command.

   Enter the following command:

   ```
   RECOVER DATAFILE 4;
   ```

   RMAN automatically restores archived redo logs and incremental backups. Because the database uses a flash recovery area, RMAN automatically deletes them after they have been applied.

5. Examine the output to see if recovery was successful. If so, bring the recovered datafile back online.

   Enter the following command to bring datafile 4 online:

```
SQL "ALTER DATABASE DATAFILE 4 ONLINE";
```

Switching to a Database Copy

In this scenario, the database is shut down, and all of the datafiles are damaged. You have image copies of all the damaged datafiles. You decide to use the existing image copies as the new datafiles, thus eliminating restore time.

To switch to a database copy and perform recovery:

1. Start RMAN and connect to a target database.
2. Mount the database.
3. Switch the database to the latest copy.

   Enter the following command to point the control file to the latest image copy of the database:

   ```
   SWITCH DATABASE TO COPY;
   ```

4. Recover the database with the `RECOVER DATABASE` command.

   Enter the following command:

   ```
   RECOVER DATABASE;
   ```

   RMAN automatically restores archived redo logs and incremental backups. Because the database uses a flash recovery area, RMAN automatically deletes them after they have been applied.

5. Examine the output to see if recovery was successful. If so, open the database.

   Enter the following command to open the database:

   ```
   ALTER DATABASE OPEN;
   ```

18

Performing Block Media Recovery

This chapter explains how to restore and recover individual data blocks within a datafile. This chapter contains the following topics:

- Overview of Block Media Recovery
- Prerequisites for Block Media Recovery
- Recovering Individual Blocks
- Recovering All Blocks in V$DATABASE_BLOCK_CORRUPTION

See Also:

- *Oracle Database Backup and Recovery Reference* for `RECOVER` syntax
- *Oracle Database Reference* for details about the `V$DATABASE_BLOCK_CORRUPTION` view

Overview of Block Media Recovery

This section explains the purpose and basic concepts of block media recovery.

Purpose of Block Media Recovery

You can use **block media recovery** to recover one or more corrupt data blocks within a datafile. Block media recovery provides the following advantages over **datafile media recovery**:

- Lowers the **Mean Time To Recover (MTTR)** because only blocks needing recovery are restored and recovered
- Enables affected datafiles to remain online during recovery

  Without block media recovery, if even a single block is corrupt, then you must take the datafile offline and restore a backup of the datafile. You must apply all redo generated for the datafile after the backup was created. The entire file is unavailable until media recovery completes. With block media recovery, only the blocks actually being recovered are unavailable during the recovery.

Block media recovery is most useful for **physical corruption** problems that involve a small, well-known number of blocks. Block-level data loss usually results from intermittent, random I/O errors that do not cause widespread data loss, as well as memory corruptions that get written to disk. Block media recovery is not intended for cases where the extent of data loss or corruption is unknown and the entire datafile requires recovery. In such cases, datafile media recovery is the best solution.

Basic Concepts of Block Media Recovery

In most cases, the database marks a block as media corrupt and then writes it to disk when the corruption is first encountered. No subsequent read of the block will be successful until the block is recovered. You can only perform block recovery on blocks that are marked corrupt or fail a corruption check.

You perform block media recovery with the `RECOVER ... BLOCK` command. By default, RMAN searches the flashback logs for good copies of the blocks, and then searches for the blocks in full or level 0 incremental backups. When RMAN finds good copies, it restores them and performs media recovery on the blocks. Block media recovery can only use redo logs for media recovery, not level 1 incremental backups.

> **See Also:** *Oracle Database Backup and Recovery Reference* for `RECOVER ... BLOCK` syntax

Identification of Corrupt Blocks

The `V$DATABASE_BLOCK_CORRUPTION` view displays blocks marked corrupt by database components such as RMAN, `ANALYZE`, `dbv`, and SQL queries. The following types of corruption result in rows added to this view:

- Physical corruption (sometimes called media corruption)

  The database does not recognize the block: the **checksum** is invalid, the block contains all zeros, or the block header is fractured.

  Physical corruption checking is enabled by default. You can turn off checksum checking by specifying the `NOCHECKSUM` option of the `BACKUP` command, but other physical consistency checks, such as checks of the block headers and footers, cannot be disabled.

- Logical corruption

  The block has a valid checksum, the header and footer match, and so on, but the contents are logically inconsistent. Block media recovery cannot repair logical block corruption.

  Logical corruption checking is disabled by default. You can turn it on by specifying the `CHECK LOGICAL` option of the `BACKUP`, `RESTORE`, `RECOVER`, and `VALIDATE` commands.

The database can detect some corruptions by validating relationships between blocks and segments, but cannot detect them by a check of an individual block. The `V$DATABASE_BLOCK_CORRUPTION` view does not record this type of corruption.

Missing Redo During Block Recovery

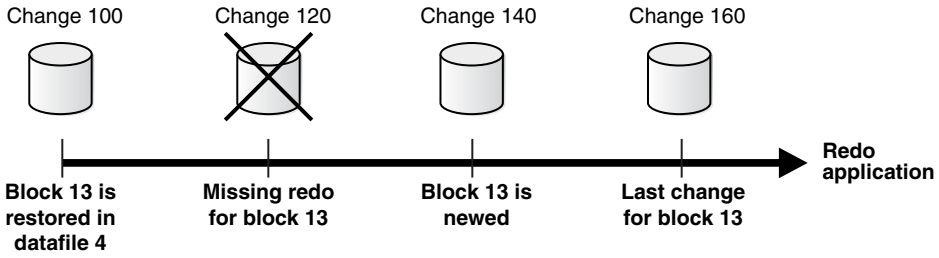
Like datafile media recovery, block media recovery cannot generally survive a missing or inaccessible archived log, although it will attempt restore failover when looking for usable copies of archived redo log files, as described in "Restore Failover" on page 13-4. Also, block media recovery cannot survive physical redo corruptions that result in **checksum** failure. However, block media recovery can survive gaps in the redo stream if the missing or corrupt redo records do not affect the blocks being recovered. Whereas datafile recovery requires an unbroken series of redo changes from the beginning of recovery to the end, block media recovery only requires an unbroken set of redo changes for the blocks being recovered.

> **Note:** Each block is recovered independently during block media recovery, so recovery may be successful for a subset of blocks.

When RMAN first detects missing or corrupt redo records during block media recovery, it does not immediately signal an error because the block undergoing recovery may become a **newed block** later in the redo stream. When a block is newed all previous redo for that block becomes irrelevant because the redo applies to an old incarnation of the block. For example, the database can new a block when users drop or truncate a table and then use the block for other data.

Assume that media recovery is performed on block 13 as depicted in the following figure.

Figure 18–1 Performing RMAN Media Recovery



After block recovery begins, RMAN discovers that change 120 is missing from the redo stream, either because the log block is corrupt or because the log cannot be found. RMAN continues recovery in the hope that block 13 will be newed later in the redo stream. Assume that in change 140 a user drops the table `employees` stored in block 13, allocates a new table in this block, and inserts data into the new table. At this point, the database formats block 13 as a new block. Recovery can now proceed with this block even though some redo preceding the newing operation was missing.

Prerequisites for Block Media Recovery

The following prerequisites apply to the `RECOVER ... BLOCK` command:

- The target database must run in `ARCHIVELOG` mode and be open or mounted with a current control file.
- If the target database is a standby database, then it must be in a consistent state, recovery cannot be in session, and the backup must be older than the corrupted file.
- The backups of the datafiles containing the corrupt blocks must be full or level 0 backups and not proxy copies.

  If only **proxy copy** backups exist, then you can restore them to a nondefault location on disk, in which case RMAN considers them datafile copies and searches them for blocks during block media recovery.
- RMAN can use only archived redo logs for the recovery.

  RMAN cannot use level 1 incremental backups. Block media recovery cannot survive a missing or inaccessible archived redo log, although it can sometimes survive missing redo records.
- Flashback Database must be enabled on the target database for RMAN to search the flashback logs for good copies of corrupt blocks.

  If flashback logging is enabled and contains older, uncorrupted versions of the corrupt blocks, then RMAN can use these blocks, possibly speeding up the recovery.

Recovering Individual Blocks

Typically, block corruption is reported in the following locations:

- Results of the LIST FAILURE, VALIDATE, or BACKUP ... VALIDATE command
- The V$DATABASE_BLOCK_CORRUPTION view
- Error messages in standard output
- The alert log
- User trace files
- Results of the SQL commands ANALYZE TABLE and ANALYZE INDEX
- Results of the DBVERIFY utility
- Third-party media management output

For example, you may discover the following messages in a user trace file:

```
ORA-01578: ORACLE data block corrupted (file # 7, block # 3)
ORA-01110: data file 7: '/oracle/oradata/trgt/tools01.dbf'
ORA-01578: ORACLE data block corrupted (file # 2, block # 235)
ORA-01110: data file 2: '/oracle/oradata/trgt/undotbs01.dbf'
```

In the following procedure, you identify the blocks that require recovery and then use any available backup to perform the restore and recovery of these blocks.

To recover specific data blocks:

1. Obtain the datafile numbers and block numbers of the corrupted blocks.

   The easiest way to locate trace files and the alert log is to connect SQL*Plus to the target database and execute the following query:

   ```
   SELECT NAME, VALUE
   FROM   V$DIAG_INFO;
   ```

2. Start RMAN and connect to the target database, which must be mounted or open.
3. Run the SHOW ALL command to make sure that the appropriate channels are preconfigured.
4. Run the RECOVER ... BLOCK command at the RMAN prompt, specifying the file and block numbers for the corrupted blocks.

   The following example recovers two blocks.

   ```
   RECOVER
     DATAFILE 8 BLOCK 13
     DATAFILE 2 BLOCK 19;
   ```

   You can also specify various options to control RMAN behavior. The following example indicates that only backups with tag mondayam will be used when searching for blocks. You could use the FROM BACKUPSET option to restrict the type of backup that RMAN searches, or EXCLUDE FLASHBACK LOG to restrict RMAN from searching the flashback logs.

   ```
   RECOVER
     DATAFILE 8 BLOCK 13
     DATAFILE 2 BLOCK 199
     FROM TAG mondayam;
   ```

Recovering All Blocks in V$DATABASE_BLOCK_CORRUPTION

In this scenario, RMAN automatically recovers all blocks listed in the V$DATABASE_BLOCK_CORRUPTION view.

To recover all blocks logged in V$DATABASE_BLOCK_CORRUPTION:

1. Start SQL*Plus and connect to the target database.
2. Query V$DATABASE_BLOCK_CORRUPTION to determine whether corrupt blocks exist. For example, execute the following statement:

   ```
   SQL> SELECT * FROM V$DATABASE_BLOCK_CORRUPTION;
   ```

3. Start RMAN and connect to the target database.
4. Recover all blocks marked corrupt in V$DATABASE_BLOCK_CORRUPTION.

   The following command repairs all physically corrupted blocks recorded in the view:

   ```
   RMAN> RECOVER CORRUPTION LIST;
   ```

   After the blocks are recovered, the database removes them from V$DATABASE_BLOCK_CORRUPTION.

   See Also: *Oracle Database Backup and Recovery Reference* to learn about the RECOVER ... BLOCK command

19

Performing RMAN Recovery: Advanced Scenarios

The preceding chapters in Part V, "Diagnosing and Responding to Failures" cover the most basic recovery scenarios and are intended to be as generic as possible. The scenarios in this chapter are advanced in the sense that they are not as common or are more complicated than the basic scenarios.

This chapter contains the following topics:

- Recovering a NOARCHIVELOG Database with Incremental Backups
- Restoring the Server Parameter File
- Performing Recovery with a Backup Control File
- Performing Disaster Recovery
- Restoring a Database on a New Host

Recovering a NOARCHIVELOG Database with Incremental Backups

Restore of a database running in `NOARCHIVELOG` mode is similar to restore of a database in `ARCHIVELOG` mode. The main differences are:

- Only consistent backups can be used in restoring a database in `NOARCHIVELOG` mode.
- Media recovery is not possible because no archived redo logs exist.

You can perform limited recovery of changes to a database running in `NOARCHIVELOG` mode by applying incremental backups. The incremental backups must be consistent, like all backups of a database run in `NOARCHIVELOG` mode, so you cannot make backups of the database when it is open.

When recovering a `NOARCHIVELOG` database, specify the `NOREDO` option on the `RECOVER` command to indicate that RMAN should not attempt to apply archived redo logs. Otherwise, RMAN returns an error.

To recover a `NOARCHIVELOG` database with incremental backups:

1. After connecting to `trgt` and the catalog database, place the database in a mounted state:

   ```
   STARTUP FORCE MOUNT
   ```

2. Restore and recover the database.

   For example, you can perform incomplete recovery with the following commands:

```
RESTORE DATABASE
  FROM TAG "consistent_whole_backup";
RECOVER DATABASE NOREDO;
```

3. Open the database with the RESETLOGS option.

   For example, enter the following command:

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

Restoring the Server Parameter File

If you lose the server parameter file, then RMAN can restore it to its default location or to a location of your choice. Unlike the loss of the control file, the loss of the server parameter file does not cause the instance to immediately stop. The instance may continue operating, although you will have to shut it down and restart it after restoring the server parameter file.

Note the following considerations when restoring the server parameter file:

- If the instance is already started with the server parameter file, then you cannot overwrite the existing server parameter file.
- When the instance is started with a client-side initialization parameter file, RMAN restores the server parameter file to the default location if the TO clause is not used. The default location is platform-specific, for example, ?/dbs/spfile.ora on Linux.
- A recovery catalog simplifies the recovery procedure because you can avoid the step of having to record and remember the DBID. This procedure assumes that you are *not* using a recovery catalog.

To restore the server parameter file from autobackup:

1. Start RMAN and do one of the following:
   - If the database instance is started at the time of the loss of the server parameter file, then connect to the target database.
   - If the database instance is not started when the server parameter file is lost, and if you are not using a recovery catalog, then run SET DBID to set the DBID of the target database. See "Determining the DBID of the Database" on page 17-5 for details on determining your DBID.
2. Shut down the database instance and restart it without mounting.

   When the server parameter file is not available, RMAN starts the instance with a dummy parameter file. For example, enter the following command:

   ```
   STARTUP FORCE NOMOUNT;
   ```

3. Execute a RUN command to restore the server parameter file.

   Depending on the situation, you may need to execute multiple commands in the RUN command. Note the following considerations:

   - If restoring from tape, then use ALLOCATE CHANNEL to allocate an SBT channel manually. If restoring from disk, then RMAN uses the default disk channel.
   - If the autobackups were not produced with the default format (%F), then use the SET CONTROLFILE AUTOBACKUP FOR DEVICE TYPE command to specify the format in effect when the autobackup was performed.

- If the most recent autobackup was not created today, then use `SET UNTIL` to specify the date from which to start the search.
- If RMAN is not connected to a recovery catalog, then you need to use `SET DBID` to set the DBID for the target database.
- If you want to restore the server parameter file to a nondefault location, then specify the `TO` clause or `TO PFILE` clause on the `RESTORE SPFILE` command.
- If you know that RMAN never produces more than *n* autobackups each day, then you can set the `RESTORE SPFILE FROM AUTOBACKUP ... MAXSEQ` parameter to *n* to reduce the search time. `MAXSEQ` is set to 255 by default, and `RESTORE` counts backward from `MAXSEQ` to find the last backup of the day. To terminate the restore operation if you do not find the autobackup in the current day (or specified day), then you can set `MAXDAYS 1` on the `RESTORE` command.

The following example illustrates a `RUN` command that restores a server parameter file from an autobackup on tape:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt PARMS ...;
  SET UNTIL TIME 'SYSDATE-7';
  SET CONTROLFILE AUTOBACKUP FORMAT
    FOR DEVICE TYPE sbt TO '/disk1/control_files/autobackup_%F';
  SET DBID 123456789;
  RESTORE SPFILE
    TO '/tmp/spfileTEMP.ora'
    FROM AUTOBACKUP MAXDAYS 10;
}
```

4. Restart the database instance with the restored file.

   If restarting with a server parameter file in a nondefault location, then create a new initialization parameter file with the line `SPFILE=`*`new_location`*, where *`new_location`* is the path name of the restored server parameter file. Then, restart the instance with the client-side initialization parameter file.

   For example, create a file `/tmp/init.ora` which contains the single line:

   ```
   SPFILE=/tmp/spfileTEMP.ora
   ```

   You can use the following RMAN command to restart the instance with the restored server parameter file:

   ```
   STARTUP FORCE PFILE=/tmp/init.ora;
   ```

Restoring the Server Parameter File from a Control File Autobackup

If you have configured control file autobackups, then the server parameter file is backed up with the control file whenever an autobackup is taken.

To restore the server parameter file from the control file autobackup, you must first set the DBID for your database and then use the `RESTORE SPFILE FROM AUTOBACKUP` command. If the autobackup is in a nondefault format, then first use the `SET CONTROLFILE AUTOBACKUP FORMAT` command to specify the format.

Example 19–1 sets the DBID and restores the server parameter file from a control file autobackup in a nondefault location.

Example 19–1 Restoring the Server Parameter File from a Control File Autobackup

```
SET DBID 320066378;
RUN
{
  SET CONTROLFILE AUTOBACKUP FORMAT
    FOR DEVICE TYPE DISK TO 'autobackup_format';
  RESTORE SPFILE FROM AUTOBACKUP;
}
```

RMAN uses the autobackup format and DBID to hunt for control file autobackups. If a control file autobackup is found, then RMAN restores the server parameter file from that backup to its default location.

To learn how to determine the correct value for *autobackup_format*, see the description of CONFIGURE CONTROLFILE AUTOBACKUP FORMAT in the entry for CONFIGURE in *Oracle Database Backup and Recovery Reference*.

> **See Also:** "Determining the DBID of the Database" on page 17-5 for details on how to determine your DBID

Creating an Initialization Parameter File with RMAN

You can also restore the server parameter file as a client-side initialization parameter file with the TO PFILE *'filename'* clause. The filename you specify should be on a file system accessible from the host where the RMAN client is running. This file need not be accessible directly from the host running the instance.

The following RMAN command creates an initialization parameter file named /tmp/initTEMP.ora on the system running the RMAN client:

```
RESTORE SPFILE TO PFILE '/tmp/initTEMP.ora';
```

To restart the instance with the initialization parameter file, use the following command, again running RMAN on the same client host:

```
STARTUP FORCE PFILE='/tmp/initTEMP.ora';
```

Performing Recovery with a Backup Control File

This section explains what to do when all current control files are lost and you must restore a control file backup.

About Recovery with a Backup Control File

If all copies of the current control file are lost or damaged, then you must restore and mount a backup control file. You must then run the RECOVER command, even if no datafiles have been restored, and open the database with the RESETLOGS option. If some copies of the current control file are usable, however, then you can follow the procedure in "Responding to the Loss of a Subset of the Current Control Files" on page 29-1 and avoid the recovery and RESETLOGS.

During recovery, RMAN automatically searches for online and archived logs that are not recorded in the RMAN repository and catalogs any that it finds. RMAN attempts to find a valid archived redo log in any current archiving destination with the current log format. The current format is specified in the initialization parameter file used to start the instance (or all instances in an Oracle RAC configuration). Similarly, RMAN attempts to find the online redo logs by using the filenames listed in the control file.

If you changed the archiving destination or format during recovery, or if you added new online log members after the backup of the control file, then RMAN may not be able to automatically catalog a needed online or archived log. Whenever RMAN cannot find online redo logs and you did not specify an UNTIL time, RMAN reports errors similar to the following:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of recover command at 08/29/2007 14:23:09
RMAN-06054: media recovery requesting unknown log: thread 1 scn 86945
```

In this case, you must use the CATALOG command to manually add the required redo logs to the repository so that recovery can proceed.

> **See Also:** The discussion of RESTORE CONTROLFILE in *Oracle Database Backup and Recovery Reference* For more details on restrictions on using RESTORE CONTROLFILE in different scenarios (such as when using a recovery catalog, or restoring from a specific backup)

Control File Locations

When restoring the control file, the default destination is all of the locations defined in the CONTROL_FILES initialization parameter. If you do not set the CONTROL_FILES initialization parameter, then the database uses the same rules to determine the destination for the restored control file that it uses when creating a control file if the CONTROL_FILES parameter is not set. These rules are described in *Oracle Database SQL Language Reference* in the description of the CREATE CONTROLFILE statement.

One way to restore the control file to one or more new locations is to change the CONTROL_FILES initialization parameter, and then use the RESTORE CONTROLFILE command with no arguments to restore the control file to the default locations. For example, if you are restoring your control file after a disk failure made some but not all CONTROL_FILES locations unusable, you can change CONTROL_FILES to replace references to the failed disk with path names pointing to another disk, and then run RESTORE CONTROLFILE with no arguments.

You can also restore the control file to any location you choose other than the CONTROL_FILES locations, by using the form RESTORE CONTROLFILE TO '*filename*' [FROM AUTOBACKUP]:

```
RESTORE CONTROLFILE TO '/tmp/my_controlfile';
```

You can perform this operation with the database in NOMOUNT, MOUNT or OPEN states, because you are not overwriting any of the control files currently in use. Any existing file named '*filename*' is overwritten. After restoring the control file to a new location, you can then update the CONTROL_FILES initialization parameter to include the new location.

> **See Also:** *Oracle Database Backup and Recovery Reference* for RESTORE CONTROLFILE syntax.

Recovery With and Without a Recovery Catalog

When RMAN is connected to a recovery catalog, the recovery procedure with a backup control file is identical to recovery with a current control file. The RMAN metadata missing from the backup control file is available from the recovery catalog. The only exception is if the database name is not unique in the catalog, in which case you must use SET DBID command before restoring the control file.

If you are not using a recovery catalog, then you must restore your control file from an autobackup. To restore the control file from autobackup, the database must be in a NOMOUNT state. As shown in Example 19–2, you must first set the DBID for your database, and then use the RESTORE CONTROLFILE FROM AUTOBACKUP command.

Example 19–2 Setting the DBID and Restoring the Control File from Autobackup

```
SET DBID 320066378;
RUN
{
  SET CONTROLFILE AUTOBACKUP FORMAT
    FOR DEVICE TYPE DISK TO 'autobackup_format';
  RESTORE CONTROLFILE FROM AUTOBACKUP;
}
```

RMAN uses the autobackup format and DBID to determine where to hunt for the control file autobackup. If one is found, RMAN restores the control file to all control file locations listed in the CONTROL_FILES initialization parameter.

> **See Also:** The description of CONFIGURE CONTROLFILE AUTOBACKUP FORMAT in the entry for CONFIGURE in *Oracle Database Backup and Recovery Reference* to learn how to determine the correct value for the autobackup format. See "Determining the DBID of the Database" on page 17-5 to learn how to determine your DBID.

Recovery When Using a Flash Recovery Area

The commands for restoring a control file are the same whether or not the database uses a flash recovery area. If the database uses a recovery area, then RMAN updates a control file restored from backup by crosschecking all disk-based backups and image copies recorded in the control file. RMAN catalogs any backups in the recovery area that are not recorded. As a result, the restored control file has a complete and accurate record of all backups in the recovery area and any other backups known to the control file at the time of the backup.

RMAN does not automatically crosscheck tape backups after restoring a control file. If you are using tape backups, then you can restore and mount the control file, and optionally crosscheck the backups on tape, as shown in the following example:

```
CROSSCHECK BACKUP DEVICE TYPE sbt;
```

Performing Recovery with a Backup Control File and No Recovery Catalog

This section assumes that you have RMAN backups of the control file, but do not use a recovery catalog. It also assumes that you enabled the control file autobackup feature for the target database and can restore an autobackup of the control file.

Because the autobackup uses a well-known format, RMAN can restore it even though it does not have a repository available that lists the available backups. You can restore the autobackup to the default or a new location. RMAN replicates the control file to all CONTROL_FILES locations automatically.

> **Note:** If you know the backup piece name that contains the control file (for example, from the media manager or because the piece is on disk), then you can specify the piece name using the RESTORE CONTROLFILE FROM *'filename'* command. The database records the location of every autobackup in the alert log.

Because you are not connected to a recovery catalog, the RMAN repository contains only information about available backups at the time of the control file backup. If you know the location of other usable backup sets or image copies, then add them to the control file RMAN repository with the CATALOG command.

To recover the database with a control file autobackup in NOCATALOG mode:

1. Start RMAN and connect to a target database.
2. Start the target database instance without mounting the database. For example:

   ```
   STARTUP NOMOUNT;
   ```

3. Set the database identifier for the target database with SET DBID.

   RMAN displays the DBID whenever you connect to a target database. You can also obtain it by inspecting saved RMAN log files, querying the catalog, or looking at the filenames of control file autobackup. For example, run:

   ```
   SET DBID 676549873;
   ```

4. Write an RMAN command file to restore the autobackup control file and perform recovery.

   The command file should contain the following steps:

   a. Optionally, specify the most recent backup time stamp that RMAN can use when searching for a control file autobackup to restore.

   b. If you know that a different control file autobackup format was in effect when the control file autobackup was created, then specify a nondefault format for the restore of the control file.

   c. If an **SBT** channel created the control file autobackup, then allocate one or more SBT channels. Because no recovery catalog is available, you cannot use preconfigured channels.

   d. Restore the autobackup of the control file, optionally setting the maximum number of days backward that RMAN can search and the initial sequence number that it should use in its search for the first day.

   e. If you know that the control file contained information about configured channels that will be useful to you in the rest of the restore process, then you can exit RMAN to clear manually allocated channels from step c.

      If you restart the RMAN client and mount the database, then these configured channels are available for your use. If you do not care about using configured channels from your control file, then you can simply mount the database.

   f. This step depends on whether the online redo logs are available. Note that OPEN RESETLOGS is always required after recovery with a backup control file, regardless of whether logs are available.

      If the online redo logs are usable, then RMAN can find and apply these logs. Perform a complete restore and recovery as described in "Performing Complete Database Recovery" on page 17-9.

      If the online redo logs are unusable, then perform DBPITR as described in "Performing Database Point-in-Time Recovery" on page 16-14. An UNTIL clause is required to specify a target time, SCN or log sequence number for the recovery prior to the first SCN of the online redo logs (otherwise, RMAN issues the RMAN-6054 error).

> **Note:** When specifying log sequences, if the last created archived redo log has sequence *n*, then specify `UNTIL SEQUENCE n+1` so that RMAN will apply *n* and then stop.

In the following example, the online redo log files have been lost, and the most recent archived redo log sequence number is 13243. This example shows how to restore the control file autobackup and recover through the latest log.

```
RUN
{
  # Optionally, set upper limit for eligible time stamps of control file
  # backups
  # SET UNTIL TIME '09/10/2007 13:45:00';
  # Specify a nondefault autobackup format only if required
  # SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK
  #   TO '?/oradata/%F.bck';
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt PARMS '...'; # allocate manually
  RESTORE CONTROLFILE FROM AUTOBACKUP
    MAXSEQ 100           # start at sequence 100 and count down
    MAXDAYS 180;         # start at UNTIL TIME and search back 6 months
  ALTER DATABASE MOUNT DATABASE;
}
# Now use automatic channels configured in restored control file
RESTORE DATABASE UNTIL SEQUENCE 13244;
RECOVER DATABASE UNTIL SEQUENCE 13244;
```

5. If recovery was successful, then open the database and reset the online logs:

```
ALTER DATABASE OPEN RESETLOGS;
```

Performing Disaster Recovery

Disaster recovery includes the restore of and recovery of the target database after the loss of the entire target database, the recovery catalog database, all current control files, all online redo log files, and all parameter files.

Prerequisites of Disaster Recovery

To perform a disaster recovery, you must have the following:

- Backups of all datafiles
- All archived redo logs generated after the creation time of the oldest backup that you intend to restore
- At least one control file autobackup
- A record of the DBID of the database

Recovering the Database After a Disaster

The procedure for disaster recovery is similar to the procedure for recovering the database with a backup control file in `NOCATALOG` mode. If you are restoring the database to a new host, then you should also review the considerations described in "Restoring a Database on a New Host" on page 19-10.

This scenario assumes that the Linux server on which your database was running has been damaged beyond repair. Fortunately, you backed up the database to Oracle Secure Backup and have the tapes available. The scenario assumes the following:

- Oracle Database is already installed on the new host.
- You are restoring the database to a new Linux host with the same directory structure as the old host.
- You have one tape drive containing backups of all the datafiles and archived redo logs through log 1124, as well as autobackups of the control file and server parameter file.
- You do not use a recovery catalog with the database.

To recover the database on the new host:

1. If possible, restore or re-create all relevant network files such as `tnsnames.ora` and `listener.ora` and a password file.
2. Start RMAN and connect to the target database instance.

   At this stage, no initialization parameter file exists. If you have set `ORACLE_SID` and `ORACLE_HOME`, then you can use operating system authentication to connect as `SYSDBA`. For example, start RMAN as follows:

   ```
   % rman
   RMAN> CONNECT TARGET /
   ```

3. Specify the DBID for the target database with the `SET DBID` command, as described in "Restoring the Server Parameter File" on page 19-2.

   For example, enter the following command:

   ```
   SET DBID 676549873;
   ```

4. Run the `STARTUP NOMOUNT` command.

   When the server parameter file is not available, RMAN attempts to start the instance with a dummy server parameter file.

5. Allocate a channel to the media manager and then restore the server parameter file from autobackup.

   For example, enter the following command to restore the server parameter file from Oracle Secure Backup:

   ```
   RUN
   {
     ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
     RESTORE SPFILE FROM AUTOBACKUP;
   }
   ```

6. Restart the instance with the restored server parameter file.

   ```
   STARTUP FORCE NOMOUNT;
   ```

7. Write the a command file to perform the restore and recovery, and then execute the command file. The command file should do the following:

   a. Allocate a channel to the media manager.

   b. Restore a control file autobackup (see "Performing Recovery with a Backup Control File and No Recovery Catalog" on page 19-6).

c. Mount the restored control file.

d. Catalog any backups not recorded in the repository with the CATALOG command.

e. Restore the datafiles to their original locations. If volume names have changed, then run SET NEWNAME commands before the restore and perform a switch after the restore to update the control file with the new locations for the datafiles, as shown in the following example.

f. Recover the datafiles. RMAN stops recovery when it reaches the log sequence number specified.

```
RMAN> RUN
{
  # Manually allocate a channel to the media manager
  ALLOCATE CHANNEL t1 DEVICE TYPE sbt;
  # Restore autobackup of the control file. This example assumes that you have
  # accepted the default format for the autobackup name.
  RESTORE CONTROLFILE FROM AUTOBACKUP;
  #  The set until command is used in case the database
  #  structure has changed in the most recent backups, and you wish to
  #  recover to that point-in-time. In this way RMAN restores the database
  #  to the same structure that the database had at the specified time.
  ALTER DATABASE MOUNT;
  SET UNTIL SEQUENCE 1124 THREAD 1;
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

The following example of the RUN command shows the same scenario except with new filenames for the restored datafiles:

```
RMAN> RUN
{
  #  If you need to restore the files to new locations,
  #  use SET NEWNAME commands:
  SET NEWNAME FOR DATAFILE 1 TO '/dev/vgd_1_0/rlvt5_500M_1';
  SET NEWNAME FOR DATAFILE 2 TO '/dev/vgd_1_0/rlvt5_500M_2';
  SET NEWNAME FOR DATAFILE 3 TO '/dev/vgd_1_0/rlvt5_500M_3';
  ALLOCATE CHANNEL t1 DEVICE TYPE sbt;
  RESTORE CONTROLFILE FROM AUTOBACKUP;
  ALTER DATABASE MOUNT;
  SET UNTIL SEQUENCE 124 THREAD 1;
  RESTORE DATABASE;
  SWITCH DATAFILE ALL; # Update control file with new location of datafiles.
  RECOVER DATABASE;
}
```

8. If recovery was successful, then open the database and reset the online logs:

```
ALTER DATABASE OPEN RESETLOGS;
```

Restoring a Database on a New Host

If your goal is to perform a test run of your disaster recovery procedures, or to permanently move a database to a new host, then you can use the procedure in this section. This procedure uses the RESTORE and RECOVER commands.

If you use the procedure in this section, then the DBID for the restored database will be the same as the DBID for the original database. You should not register a test database

created in this way in the same recovery catalog as the source database. Because the DBID of the two databases is the same, the metadata for the test database can interfere with RMAN's ability to restore and recover the source database.

If your goal is to create a new copy of your target database for ongoing use on a new host, then use the RMAN `DUPLICATE` command instead of this procedure. The `DUPLICATE` command assigns a new DBID to the database it creates, enabling it to be registered in the same recovery catalog as the original database.

> **See Also:** "Overview of RMAN Database Duplication" on page 23-1 to learn how to duplicate a database

Preparing to Restore a Database on a New Host

To prepare for the restore of the database to a new host, take the following steps:

- Record the DBID for your source database. If you do not know the DBID for your database, then see "Determining the DBID of the Database" on page 17-5 to learn how to determine the DBID.
- Make the source database initialization parameter file accessible on the new host. Copy the file from the old host to a new host by using an operating system utility.
- If you perform a test restore only, then make sure that RMAN is **not** connected to the recovery catalog. Otherwise, RMAN records metadata about the restored datafiles in the recovery catalog. This metadata interferes with future attempts to restore and recover the primary database.

  If you must use a recovery catalog because the control file is not large enough to contain the RMAN repository data on all of the backups that you need to restore, then use Oracle Data Pump to export the catalog and import it into a different schema or database. Afterward, use the copied recovery catalog for the test restore. Otherwise, the recovery catalog considers the restored database as the current target database.
- Make sure backups used for the restore are accessible on the restore host. For example, if the backups were made with a media manager, then make sure the tape device is connected to the new host. If you are using disk copies, then use the procedure in the following section.
- If you are performing a trial restore of the production database, then perform either of the following actions before restoring the database in the test environment:
  - If the test database will use a flash recovery area that is physically *different* from the recovery area used by the production database, then set `DB_RECOVERY_FILE_DEST` in the test database instance to the new location.
  - If the test database will use a flash recovery area that is physically the *same* as the recovery area used by the production database, then set `DB_UNIQUE_NAME` in the test database instance to a different name from the production database.

  If you do not perform either of the preceding actions, then RMAN assumes that you are restoring the production database and deletes flashback logs from flash recovery area because they are considered unusable.

Restoring Disk Backups to a New Host

To move the database to a new host by means of datafile copies or backup sets on disk, you must transfer the files manually to the new host. This example assumes that RMAN is using a recovery catalog.

To restore backup files to a new host:

1. Start RMAN and connect to a target database and recovery catalog.
2. Run a `LIST` command to see a listing of backups of the datafile and control file autobackups.

   For example, enter the following command to view datafile copies:

   ```
   LIST COPY;
   ```

   For example, enter the following command to view control file backups:

   ```
   LIST BACKUP OF CONTROLFILE;
   ```

   The piece name of the autobackup must use the `%F` substitution variable, so the autobackup piece name will include the string `c-IIIIIIIIII-YYYYMMDD-QQ`, where `IIIIIIIIII` stands for the DBID, `YYYYMMDD` is a time stamp in the Gregorian calendar of the day the backup is generated, and `QQ` is the sequence in hexadecimal.
3. Copy the backups to the new host with an operating system utility.

   Enter a command such as the following to copy all datafile copies to the `?/oradata/trgt` directory on the new host:

   ```
   % cp -r /disk1/*dbf /net/new_host/oracle/oradata/trgt
   ```

   Enter a command such as the following to copy the autobackup backup piece to the `/tmp` directory on the new host:

   ```
   % cp -r /disk1/auto_bkp_loc/c-1618370911-20070208-00 /net/new_host/tmp
   ```

   As explained in "Restoring the Server Parameter File from a Control File Autobackup" on page 19-3, you will need to use the `SET CONTROLFILE AUTOBACKUP FORMAT` command when restoring an autobackup from a nondefault location.

Testing the Restore of a Database on a New Host

This scenario assumes that you want to test whether you can restore your database to a new host. The scenario assumes that you have two networked Linux hosts, `hosta` and `hostb`. A target database named `trgta` is on `hosta` and is registered in recovery catalog `catdb`. You want to test the restore and recovery of `trgta` on `hostb`, while keeping database `trgta` up and running on `hosta`.

For the sake of illustration, assume that the directory structure of `hostb` is different from `hosta`. The target database is located in `/net/hosta/dev3/oracle/dbs`, but you want to restore the database to `/net/hostb/oracle/oradata/test`. You have tape backups of datafiles, control files, archived redo logs, and the server parameter file on a media manager accessible by both hosts. The `ORACLE_SID` for the `trgta` database is `trgta` and will not change for the restored database

---

Caution: If you are restoring the database for test purposes, then *never* connect RMAN to the test database and the recovery catalog.

To restore the database on a new host:

1. Ensure that the backups of the target database are accessible on the new host.

   To test disaster recovery, you need to have a recoverable backup of the target database. When preparing your disaster recovery strategy, ensure that the backups of the datafiles, control files, and server parameter file are restorable on `hostb`. Thus, you must configure the media management software so that `hostb` is a media manager client and can read the backup sets created on `hosta`. Consult the media management vendor for support on this issue.

2. Configure the `ORACLE_SID` on `hostb`.

   This scenario assumes that you want to start the RMAN client on `hostb` and authenticate yourself through the operating system. However, you must be connected to `hostb` either locally or through a net service name.

   After logging in to `hostb` with administrator privileges, edit the `/etc/group` file so that you are included: in the DBA group:

   ```
   dba:*:614:<your_user_name>
   ```

   Set the `ORACLE_SID` environment variable on `hostb` to the same value used on `hosta`:

   ```
   % setenv ORACLE_SID trgta
   ```

3. Start RMAN on `hostb` and connect to the target database *without* connecting to the recovery catalog.

   For example, enter the following command:

   ```
   % rman NOCATALOG
   RMAN> CONNECT TARGET /
   ```

4. Set the DBID and start the database instance without mounting the database.

   For example, run `SET DBID` to set the DBID, then run `STARTUP NOMOUNT`:

   ```
   SET DBID 1340752057;
   STARTUP NOMOUNT
   ```

   RMAN will fail to find the server parameter file, which has not yet been restored, but will start the instance with a "dummy" file. Sample output follows:

   ```
   startup failed: ORA-01078: failure in processing system parameters
   LRM-00109: could not open parameter file '/net/hostb/oracle/dbs/inittrgta.ora'

   trying to start the Oracle instance without parameter files ...
   Oracle instance started
   ```

5. Restore and edit the server parameter file.

   Because you enabled the control file autobackup feature when making your backups, the server parameter file is included in the backup. If you are restoring an autobackup that has a nondefault format, then use the `SET CONTROLFILE AUTOBACKUP FORMAT` command to indicate the format.

   Allocate a channel to the media manager, then restore the server parameter file as a client-side parameter file and use the `SET` command to indicate the location of the autobackup (in this example, the autobackup is in `/tmp`):

   ```
   RUN
   {
     ALLOCATE CHANNEL c1 DEVICE TYPE sbt PARMS '...';
   ```

```
  SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/tmp/%F';
  RESTORE SPFILE
    TO PFILE '?/oradata/test/inittrgta.ora'
    FROM AUTOBACKUP;
  SHUTDOWN ABORT;
}
```

6. Edit the restored initialization parameter file.

   Change any location-specific parameters, for example, those ending in _DEST, to reflect the new directory structure. For example, edit the following parameters:

   ```
   - IFILE
   - LOG_ARCHIVE_DEST_1
   - CONTROL_FILES
   ```

7. Restart the instance with the edited initialization parameter file.

   For example, enter the following command:

   ```
   STARTUP FORCE NOMOUNT PFILE='?/oradata/test/inittrgta.ora';
   ```

8. Restore the control file from an autobackup and then mount the database.

   For example, enter the following command:

   ```
   RUN
   {
     ALLOCATE CHANNEL c1 DEVICE TYPE sbt PARMS '...';
     RESTORE CONTROLFILE FROM AUTOBACKUP;
     ALTER DATABASE MOUNT;
   }
   ```

   RMAN restores the control file to whatever locations you specified in the CONTROL_FILES initialization parameter.

9. Catalog the datafile copies that you copied in "Restoring Disk Backups to a New Host" on page 19-12, using their new filenames or CATALOG START WITH (if you know all the files are in directories with a common prefix easily addressed with a CATALOG START WITH). For example, run:

   ```
   CATALOG START WITH '/oracle/oradata/trgt/';
   ```

   If you want to specify files individually, then you can execute a CATALOG command as follows:

   ```
   CATALOG DATAFILECOPY
     '/oracle/oradata/trgt/system01.dbf', '/oracle/oradata/trgt/undotbs01.dbf',
     '/oracle/oradata/trgt/cwmlite01.dbf', '/oracle/oradata/trgt/drsys01.dbf',
     '/oracle/oradata/trgt/example01.dbf', '/oracle/oradata/trgt/indx01.dbf',
     '/oracle/oradata/trgt/tools01.dbf', '/oracle/oradata/trgt/users01.dbf';
   ```

10. Start a SQL*Plus session on the new database and query the database filenames recorded in the control file.

    Because the control file is from the trgta database, the recorded filenames use the original hosta filenames. You can query V$ views to obtain this information. Run the following query in SQL*Plus:

    ```
    COLUMN NAME FORMAT a60
    SPOOL  LOG '/tmp/db_filenames.out'
    SELECT FILE# AS "File/Grp#", NAME
    FROM   V$DATAFILE
    ```

```
UNION
SELECT GROUP#,MEMBER
FROM   V$LOGFILE;
SPOOL OFF
EXIT
```

11. Write the RMAN restore and recovery script. The script must include the following steps:

    a. For each datafile on the destination host that is restored to a different path than it had on the source host, use a `SET NEWNAME` command to specify the new path on the destination host. If the file systems on the destination system are set up to have the same paths as the source host, then do not use `SET NEWNAME` for those files restored to the same path as on the source host.

    b. For each online redo log that is to be created at a different location than it had on the source host, use SQL `ALTER DATABASE RENAME FILE` commands to specify the pathname on the destination host. If the file systems on the destination system are set up to have the same paths as the source host, then do not use `ALTER DATABASE RENAME FILE` for those files restored to the same path as on the source host.

    c. Perform a `SET UNTIL` to limit recovery to the end of the archived redo logs. Note that recovery stops with an error if no `SET UNTIL` is specified.

    d. Restore and recover the database.

    e. Run `SWITCH DATAFILE ALL` so that the control file recognizes the new path names as the official new names of the datafiles.

    Example 19–3 shows the RMAN script `reco_test.rman` that can perform the restore and recovery.

Example 19–3 Restoring a Database on a New Host

```
RUN
{
  # allocate a channel to the tape device
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt PARMS '...';

  # rename the datafiles and online redo logs
  SET NEWNAME FOR DATAFILE 1 TO '?/oradata/test/system01.dbf';
  SET NEWNAME FOR DATAFILE 2 TO '?/oradata/test/undotbs01.dbf';
  SET NEWNAME FOR DATAFILE 3 TO '?/oradata/test/cwmlite01.dbf';
  SET NEWNAME FOR DATAFILE 4 TO '?/oradata/test/drsys01.dbf';
  SET NEWNAME FOR DATAFILE 5 TO '?/oradata/test/example01.dbf';
  SET NEWNAME FOR DATAFILE 6 TO '?/oradata/test/indx01.dbf';
  SET NEWNAME FOR DATAFILE 7 TO '?/oradata/test/tools01.dbf';
  SET NEWNAME FOR DATAFILE 8 TO '?/oradata/test/users01.dbf';
  SQL "ALTER DATABASE RENAME FILE ''/dev3/oracle/dbs/redo01.log''
      TO ''?/oradata/test/redo01.log'' ";
  SQL "ALTER DATABASE RENAME FILE ''/dev3/oracle/dbs/redo02.log''
      TO ''?/oradata/test/redo02.log'' ";

  # Do a SET UNTIL to prevent recovery of the online logs
  SET UNTIL SCN 123456;
  # restore the database and switch the datafile names
  RESTORE DATABASE;
  SWITCH DATAFILE ALL;

  # recover the database
```

```
  RECOVER DATABASE;
}
EXIT
```

12. Execute the script created in the previous step.

For example, start RMAN to the target database and run the `@` command:

```
% rman TARGET / NOCATALOG
RMAN> @reco_test.rman
```

13. Open the restored database with the `RESETLOGS` option.

From the RMAN prompt, open the database with the `RESETLOGS` option:

```
ALTER DATABASE OPEN RESETLOGS;
```

> **Caution:** When you re-open your database in the next step, do not connect to the recovery catalog. Otherwise, the new database incarnation created is registered automatically in the recovery catalog, and the filenames of the production database are replaced by the new filenames specified in the script.

14. Optionally, delete the test database with all of its files.

> **Note:** If you used an ASM disk group, then `DROP DATABASE` is the only way to safely remove the files of the test database. If you restored to non-ASM storage then you can also use operating system commands to remove the database.

Use the `DROP DATABASE` command to delete all files associated with the database automatically. The following example deletes the database files:

```
STARTUP FORCE NOMOUNT PFILE='?/oradata/test/inittrgta.ora';
DROP DATABASE;
```

Because you did not perform the restore and recovery when connected to the recovery catalog, the recovery catalog contains no records for any of the restored files or the procedures performed during the test. Likewise, the control file of the `trgta` database is completely unaffected by the test.

20

Performing RMAN Tablespace Point-in-Time Recovery (TSPITR)

This chapter explains how to perform RMAN tablespace point-in-time recovery. This chapter contains the following sections:

- Overview of RMAN TSPITR
- Prerequisites and Consequences of TSPITR
- Planning and Preparing for TSPITR
- Performing Fully Automated RMAN TSPITR
- Performing Customized RMAN TSPITR with an RMAN-Managed Auxiliary Instance
- Performing RMAN TSPITR Using Your Own Auxiliary Instance
- Troubleshooting RMAN TSPITR

Overview of RMAN TSPITR

This section explains the basic concepts and tasks involved in RMAN **tablespace point-in-time recovery (TSPITR)**.

Purpose of RMAN TSPTIR

Recovery Manager (RMAN) automatic TSPITR enables you to quickly recover one or more tablespaces in a database to an earlier time without affecting the rest of the tablespaces and objects in the database.

RMAN TSPITR is most useful for the following situations:

- You want to recover a logical database to a point different from the rest of the physical database, when multiple logical databases exist in separate tablespaces of one physical database. For example, you maintain logical databases in the `orders` and `personnel` tablespaces. An incorrect batch job or DML statement corrupts the data in only one of the tablespaces.
- You want to recover data lost after DDL operations that change the structure of tables. You cannot use Flashback Table to rewind a table to before the point of a structural change such as a truncate table operation.
- You want to recover a table after it has been dropped with the `PURGE` option.
- You want to recover from the logical corruption of a table.

You can also use Flashback Database to rewind data, but you must rewind the entire database rather than just a subset. Also, unlike TSPITR, the Flashback Database feature necessitates the overhead of maintaining flashback logs. The point in time to which you can flash back the database is more limited than the TSPITR window, which extends back to your earliest recoverable backup.

Basic Concepts of RMAN TSPITR

You perform TSPITR by using the RMAN `RECOVER TABLESPACE` command. The **target instance** contains the tablespace to be recovered to the **target time**. The target time is the point in time or SCN of the tablespace after TSPITR completes.

The **auxiliary instance** is a database instance used in the recovery process to perform the work of recovery. The auxiliary instance has other files associated with it, such as a control file, parameter file, and online logs, but they are not part of the auxiliary set.

The **recovery set** includes the datafiles that are in the tablespaces that you intend to recover. The **auxiliary set** includes the datafiles required for TSPITR of the recovery set that are not themselves part of the recovery set. The auxiliary set typically includes:

- The `SYSTEM` and `SYSAUX` tablespaces
- Datafiles containing rollback or undo segments from the target database instance
- Temporary tablespaces

The **auxiliary destination** is an optional location on disk which can be used to store any of the auxiliary set datafiles, control files and online redo logs of the auxiliary instance during TSPITR. Files stored here can be deleted after TSPITR is complete.

The simplest form of TSPITR, which is described in "Performing Fully Automated RMAN TSPITR" on page 20-8, involves specifying the tablespaces of the recovery set and the target time. RMAN automatically performs the following actions:

1. Queries `SYS.TS_PITR_CHECK` for the tablespaces in the recovery set, as shown in Example 20–1. If the query returns rows, then RMAN does not proceed with the TSPITR.
2. Creates the auxiliary instance, starts it, and connects to it (if there is no existing auxiliary instance).
3. Takes the tablespaces to be recovered offline in the target database.
4. Restores a backup control file from a point in time before the target time to the auxiliary instance.
5. Restores the datafiles from the recovery set and the auxiliary set to the auxiliary instance.

   Files are restored either in locations you specify for each file, or the original location of the file (for recovery set files) or in the auxiliary destination (for auxiliary set files, if you used the `AUXILIARY DESTINATION` argument of `RECOVER TABLESPACE`).
6. Recovers the restored datafiles in the auxiliary instance to the specified time.
7. Opens the auxiliary database with the `RESETLOGS` option.
8. Exports the dictionary metadata about objects in the recovered tablespaces to the target database.
9. Shuts down the auxiliary instance.

10. Issues `SWITCH` commands on the target database instance if new names were given to the datafiles in the recovery set. The target database control file now points to the datafiles in the recovery set that were just recovered at the auxiliary instance.
11. Imports the dictionary metadata from the auxiliary instance to the target instance, allowing the recovered objects to be accessed.
12. Deletes all auxiliary set files.

At this point the TSPITR is complete. The recovery set datafiles are returned to their contents at the specified point in time, and belong to the target database.

Basic Steps of RMAN TSPITR

Before performing TSPITR, review "Prerequisites and Consequences of TSPITR" on page 20-3 to determine whether TSPITR is a viable option. If you decide to perform TSPITR, then you can then proceed to the preparatory stage described "Planning and Preparing for TSPITR" on page 20-5.

When you are ready to perform the actual TSPITR, the basic steps depend on which technique you use. The following sections describe your options:

- Performing Fully Automated RMAN TSPITR

  You specify an auxiliary destination and allow RMAN to manage all aspects of the TSPITR. This is the simplest technique and is recommended unless you specifically need more control over the operation.

- Performing Customized RMAN TSPITR with an RMAN-Managed Auxiliary Instance

  You base your TSPITR on the behavior of fully automated TSPITR, possibly still using an auxiliary destination, but customize one or more aspects of the behavior. For example, you could specify the location of auxiliary set or recovery set files, or specify initialization parameters or channel configurations for the auxiliary instance created and managed by RMAN.

- Performing RMAN TSPITR Using Your Own Auxiliary Instance

  In this technique, you take responsibility for setting up, starting, stopping and cleaning up the auxiliary instance used in TSPITR, and possibly also manage the TSPITR process using some of the methods available in customized TSPITR with an automatic auxiliary instance.

Prerequisites and Consequences of TSPITR

A number of database problems cannot be resolved with TSPITR. The following list explains TSPITR prerequisites in terms of when you *cannot* perform TSPITR:

- You cannot perform TSPITR if you do not have archived redo logs or if the database runs in `NOARCHIVELOG` mode.
- You cannot recover dropped tablespaces.
- You cannot recover a renamed tablespace to a point in time before it was renamed. If you try to perform a TSPITR to an SCN earlier than the rename operation, then RMAN cannot find the new tablespace name in the repository as of that earlier SCN (because the tablespace did not have that name at that SCN).

In this situation, you must recover the entire database to a point in time before the tablespace was renamed. The tablespace will be found under the name it had at that earlier time.

- If the constraints for the tables in tablespace `tbs1` are contained in tablespace `tbs2`, then you cannot recover `tbs1` without recovering `tbs2` as well.
- You cannot use TSPITR to recover any of the following objects:
  - Replicated master tables
  - Partial tables (for example, if you perform RMAN TSPITR on partitioned tables and spread partitions across multiple tablespaces, then you must recover all tablespaces which include partitions of the table.)
  - Tables with `VARRAY` columns, nested tables, or external files
  - Snapshot logs and snapshot tables
  - Tablespaces containing undo or rollback segments
  - Tablespaces that contain objects owned by `SYS`, including rollback segments

Consequences of TSPITR

After TSPITR completes, RMAN recovers the datafiles in the recovery set to the target time. Note the following consequences of TSPITR:

- If a datafile was added after the point to which RMAN is recovering, then an empty datafile by the same name will be included in the tablespace after RMAN TSPITR.
- TSPITR will not recover query optimizer statistics for recovered objects.

  You must gather new statistics after the TSPITR completes.
- If you run TSPITR on a tablespace and bring the tablespace online at time *t*, then backups of the tablespace created before time *t* are no longer usable for recovery with a current control file.

  You cannot run TSPITR again on this tablespace to recover it to any time less than or equal to time *t*, nor can you use the current control file to recover the database to any time less than or equal to *t*. Therefore, you must back up the recovered tablespace as soon as TSPITR is complete.

Special Considerations When Not Using a Recovery Catalog

If you do not use a recovery catalog when performing TSPITR, then note the following special considerations:

- The undo segments at the time of the TSPITR must be part of the auxiliary set. Because RMAN has no historical record of the undo in the control file, RMAN assumes that the current rollback or undo segments were the same segments present at the time to which recovery is performed. If the undo segments have changed since that time, then TSPITR will fail.
- TSPITR to a time that is too old may not succeed if Oracle has reused the control file records for needed backups. (In planning your database, set the `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter to a value large enough to ensure that control file records needed for TSPITR are kept.)
- Assume that you run TSPITR on a tablespace, and then bring the tablespace online at time *t*. When not using a recovery catalog, the current control file has no record

of the older incarnation of the recovered tablespace. Thus, recovery with a current control file that involves this tablespace cannot use a backup taken prior to time *t*. You can, however, perform incomplete recovery of the whole database to any time less than or equal to *t*, if you can restore a backup control file from before time *t*.

Planning and Preparing for TSPITR

This section assumes that you have read "Prerequisites and Consequences of TSPITR" on page 20-3. You must carry out the following steps when preparing for TSPITR:

- Choosing the Right Target Time for TSPITR
- Determining the Recovery Set
- Identifying and Preserving Objects That Will Be Lost After TSPITR

Choosing the Right Target Time for TSPITR

It is extremely important that you choose the right target time or SCN for your TSPITR. As noted in "Prerequisites and Consequences of TSPITR" on page 20-3, after you bring a tablespace online after TSPITR, you cannot use any backup from a time earlier than the moment you brought the tablespace online. In practice, you cannot make a second attempt at TSPITR if you choose the wrong target time the first time, unless you are using a recovery catalog.

If you have a recovery catalog, then you can perform repeated TSPITR operations to different target times because the catalog contains tablespace history information. If RMAN uses only a control file, however, then it does not have the tablespace history. In this case, RMAN knows only the current set of tablespaces. The tablespace on which TSPITR was performed has a creation time of the time it was brought online.

Assume a situation in which you are *not* using a recovery catalog. You run TSPITR on a tablespace and then bring the tablespace online at 5 p.m. on Friday. Backups of the tablespace created before 5 p.m. Friday are no longer usable for recovery with a current control file. You cannot run TSPITR again on this tablespace with a target time earlier than 5 p.m. Friday, nor can you use the current control file to recover the database to any time earlier than 5 p.m. Friday. Your only option is point-in-time recovery of the entire database using a restored control file.

To investigate past states of your data to identify the target time for TSPITR, you can use Oracle Flashback Query, Oracle Transaction Query and Oracle Flashback Version Query to find the point in time when unwanted database changes occurred.

> **See Also:** *Oracle Database Advanced Application Developer's Guide* for more information on Flashback Query, Flashback Transaction Query, and Flashback Version Query

Determining the Recovery Set

Initially, your recovery set includes the datafiles for the tablespaces you intend to recover. However, if objects in the tablespaces you need have relationships (such as constraints) to objects in other tablespaces, then you will have to account for this relationship before you can perform TSPITR. You have the following choices when faced with such a relationship:

- Add the tablespace including the related objects to your recovery set
- Remove the relationship
- Suspend the relationship for the duration of TSPITR

Identifying and Resolving Dependencies on the Primary Database

You can use the TS_PITR_CHECK view to identify relationships between objects that span the recovery set boundaries. If this view returns rows when queried, then investigate and correct the problem. Proceed with TSPITR *only* when TS_PITR_CHECK view returns no rows for the tablespaces not in the recovery set. Record all actions performed during this step so that you can re-create any suspended or removed relationships after completing TSPITR.

The query in Example 20–1 illustrates how to use the TS_PITR_CHECK view. For an example with an initial recovery set consisting of tools and users, the SELECT statement against TS_PITR_CHECK would be as follows:

Example 20–1 Querying TS_PITR_CHECK for a Subset of Tablespaces

```
SELECT *
FROM SYS.TS_PITR_CHECK
WHERE (
        TS1_NAME IN ('USERS','TOOLS')
        AND TS2_NAME NOT IN ('USERS','TOOLS')
      )
OR    (
        TS1_NAME NOT IN ('USERS','TOOLS')
        AND TS2_NAME IN ('USERS','TOOLS')
      );
```

To run a complete TSPITR check on all the tablespaces in the database (not just the tablespaces in the recovery set), you can run the query in Example 20–2.

Example 20–2 Querying TS_PITR_CHECK for All Tablespaces

```
SELECT *
FROM SYS.TS_PITR_CHECK
WHERE (
         'SYSTEM' IN (TS1_NAME, TS2_NAME)
         AND TS1_NAME <> TS2_NAME
         AND TS2_NAME <> '-1'
      )
OR    (
         TS1_NAME <> 'SYSTEM'
         AND TS2_NAME = '-1'
      );
```

Because of the number and width of the columns in the TS_PITR_CHECK view, you may want to format the columns as follows when running the query:

```
SET LINESIZE 120
COLUMN OBJ1_OWNER HEADING "own1"
COLUMN OBJ1_OWNER FORMAT a6
COLUMN OBJ1_NAME HEADING "name1"
COLUMN OBJ1_NAME FORMAT a5
COLUMN OBJ1_SUBNAME HEADING "subname1"
COLUMN OBJ1_SUBNAME FORMAT a8
COLUMN OBJ1_TYPE HEADING "obj1type"
COLUMN OBJ1_TYPE FORMAT a8 word_wrapped
COLUMN TS1_NAME HEADING "ts1_name"
COLUMN TS1_NAME FORMAT a6
COLUMN OBJ2_NAME HEADING "name2"
COLUMN OBJ2_NAME FORMAT a5
COLUMN OBJ2_SUBNAME HEADING "subname2"
COLUMN OBJ2_SUBNAME FORMAT a8
```

```
COLUMN OBJ2_TYPE HEADING "obj2type"
COLUMN OBJ2_TYPE FORMAT a8 word_wrapped
COLUMN OBJ2_OWNER HEADING "own2"
COLUMN OBJ2_OWNER FORMAT a6
COLUMN TS2_NAME HEADING "ts2_name"
COLUMN TS2_NAME FORMAT a6
COLUMN CONSTRAINT_NAME HEADING "cname"
COLUMN CONSTRAINT_NAME FORMAT a5
COLUMN REASON HEADING "reason"
COLUMN REASON FORMAT a25 word_wrapped
```

Assume a case in which the partitioned table `tp` has two partitions, `p1` and `p2`, that exist in tablespaces `users` and `tools` respectively. Also assume that a partitioned index called `tpind` is defined on `tp`, and that the index has two partitions `id1` and `id2` (that exist in tablespaces `id1` and `id2` respectively). In this case, you would get the output shown in Example 20–3 when you run the query in Example 20–1.

Example 20–3 Output for Query of TS_PITR_CHECK

```
own1   name1 subname1 obj1type ts1_name name2 subname2 obj2type own2      ts2_name   cname reason
---    ----  -------  -------  -------  ----  -------  -------- ---       --------   ---   ------
SYSTEM  TP   P1       TABLE    USER     TPIND IP1      INDEX    PARTITION PARTITION  SYS   ID1 Partitioned
Objects not fully contained in the recovery set
SYSTEM  TP   P2       TABLE    TOOLS    TPIND IP2      INDEX    PARTITION PARTITION  SYS   ID2 Partitioned
Objects not fully contained in the recovery set
```

Example 20–3 shows that `SYSTEM.tp` has a partitioned index `tpind` that consists of two partitions, `ip1` in tablespace `id1` and `ip2` in tablespace `id2`. To perform TSPITR, you must either drop `tpind` or include `id1` and `id2` in the recovery set.

> **See Also:** *Oracle Database Reference* for more information about the `TS_PITR_CHECK` view

Identifying and Preserving Objects That Will Be Lost After TSPITR

When you perform RMAN TSPITR on a tablespace, objects created after the target recovery time are lost. You can preserve such objects after they are identified by exporting them before TSPITR with the Data Pump Export utility and re-importing them afterward with Data Pump Import.

To determine which objects will be lost in TSPITR, query the `TS_PITR_OBJECTS_TO_BE_DROPPED` view on the primary database. Table 20–1describes the contents of the view.

Table 20–1 TS_PITR_OBJECTS_TO_BE_DROPPED View

Column Name	Meaning
`OWNER`	Owner of the object to be dropped.
`NAME`	The name of the object that will be lost as a result of undergoing TSPITR
`CREATION_TIME`	Creation timestamp for the object.
`TABLESPACE_NAME`	Name of the tablespace containing the object.

Filter the view for objects whose `CREATION_TIME` is after the target time for TSPITR. For example, with a recovery set consisting of `users` and `tools`, and a recovery point in time of November 2, 2007, 7:03:11 AM, issue the statement shown in Example 20–4.

Example 20–4 Querying TS_PITR_OBJECTS_TO_BE_DROPPED

```
SELECT OWNER, NAME, TABLESPACE_NAME,
       TO_CHAR(CREATION_TIME, 'YYYY-MM-DD:HH24:MI:SS')
       FROM TS_PITR_OBJECTS_TO_BE_DROPPED
WHERE TABLESPACE_NAME IN ('USERS','TOOLS')
AND CREATION_TIME > TO_DATE('02-NOV-06:07:03:11','YY-MON-DD:HH24:MI:SS')
ORDER BY TABLESPACE_NAME, CREATION_TIME;
```

The `TO_CHAR` and `TO_DATE` functions are used to avoid issues with different national date formats. Of course, you can use local date formats in your own work.

> **See Also:** *Oracle Database Reference* for more information about the `TS_PITR_OBJECTS_TO_BE_DROPPED` view

Performing Fully Automated RMAN TSPITR

When you perform fully automated TSPITR, RMAN manages the entire process.

RMAN bases as much of the configuration for TSPITR as possible on the target database. During TSPITR, the recovery set datafiles are written in their current locations on the target database. The same channel configurations in effect on the target database are used on the auxiliary instance when restoring files from backup. Auxiliary set datafiles and other auxiliary instance files, however, are stored in the auxiliary destination.

Use the `AUXILIARY DESTINATION` parameter to set the auxiliary destination for RMAN to use for the auxiliary set datafiles and other files for the auxiliary instance. The auxiliary destination must be a location on disk with enough space to hold auxiliary set datafiles. Even if you use other techniques to rename some or all of the auxiliary set datafiles, specifying an `AUXILIARY DESTINATION` parameter provides a default location for auxiliary set datafiles for which names are not specified. TSPITR will not fail if you inadvertently do not provide names for all auxiliary set datafiles.

To perform fully automated RMAN TSPITR:

1. Review the information in "Prerequisites and Consequences of TSPITR" on page 20-3.
2. Perform the tasks in "Planning and Preparing for TSPITR" on page 20-5.
3. Start an RMAN session on the target database and, if applicable, connect to a recovery catalog.

   > **Note:** Do not connect to an auxiliary instance when starting the RMAN client for automated TSPITR. If RMAN is connected to an auxiliary instance when you run `RECOVER TABLESPACE`, then RMAN assumes that you are trying to manage your own auxiliary instance, as described in "Performing RMAN TSPITR Using Your Own Auxiliary Instance" on page 20-18, and will try to use the connected auxiliary for TSPITR.

4. Configure any channels required for the TSPITR on the target instance.

   The auxiliary instance uses the same channel configuration as the target instance when performing the TSPITR.

5. Run the `RECOVER TABLESPACE` command, specifying both the `UNTIL` clause and the `AUXILIARY DESTINATION` parameter.

Example 20–5 returns the `users` and `tools` tablespaces to the end of log sequence number 1300, and stores the auxiliary instance files in the `/disk1/auxdest` directory.

Example 20–5 Performing TSPITR on Two Tablespaces

```
RECOVER TABLESPACE users, tools
  UNTIL LOGSEQ 1300 THREAD 1
  AUXILIARY DESTINATION '/disk1/auxdest';
```

The next step depends on the results of the `RECOVER` command:

- If no error occurs during TSPITR, then proceed to step 6.

  The tablespaces are taken offline by RMAN, restored from backup and recovered to the desired point in time on the auxiliary instance, and then re-imported to the target database. The tablespaces are left offline. All auxiliary set datafiles and other auxiliary instance files are cleaned up from the auxiliary destination.

- In an error occurs during TSPITR, then proceed to "Troubleshooting RMAN TSPITR" on page 20-23.

  The auxiliary set datafiles and other auxiliary instance files will be left in place in the auxililary destination as an aid to troubleshooting. The state of the recovery set files is determined by the type of failure. After you resolve the problem, you can try TSPITR again.

  Note: The files in the auxiliary destination are Oracle Managed Files. The target database does not catalog them, so a second TSPITR will not use them. However, if you specify `SET NEWNAME` instead of `AUXILIARY DESTINATION` for the auxiliary set, and if the failure occurred before the online logs are reset, then RMAN can reuse the existing datafiles.

  ---

6. If TSPITR completes successfully, then back up the recovered tablespaces before bringing them online.

   For example, enter the following command:

   ```
   BACKUP TABLESPACE users, tools;
   ```

   After you perform TSPITR on a tablespace, you cannot use backups of that tablespace from before the TSPITR was completed and the tablespace brought back online. If you use the recovered tablespaces without taking a backup, then you are running your database without a usable backup of these tablespaces.

7. Bring the tablespaces back online.

   For example, enter the following command:

   ```
   RMAN> SQL "ALTER TABLESPACE users, tools ONLINE";
   ```

   Your recovered tablespaces are now ready for use.

Performing Customized RMAN TSPITR with an RMAN-Managed Auxiliary Instance

You can customize the following aspects of RMAN TSPITR while still mostly following the procedure described in "Performing Fully Automated RMAN TSPITR" on page 20-8:

- Rename files in an Oracle Managed Files format.

  This task is described in "Renaming Oracle Managed Files in TSPITR" on page 20-10.

- Rename or relocate your recovery set datafiles so that the datafiles making up the recovered tablespaces are not stored in the original locations after TSPITR. This may be necessary if the disk that originally contained the tablespace is not usable.

  This task is described in "Renaming TSPITR Recovery Set Datafiles with SET NEWNAME" on page 20-10.

- Specify a location other than the auxiliary destination for some or all auxiliary set datafiles. You might choose this option if no single location on disk has enough space for all auxiliary set files.

  This task is described in "Naming TSPITR Auxiliary Set Datafiles" on page 20-11.

- Set up image copy backups of your datafiles in advance to avoid having to restore datafiles during TSPITR.

  This task is described in "Using Image Copies for Faster RMAN TSPITR Performance" on page 20-14.

- Customize initialization parameters for your RMAN-managed auxiliary instance.

  This task is described in "Customizing Initialization Parameters for the Automatic Auxiliary Instance in TSPITR" on page 20-14.

Renaming Oracle Managed Files in TSPITR

If you customize TSPITR when datafiles or online redo log files are in an Oracle Managed Files (OMF) format, then you can use the following techniques to rename them. The naming techniques are listed in order from most recommended to least recommended:

1. Use an auxiliary destination, as described in "Performing Fully Automated RMAN TSPITR" on page 20-8.
2. Specify a location with one or more of the Oracle Managed Files initialization parameters for the auxiliary instance: `DB_CREATE_FILE_DEST`, `DB_RECOVERY_FILE_DEST`, and `DB_CREATE_ONLINE_LOG_DEST_n`. Do *not* set the `DB_FILE_NAME_CONVERT` or `LOG_FILE_NAME_CONVERT` initialization parameters.
3. Set the `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` initialization parameters for the auxiliary instance. For Oracle Managed Files in ASM, RMAN uses the pattern to convert the ASM disk group name *only* and generates a valid OMF filename in the converted disk group.

Renaming TSPITR Recovery Set Datafiles with SET NEWNAME

You may not want the recovery set datafiles restored and recovered in their original locations. The `SET NEWNAME` command enables you to specify a new destination. You can also use `CONFIGURE AUXNAME` to rename recovery set datafiles, but the effects of

are different. Refer to the discussion of "Using Image Copies for Faster RMAN TSPITR Performance" on page 20-14 for details about CONFIGURE AUXNAME.

To specify new recovery set filenames, create a RUN block and use SET NEWNAME commands within it. Be sure to assign names that do not conflict with each other or with the names of your current datafiles. Example 20–6 illustrates the basic technique.

Example 20–6 Renaming Recovery Set Files

```
RUN
{
   .
   .
   .
   SET NEWNAME FOR DATAFILE 'ORACLE_HOME/oradata/trgt/users01.dbf'
     TO '/newfs/users01.dbf';
   ...other setup commands...
   RECOVER TABLESPACE users, tools UNTIL SEQUENCE 1300 THREAD 1;
}
```

RMAN restores each specified datafile to the new location during TSPITR, recovers it in the new location, and updates the control file so that the newly recovered datafile replaces the old one in the control file. RMAN overwrites any existing image copy backup of a datafile found at the new specified location.

RMAN does not detect conflicts between names set with SET NEWNAME and current datafile names on the target database until the actual recovery. If RMAN detects a conflict, then TSPITR fails and RMAN reports an error. The valid datafile is not overwritten.

Naming TSPITR Auxiliary Set Datafiles

Unlike recovery set datafiles, which are usually stored in their original locations, auxiliary set datafiles must not overwrite the corresponding original files in the target database. If you do not specify an auxiliary set file location that is different from its original location, then TSPITR fails. The failure occurs when RMAN attempts to overwrite the corresponding file in the original database and discovers the file in use.

The simplest way to provide locations for auxiliary set datafiles is to specify an auxiliary destination for TSPITR. However, RMAN supports the following alternatives for controlling the location of auxiliary set datafiles, which are listed in order of precedence shown in Table 20–2.

Table 20–2 Order of Precedence for Naming Files

Order	Technique	Section
1	SET NEWNAME	"Using SET NEWNAME to Name Auxiliary Set Datafiles" on page 20-12
2	CONFIGURE AUXNAME	"Using SET NEWNAME and CONFIGURE AUXNAME with Auxiliary Set Image Copies" on page 20-15
3	DB_FILE_NAME_CONVERT	"Using DB_FILE_NAME_CONVERT to Name Auxiliary Set Datafiles" on page 20-12
4	AUXILIARY DESTINATION argument to RECOVER TABLESPACE	

Settings higher on the list override settings lower on the list in situations where both have been applied. For example, you might run `RECOVER TABLESPACE... AUXILIARY DESTINATION` on a target database when some auxiliary set datafiles have auxiliary names configured with `CONFIGURE AUXNAME`.

Even if you intend to use either of the preceding techniques to provide locations for specific files, Oracle recommends that you provide an `AUXILIARY DESTINATION` argument to `RECOVER TABLESPACE`. If you overlook renaming some auxiliary set datafiles, then TSPITR will still succeed. Any files not otherwise renamed will be placed in the auxiliary destination.

Note: You can view any current `CONFIGURE AUXNAME` settings by running the `SHOW AUXNAME` command, which is described in *Oracle Database Backup and Recovery Reference*.

Using SET NEWNAME to Name Auxiliary Set Datafiles

To specify a new name for an auxiliary set datafile, you can enclose `RECOVER TABLESPACE` in a `RUN` command and use a `SET NEWNAME` command within the `RUN` block to rename the file. Example 20–7 illustrates the basic technique.

Example 20–7 Renaming Auxiliary Set Files

```
RUN
{
  SET NEWNAME FOR DATAFILE '?/oradata/prod/system01.dbf'
    TO '/disk1/auxdest/system01.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/prod/sysaux01.dbf'
    TO '/disk1/auxdest/sysaux01.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/prod/undotbs01.dbf'
    TO '/disk1/auxdest/undotbs01.dbf';
  RECOVER TABLESPACE users, tools
    UNTIL LOGSEQ 1300 THREAD 1
    AUXILIARY DESTINATION '/disk1/auxdest';
}
```

The result depends on whether `/disk1/auxdest/system01.dbf` exists when `RECOVER TABLESPACE` is executed. If `?/oradata/system01.dbf` exists at the specified location and was created at an SCN before the `UNTIL` time for TSPITR, then the behavior is as described in "Using SET NEWNAME and CONFIGURE AUXNAME with Auxiliary Set Image Copies" on page 20-15. Otherwise, RMAN restores the auxiliary set datafile to the `NEWNAME` instead of the default location. If your intention is only to control where the auxiliary set datafiles are stored, then ensure that no file is stored at the location specified by `SET NEWNAME` before performing TSPITR.

Using DB_FILE_NAME_CONVERT to Name Auxiliary Set Datafiles

Assume that you do not want to use an auxiliary destination for all of your auxiliary set datafiles, but you also do not want to name every file individually. In this case, you can include a `DB_FILE_NAME_CONVERT` initialization parameter in the initialization parameter file used by the auxiliary instance. You can only use this technique in the following circumstances:

- You create your own initialization parameter file for an automatically managed auxiliary instance, as described in "Customizing Initialization Parameters for the Automatic Auxiliary Instance in TSPITR" on page 20-16.

- You create your own auxiliary instance, as described in "Performing RMAN TSPITR Using Your Own Auxiliary Instance" on page 20-18.

The DB_FILE_NAME_CONVERT initialization parameter in the auxiliary instance specifies how to derive names for files in the auxiliary instance from the original names of the corresponding files in the target database instance. The parameter consists of a list of pairs of strings. For any filename that contains the first string of a pair as a substring, the name of the corresponding file in the auxiliary instance is generated by substituting the second string of the pair into the original filename.

For example, assume that the target instance contains the following files:

- ?/oradata/trgt/system01.dbf of the SYSTEM tablespace
- ?/oradata/trgt/sysaux01.dbf of the SYSAUX tablespace
- ?/oradata/trgt/undotbs01.dbf of the undotbs tablespace

To place the corresponding files of the auxiliary instance in /bigtmp, then you would add the following line to the auxiliary instance parameter file:

```
DB_FILE_NAME_CONVERT=('?/oradata/trgt', '/bigtmp')
```

New filenames for the corresponding auxiliary instance files would be /bigtmp/trgt/system01.dbf, /bigtmp/trgt/sysaux01.dbf, and /bigtmp/trgt/undotbs01.dbf.

The most important point to remember is that DB_FILE_NAME_CONVERT **must** be present in the auxiliary instance parameter file. If the auxiliary instance was manually created, then add DB_FILE_NAME_CONVERT to the auxiliary instance parameter file.

Note that you can still rename individual auxiliary set datafiles with SET NEWNAME or CONFIGURE AUXNAME. Also, files that do not match the patterns provided in DB_FILE_NAME_CONVERT will not be renamed. You may wish to use the AUXILIARY DESTINATION parameter of RECOVER TABLESPACE to ensure that all auxiliary set datafiles are sent to some destination. If none of the renaming methods used provide a new name for a file at the auxiliary instance, then TSPITR will fail.

Renaming OMF Datafiles Using DB_FILE_NAME_CONVERT in TSPITR Oracle Managed Files (OMF) can exist in ASM or non-ASM storage. When the DB_FILE_NAME_CONVERT initialization parameter is set, TSPITR performs name conversion differently depending on whether OMF storage is ASM. For OMF files in ASM, the database converts only disk group names, for example, +DISK1 to +DISK2. For OMF files not stored in ASM, the database stores the files in the DB_FILE_NAME_CONVERT location and replaces the substring. It is not possible to generate valid non-ASM OMF filenames for the auxiliary instance by replacing a substring of the target instance OMF filename. Thus, you cannot use DB_FILE_NAME_CONVERT to control generation of new names in this situation.

To avoid this issue, use one of the other supported options for generating new names for OMF files (including files stored in ASM):

- Use an auxiliary destination, as described in "Performing Fully Automated RMAN TSPITR" on page 20-8.
- Use the DB_CREATE_FILE_DEST initialization parameter in the auxiliary instance parameter file to specify a location for all auxiliary instance files for which no new name is specified using SET NEWNAME or CONFIGURE AUXNAME.
- For ASM files, you can use SET NEWNAME to redirect individual files to a specific disk group accessible from the auxiliary instance (and allow the database to generate the filename within the disk group). For example:

```
RUN
{
  SET NEWNAME FOR DATAFILE 1 TO "+DISK2";
  SET NEWNAME FOR DATAFILE 2 TO "+DISK3";
  RECOVER TABLESPACE users, tools
    UNTIL LOGSEQ 1300 THREAD 1
    AUXILIARY DESTINATION '/disk1/auxdest';
}
```

Renaming Tempfiles During TSPITR Tempfiles are considered part of the auxiliary set for your database. When the auxiliary instance is instantiated, RMAN re-creates the temporary tablespaces of the target database and generates their names by means of the regular rules for the auxiliary datafile names.

You can rename tempfiles with the `SET NEWNAME FOR TEMPFILE`, `DB_FILE_NAME_CONVERT`, or `AUXILIARY DESTINATION` commands. When RMAN opens the auxiliary instance, RMAN re-creates the tempfiles according to the applicable renaming rule. When the auxiliary instance is cleaned up, the tempfiles are deleted along with the rest of the auxiliary instance files.

Using Image Copies for Faster RMAN TSPITR Performance

You can enhance TSPITR performance by redirecting RMAN to use existing image copies of the recovery set and auxiliary set datafiles. In this case, RMAN does not need to restore the datafiles from backup. You can use the following techniques to tell RMAN about the possible existence of an image copy of a datafile:

- Use `CONFIGURE AUXNAME` command with image copies of recovery set datafiles or auxiliary set datafiles
- Use `SET NEWNAME` command with image copies of auxiliary set datafiles

In general, if a suitable image copy is available in the specified location, then RMAN uncatalogs the image copy from the RMAN repository of the target instance and catalogs it in the control file of the auxiliary instance. The auxiliary instance then performs point-in-time recovery using the image copy. The details vary depending on the command used and whether the file is an auxiliary set or recovery set file.

Using CONFIGURE AUXNAME with Recovery Set Image Copies

During TSPITR, RMAN looks in the specified `AUXNAME` location for the datafile. RMAN checks whether an image copy backup of the datafile exists with a **datafile checkpoint** SCN early enough that it can be recovered to the target time. If RMAN finds a usable image copy, then RMAN uses it in TSPITR. Otherwise, RMAN restores the datafile and recovers it in its original location. Any file in the location specified by the `AUXNAME` is not changed or deleted. Example 20–8 illustrates this technique.

Example 20–8 Using CONFIGURE AUXNAME

```
CONFIGURE AUXNAME FOR DATAFILE 'ORACLE_HOME/oradata/trgt/users01.dbf'
  TO '/newfs/users1.dbf';
...other RMAN commands, if any...
RECOVER TABLESPACE users, tools UNTIL SEQUENCE 1300 THREAD 1;
```

The primary use of `CONFIGURE AUXNAME` is to make TSPITR faster by eliminating restore times. If you have tablespaces on which you anticipate performing TSPITR, then you can include in your backup routine the maintenance of a set of image copies of the affected datafiles, and update these periodically to the earliest point to which you expect to perform TSPITR. The expected usage model is:

1. Configure the AUXNAME for the files once, when setting up this strategy.
2. Perform BACKUP AS COPY DATAFILE *n* FORMAT *auxname* regularly to maintain the updated image copy. For better performance, use an incrementally updated backup strategy to keep the image copies up to date without performing full backups of the datafiles.
3. When TSPITR is needed, specify a target time after the last update of the image copy.

Remember that you may not know which tablespaces will require image copies in advance. As discussed in "Determining the Recovery Set" on page 20-5, relationships between the tablespaces to be recovered and other tablespaces may require that you add tablespaces to your final recovery set. Other tablespaces may wind up in the auxiliary set as well. You should configure an AUXNAME for each datafile that is likely to be part of your recovery set, and update image copies of all datafiles often.

It is possible for you not to correctly anticipate all tablespaces to include in the recovery set. Alternatively, for reasons of overhead you may not want to maintain copies of all possible recovery set tablespaces. In either case, you can prepare only a subset of the datafiles: the TSPITR process is still the same. The process simply takes longer because RMAN must recover recovery set datafiles for which there are no image copies in their original locations.

Note that the order of precedence of naming methods is still respected when you use CONFIGURE AUXNAME to rename a recovery set file. A SET NEWNAME for a recovery set file set as part of the TSPITR command overrides the effect of the persistent CONFIGURE AUXNAME command for the same file. Behavior in this instance will be as described in "Renaming TSPITR Recovery Set Datafiles with SET NEWNAME" on page 20-10. SET NEWNAME used with a recovery set file never refers to an image copy.

Using SET NEWNAME and CONFIGURE AUXNAME with Auxiliary Set Image Copies

As with recovery set datafiles, CONFIGURE AUXNAME sets a persistent alternative location for an auxiliary set datafile image copy, while SET NEWNAME sets an alternative location for the duration of a RUN command. Nevertheless, RMAN handles values for auxiliary set datafiles differently from recovery set datafiles. In short, auxiliary set files always use the AUXNAME or NEWNAME, but recovery set files only use the AUXNAME or NEWNAME if a valid copy exists.

Assume that you use SET NEWNAME or CONFIGURE AUXNAME to specify a new location for an auxiliary set datafile. Also assume that there is an image copy at that location with an SCN that can be used in TSPITR. In this case, RMAN uses the image copy. If there is no usable image copy at that location, however, then RMAN restores a usable copy from backup. (If an image copy is present but the SCN is after the target time for TSPITR, then the datafile is overwritten by the restored file.)

As with all auxiliary set files, the file is deleted after successful TSPITR. If TSPITR fails, the file is left for use in troubleshooting. This behavior occurs regardless of whether it was an image copy created before TSPITR or restored by RMAN during TSPITR.

Performing TSPITR with CONFIGURE AUXNAME and Image Copies: Scenario

Assume that you have enough disk space to save image copies of your entire database for use in TSPITR. In preparation for the possibility of TSPITR, you do the following:

- Configure an AUXNAME for each datafile in your database by using a command of the following form:

```
CONFIGURE AUXNAME FOR DATAFILE n TO auxname_n;
```

- Take an image copy of the database every Sunday by using a command of the following form:

  ```
  BACKUP AS COPY DATAFILE n FORMAT auxname_n
  ```

  If the image copies are all in the same location on disk, and if they are named similarly to the original datafiles, then you can avoid performing backups of every datafile. Instead, you can use the FORMAT or DB_FILE_NAME_CONVERT options of the BACKUP command and use BACKUP AS COPY DATABASE. For example, if the configured auxiliary names are a translation of the location maindisk to auxdisk, then you could use the following command:

  ```
  BACKUP AS COPY
    DATABASE
    DB_FILE_NAME_CONVERT (maindisk, auxdisk);
  ```

  Note: Because Oracle-managed filenames cannot generally be translated using a simple substitution, you cannot typically use the technique of using DB_FILE_NAME_CONVERT to generate names for image copies stored in OMF.

You are then prepared for TSPITR without restoring from backup. For example, if an erroneous batch job, started on November 15, 2007, at 19:00:00, incorrectly updates the tables in the tablespace parts, you could use the following command to perform TSPITR on tablespace parts:

```
RECOVER TABLESPACE parts UNTIL TIME 'November 15 2007, 19:00:00';
```

Because AUXNAME locations are configured and refer to datafile copies from an SCN before the TSPITR target time, the auxiliary set and recovery set datafiles are not restored from backup. Instead, the datafile copies are directly used in recovery, eliminating the restore overhead.

Note that at the end of the TSPITR, the datafiles for tablespace parts will not be located in the original datafile locations. Instead, they will be in the AUXNAME locations. If you want only to use only the AUXNAME locations for the auxiliary set, then run CONFIGURE AUXNAME . . . CLEAR for the files in the recovery set before starting TSPITR. In this case, RMAN will have to restore the datafiles.

Customizing Initialization Parameters for the Automatic Auxiliary Instance in TSPITR

The automatic auxiliary instance looks for initialization parameters in a file that is operating system-dependent. RMAN always looks for this default parameter file for the automatic auxiliary instance when performing TSPITR. If the file is not found, then RMAN does not generate an error. Instead, RMAN defines the basic initialization parameters in Table 20–3 for the automatic auxiliary instance.

Table 20–3 Default Initialization Parameters

Initialization Parameter	Value
DB_NAME	Same as DB_NAME of the source database.
COMPATIBLE	Same as the compatible setting of the target database.
DB_UNIQUE_NAME	Generated unique value based on DB_NAME.
DB_BLOCK_SIZE	Same as the DB_BLOCK_SIZE of the target database.

Table 20–3 (Cont.) Default Initialization Parameters

Initialization Parameter	Value
DB_CREATE_FILE_DEST	Auxiliary destination (only if the AUXILIARY DESTINATION argument is set). RMAN creates Oracle-managed control files and online logs in this location.
CONTROL_FILES	A generated filename in the auxiliary destination (only if the AUXILIARY DESTINATION argument is set). RMAN creates control files in this location. By default, RMAN creates one control file for the auxiliary instance in an operating system-specific location. In UNIX, the default location is ?/dbs/cntrl_@.dbf, where ? stands for ORACLE_HOME and @ stands for ORACLE_SID. For an automatic auxiliary instance, RMAN randomly generates the ORACLE_SID. **See Also:** *Oracle Database Reference* for more on the use of the CONTROL_FILES initialization parameter
LARGE_POOL_SIZE	1M
SHARED_POOL_SIZE	110M
PROCESSES	50

It is rarely necessary to alter the parameter file, especially if you provide an AUXILIARY DESTINATION argument to RECOVER TABLESPACE. If you override one of the initialization parameters in Table 20–3 with an inappropriate value, then TSPITR may fail due to problems with the auxiliary instance. Nevertheless, you can add other parameters besides these basic parameters if needed. For example, you can use DB_FILE_NAME_CONVERT to specify the names of the datafiles in the auxiliary set.

Another way to specify parameters for the automatic auxiliary instance is to place the initialization parameters in a file. You can then provide the location of this file with the SET AUXILIARY INSTANCE PARAMETER FILE command before executing TSPITR. Note that the path specified when using SET AUXILIARY INSTANCE PARAMETER is a path on the system running the RMAN client, not the target or auxiliary instances.

Specifying the Auxiliary Instance Control File Location in TSPITR

If you use an initialization parameter file, then you can specify your own location for the control file of your auxiliary instance. Set the CONTROL_FILES initialization parameter to specify a location for the control files.

If you do not explicitly specify a control file location, and if you use the AUXILIARY DESTINATION parameter, then RMAN locates the control file in the auxiliary destination. If you do not use AUXILIARY DESTINATION, then the auxiliary instance control files are stored in an operating system-specific location.

No matter where you store your auxiliary instance control file, it is removed at the end of a successful TSPITR operation. Because control files are relatively small, it is rare that RMAN will encounter a problem creating an auxiliary control file. If there is not enough space to create the control file, however, then TSPITR will fail.

Specifying the Auxiliary Instance Online Log Location in TSPITR

If you specify the LOG_FILE_NAME_CONVERT initialization parameter in your auxiliary instance parameter file, then this parameter determines the online redo log location. Otherwise, if RMAN is using an auxiliary destination and managing the auxiliary instance for you, it creates the online redo log in the auxiliary destination.

If you do not specify a location for the online redo logs using `LOG_FILE_NAME_CONVERT` or `AUXILIARY DESTINATION`, then TSPITR will fail trying to create the online redo logs. Even if `DB_CREATE_FILE_DEST` or `LOG_FILE_CREATE_DEST` are specified in the initialization parameter file, in TSPITR they do not control the creation of the online redo logs of the auxiliary instance.

Performing RMAN TSPITR Using Your Own Auxiliary Instance

Typically, you should allow RMAN to manage the creation and destruction of the auxiliary instance used during RMAN TSPITR. Nevertheless, creating and using your own auxiliary instance is also supported.

One reason you might want to create your own instance is to exercise control of channels used in TSPITR. The automatic auxiliary instance uses the configured channels of the target database as the basis for the channels to configure on the auxiliary instance and use during restore. You may need different channel settings and may not want to use `CONFIGURE` to change the settings on the target database. In this case, you can operate your own auxiliary instance.

Preparing Your Own Auxiliary Instance for RMAN TSPITR

Creating an Oracle instance suitable for use as an auxiliary instance requires that you carry out all of the following steps:

- Step 1: Create an Oracle Password File for the Auxiliary Instance
- Step 2: Create an Initialization Parameter File for the Auxiliary Instance
- Step 3: Check Oracle Net Connectivity to the Auxiliary Instance

Step 1: Create an Oracle Password File for the Auxiliary Instance

For instructions on how to create and maintain Oracle password files, refer to *Oracle Database Administrator's Guide*.

Step 2: Create an Initialization Parameter File for the Auxiliary Instance

Use a text editor to create an initialization parameter file for the auxiliary instance on the target database host. For this example, assume your parameter file is placed at `/tmp/initAux.ora`. Set the parameters described in Table 20–4.

Note: For TSPITR, the target and auxiliary database instances must be on the same host.

Table 20–4 Initialization Parameters in the Auxiliary Instance

Parameter	Mandatory?	Value
`DB_NAME`	YES	The same name as the target database.
`DB_UNIQUE_NAME`	YES	A value different from any database in the same Oracle home. For simplicity, specify *_dbname*. For example, if the target database name is `trgt`, then specify `_trgt`.
`REMOTE_LOGIN_PASSWORDFILE`	YES	Set to `EXCLUSIVE` when connecting to the auxiliary instance by means of a password file. Otherwise, set to `NONE`.
`COMPATIBLE`	YES	The same value as the parameter in the target database.

Table 20–4 (Cont.) Initialization Parameters in the Auxiliary Instance

Parameter	Mandatory?	Value
DB_BLOCK_SIZE	YES	If this initialization parameter is set in the target database, then it must be set to the same value in the auxiliary instance.
LOG_FILE_NAME_CONVERT	YES	Patterns to generate filenames for the online redo logs of the auxiliary database based on the online redo log names of the target database. Query V$LOGFILE.MEMBER, to obtain target instance online redo log filenames, and ensure that the conversion pattern matches the format of the filename shown in the view. This parameter is the only way to name the online redo logs for the auxiliary instance. Without it, TSPITR will fail when trying to open the auxiliary instance because the online logs cannot be created. **Note:** Some platforms do not support ending patterns in a forward or backward slash (\ or /). **See Also:** "Specifying the Auxiliary Instance Online Log Location in TSPITR" on page 20-17 for restrictions on possible values for LOG_FILE_NAME_CONVERT with OMF filenames
DB_FILE_NAME_CONVERT	NO	Patterns to convert filenames for the datafiles of the auxiliary database. You can use this parameter to generate filenames for those files that you did not name with SET NEWNAME or CONFIGURE AUXNAME. Obtain the datafile filenames by querying V$DATAFILE.NAME, and ensure that the conversion pattern matches the format of the filename displayed in the view. You can also specify this parameter on the RECOVER command itself. **Note:** Some platforms do not support ending patterns in a forward or backward slash (\ or /). **See Also:** "Using DB_FILE_NAME_CONVERT to Name Auxiliary Set Datafiles" on page 20-12
CONTROL_FILES	NO	Filenames that do not conflict with the control file names of the target instance (or any other existing file).

Set other parameters as needed, including the parameters that allow you to connect as SYSDBA through Oracle Net.

The following example shows possible initialization parameter settings for an auxiliary instance for TSPITR:

```
DB_NAME=trgt
DB_UNIQUE_NAME=_trgt
CONTROL_FILES=/tmp/control01.ctl
DB_FILE_NAME_CONVERT=('/oracle/oradata/trgt/','/tmp/')
LOG_FILE_NAME_CONVERT=('/oracle/oradata/trgt/redo','/tmp/redo')
REMOTE_LOGIN_PASSWORDFILE=exclusive
COMPATIBLE =11.0.0
DB_BLOCK_SIZE=8192
```

Note: After setting these initialization parameters, ensure that you do not overwrite the initialization settings for the production files at the target database.

See Also: *Oracle Database Net Services Administrator's Guide* for more information about Oracle Net

Step 3: Check Oracle Net Connectivity to the Auxiliary Instance

The auxiliary instance must have a valid net service name. Before proceeding, use SQL*Plus to ensure that you can establish a SYSDBA connection to the auxiliary instance.

Preparing RMAN Commands for TSPITR with Your Own Auxiliary Instance

If you are running your own auxiliary instance, then it is possible for the sequence of commands required for TSPITR to be long. This situation can occur when you allocate a complex channel configuration for restoring from backup and you are not using DB_FILE_NAME_CONVERT to determine file naming.

You may wish to store the series of commands for TSPITR in an RMAN command file. Review the command file carefully to catch any errors. To read the command file into RMAN, use the @ command (or the CMDFILE command line argument when starting RMAN).

The following example runs the command file named /tmp/tspitr.rman:

```
@/tmp/tspitr.rman;
```

The results will be the same as in the previous example.

> **See Also:** "Using Command Files with RMAN" on page 4-3

Planning Channels for TSPITR with Your Own Auxiliary Instance

When you run your own auxiliary instance, the default behavior is to use the automatic channel configuration of the target instance. If you decide to allocate your own channel configuration, however, then you can include ALLOCATE AUXILIARY CHANNEL commands in a RUN block along with the RECOVER TABLESPACE command for TSPITR. Plan out these commands, if necessary, and add them to the sequence of commands you will run for TSPITR.

> **See Also:** "Performing TSPITR with Your Own Auxiliary Instance: Scenario" on page 20-21 to learn how to include channel allocation in your TSPITR script

Planning Datafile Names with Your Own Auxiliary Instance: SET NEWNAME

You may wish to use SET NEWNAME commands to refer to existing image copies of auxiliary set files to improve TSPITR performance, or to assign new names to the recovery set files for after TSPITR. Plan out these commands, if necessary, and add them to the sequence of commands you will run for TSPITR.

Executing TSPITR with Your Own Auxiliary Instance

With the preparations complete and your TSPITR commands completely planned, you are now ready to perform TSPITR. The following steps are required:

- Step 1: Start the Auxiliary Instance in NOMOUNT Mode
- Step 2: Connect the RMAN Client to Target and Auxiliary Instances
- Step 3: Execute the RECOVER TABLESPACE Command

Step 1: Start the Auxiliary Instance in NOMOUNT Mode

Before beginning RMAN TSPITR, start SQL*Plus and connect to the auxiliary instance with SYSOPER privileges.

Start the auxiliary instance in NOMOUNT mode, specifying a parameter file if necessary. For example, enter the following SQL*Plus command:

```
SQL> STARTUP NOMOUNT PFILE='/tmp/initAux.ora'
```

Remember that if you specify PFILE, then the path for the PFILE will be a client-side path on the host from which you run SQL*Plus.

Because the auxiliary instance does not yet have a control file, you can only start the instance in NOMOUNT mode. Do not create a control file or try to mount or open the auxiliary instance for TSPITR.

Step 2: Connect the RMAN Client to Target and Auxiliary Instances

Start RMAN and connect to the target database and the manually created auxiliary instance.

Step 3: Execute the RECOVER TABLESPACE Command

In the simplest case, execute the RECOVER TABLESPACE... UNTIL command at the RMAN prompt:

```
RECOVER TABLESPACE ts1, ts2... UNTIL TIME 'time';
```

If you want to use the ALLOCATE AUXILIARY CHANNEL or SET NEWNAME commands, then include these commands before the RECOVER TABLESPACE command within a RUN command. The following example illustrates this technique:

```
RUN
{
   ALLOCATE AUXILIARY CHANNEL c1 DEVICE TYPE DISK;
   ALLOCATE AUXILIARY CHANNEL c2 DEVICE TYPE sbt;
  # and so on...
   RECOVER TABLESPACE ts1, ts2 UNTIL TIME 'time';
}
```

Performing TSPITR with Your Own Auxiliary Instance: Scenario

This scenario shows the execution of a RECOVER TABLESPACE... UNTIL operation. This scenario illustrates the following features of RMAN TSPITR:

- Managing your own auxiliary instance
- Configuring channels for restore of backups from disk and **SBT** devices
- Using recoverable image copies for some auxiliary set datafiles using SET NEWNAME
- Specifying new names for recovery set datafiles using SET NEWNAME

Assume that you follow these steps:

1. Prepare the auxiliary instance as described in "Preparing Your Own Auxiliary Instance for RMAN TSPITR" on page 20-18. Specify a password for the auxiliary instance in the password file, and set up the auxiliary instance parameter file /bigtmp/init_tspitr_prod.ora with the following settings:

   ```
   DB_NAME=PROD
   DB_UNIQUE_NAME=tspitr_PROD
   CONTROL_FILES=/bigtmp/tspitr_cntrl.dbf'
   DB_FILE_NAME_CONVERT=('?/oradata/prod', '/bigtmp')
   LOG_FILE_NAME_CONVERT=('?/oradata/prod', '/bigtmp')
   COMPATIBLE=11.0.0
   ```

```
BLOCK_SIZE=8192
REMOTE_LOGIN_PASSWORD=exclusive
```

2. Create service name `pitprod` for the auxiliary instance, and check for connectivity.
3. Using SQL*Plus, connect to the auxiliary instance with `SYSOPER` privileges. Start the instance in `NOMOUNT` mode:

```
SQL> STARTUP NOMOUNT PFILE=/bigtmp/init_tspitr_prod.ora
```

4. Start RMAN and connect to the target and auxiliary database instances.
5. Enter the following commands in a `RUN` block to set up and execute TSPITR:

```
RUN
{
# Specify NEWNAMES for recovery set datafiles
  SET NEWNAME FOR DATAFILE '?/oradata/prod/clients01.dbf'
                        TO '?/oradata/prod/clients01_rec.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/prod/clients02.dbf'
                        TO '?/oradata/prod/clients02_rec.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/prod/clients03.dbf'
                        TO '?/oradata/prod/clients03_rec.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/prod/clients04.dbf'
                        TO '?/oradata/prod/clients04_rec.dbf';

# Specified newnames for some of the auxiliary set
# datafiles that have a valid image copy to avoid restores:
  SET NEWNAME FOR DATAFILE '?/oradata/prod/system01.dbf'
                        TO '/backups/prod/system01_monday_noon.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/prod/system02.dbf'
                        TO '/backups/prod/system02_monday_noon.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/prod/sysaux01.dbf'
                        TO '/backups/prod/sysaux01_monday_noon.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/prod/undo01.dbf'
                        TO '/backups/prod/undo01_monday_noon.dbf';

# Specified the types of channels to use
  ALLOCATE AUXILIARY CHANNEL c1 DEVICE TYPE DISK;
  ALLOCATE AUXILIARY CHANNEL t1 DEVICE TYPE sbt;

# Recovered the clients tablespace to 24 hours ago:
  RECOVER TABLESPACE clients UNTIL TIME 'sysdate-1';
}
```

Consider storing this command sequence in a command file and executing the command file.

If the TSPITR operation is successful, then the results are:

- The recovery set datafiles are registered in the target database control file under the names specified with `SET NEWNAME`, with their contents as of the time specified time for the TSPITR.
- The auxiliary files are removed by RMAN, including the control files, online logs and auxiliary set datafiles of the auxiliary instance.
- The auxiliary instance is shut down.

If the TSPITR operation fails, then the auxiliary files are left on disk for troubleshooting purposes. If RMAN created the auxiliary instance, it is shut down; otherwise, it is left in whatever state it was in when the TSPITR operation failed.

Troubleshooting RMAN TSPITR

A variety of problems can cause TSPITR to fail before the process is complete.

- Name conflicts can occur between files already in the target database, filenames assigned by the `SET NEWNAME` or `CONFIGURE AUXNAME` commands, and filenames generated by the effect of the `DB_FILE_NAME_CONVERT` parameter.
- When RMAN exports the metadata about recovered objects from the auxiliary instance, it uses space in the temporary tablespace for sorting. If there is insufficient space in the temporary tablespace for the sorting operation, then you need to increase the amount of sort space available.

Troubleshooting Filename Conflicts

Suppose that `SET NEWNAME`, `CONFIGURE AUXNAME`, and `DB_FILE_NAME_CONVERT` cause multiple files in the auxiliary or recovery sets to have the same name. In this case, RMAN reports an error during TSPITR. To correct the problem, use different values for these parameters to eliminate the duplicate name.

Troubleshooting Identification of Tablespaces with Undo Segments

During TSPITR, RMAN needs information about which tablespaces had undo segments at the TSPITR target time. This information is usually available in the recovery catalog, if one is used.

Assume a case in which there is no recovery catalog or the information is not found in the recovery catalog. In this case, RMAN assumes that the set of tablespaces with undo segments at the target time is the same as the set of tablespaces with undo segments at the present time. If this assumption is not correct, then TSPITR will fail with an error. In this case, use the `UNDO TABLESPACE` clause to provide a list of tablespaces with undo segments at the target time.

Troubleshooting the Restart of a Manual Auxiliary Instance After TSPITR Failure

Assume that you are managing your own auxiliary instance when there is a failure in TSPITR. Before you can try TSPITR again, you must shut down the auxiliary instance, correct the problem which interfered with TSPITR, and then start the auxiliary instance `NOMOUNT` before trying TSPITR again.

Part VI

Tuning and Troubleshooting

The following chapters describe how to tune and troubleshoot RMAN operations. This part of the book contains these chapters:

- Chapter 21, "Tuning RMAN Performance"
- Chapter 22, "Troubleshooting RMAN Operations"

21

Tuning RMAN Performance

This chapter contains the following sections:

- Purpose of RMAN Performance Tuning
- Basic Concepts of RMAN Performance Tuning
- Using V$ Views to Diagnose RMAN Performance Problems
- Tuning RMAN Backup Performance

Purpose of RMAN Performance Tuning

An RMAN backup or restore job can be divided into separate phases or components. The slowest of these phases in any RMAN job is called the **bottleneck**. The purpose of RMAN tuning is to identify the bottlenecks for a given job and use RMAN commands, initialization parameters, or adjustments to physical media to improve performance.

Basic Concepts of RMAN Performance Tuning

Tuning RMAN performance requires a detailed understanding of how RMAN creates a backup. As explained in "RMAN Channels" on page 3-3, the work of a backup is performed by one or more channels. A **channel** represents a stream of bytes to a storage device.

For the purposes of illustration, you can think of the byte stream as passing from the input buffers in memory through the CPU to the output buffers, and from there to the storage device. To direct a backup to two tape devices, you allocate two tape channels so that each byte stream goes to a different device.

The work of each channel, whether of type disk or SBT, is subdivided into the following distinct phases:

1. Read Phase

   A channel reads blocks from disk into input I/O buffers.

2. Copy Phase

   A channel copies blocks from input buffers to output buffers and performs additional processing on the blocks.

3. Write Phase

   A channel writes the blocks from output buffers to storage media. The write phase can take either of the following mutually exclusive forms, depending on the type of backup media:

- Write Phase for SBT
- Write Phase for Disk

Figure 21–1 depicts two channels backing up data stored on three disks. Each channel reads the data into the input buffers, processes the data while copying it from the input to the output buffers, and the writes the data from the output buffers to disk.

Figure 21–1 Phases of a Multichannel Backup to Disk

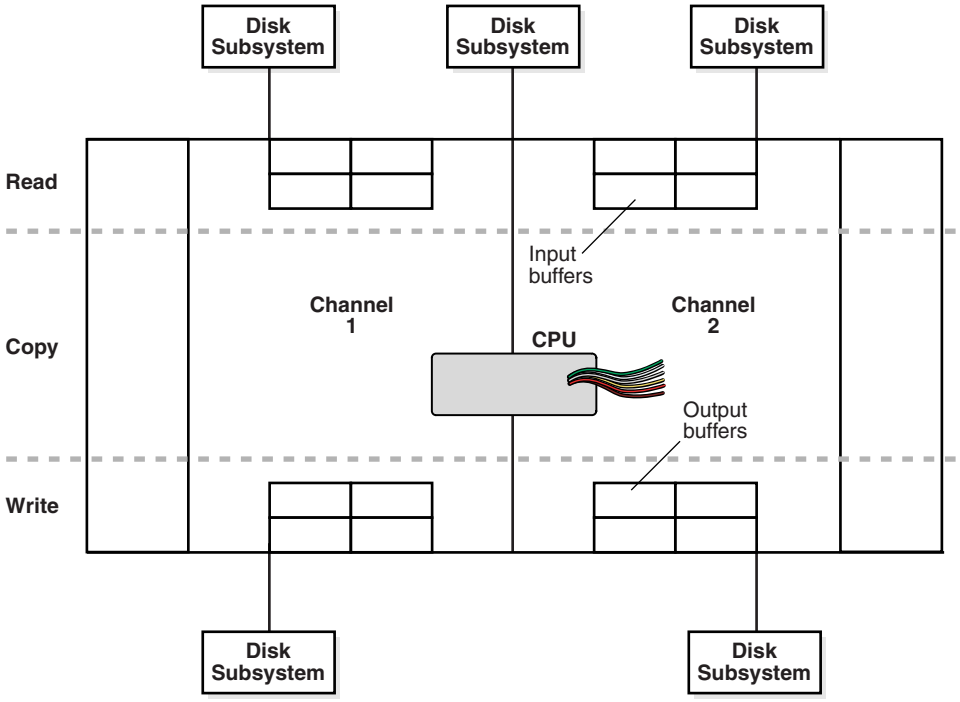
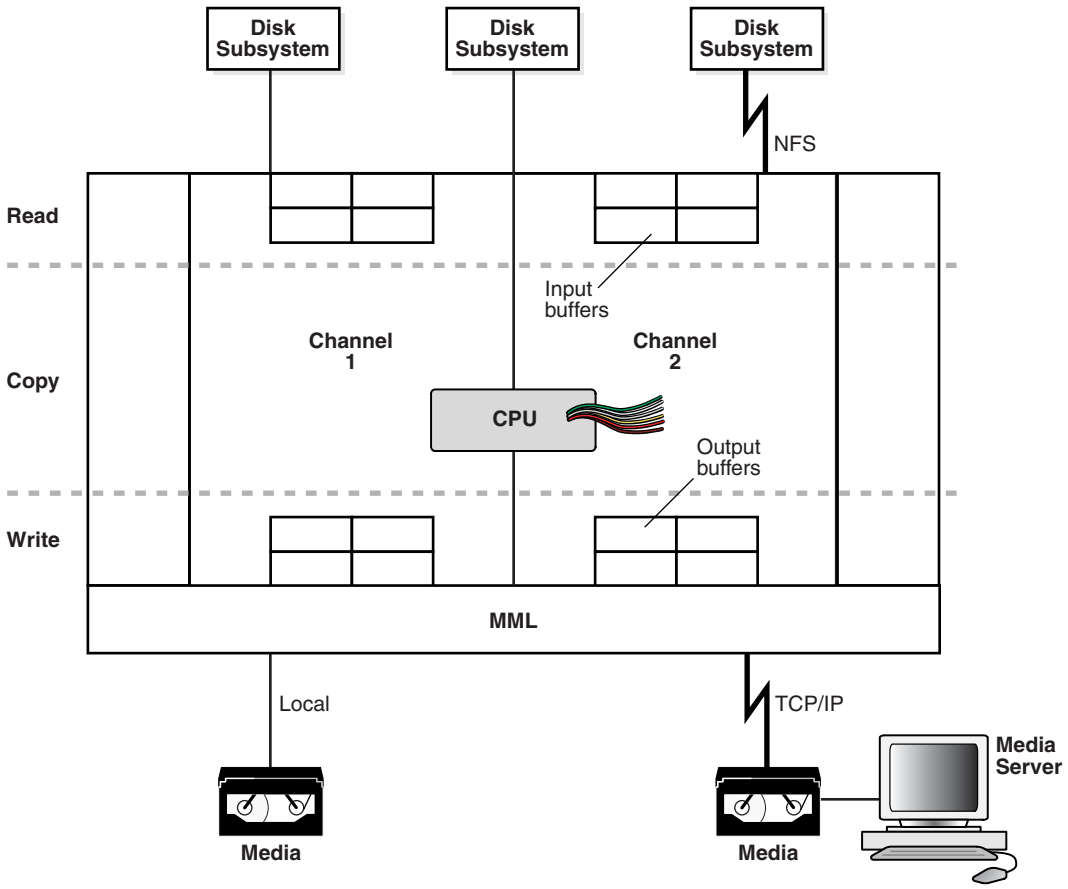


Figure 21–2 also depicts two channels backing up data stored on three disks, but one of the disks is mounted remotely over the network. Each channel reads the data into the input buffers, processes the data while copying it from the input to the output buffers, and the writes the data from the output buffers to tape. Channel 1 writes the data to a locally-attached tape drive, whereas channel 2 sends the data over the network to a remote media server.

Figure 21–2 Phases of a Multichannel Backup to Tape



When restoring data, a channel performs these steps in reverse order and reverses the reading and writing operations. The following sections explain RMAN tuning concepts in terms of a backup.

Read Phase

This section explains factors that affect performance when an RMAN channel is reading data from disk:

- Allocation of Input Disk Buffers
- Synchronous and Asynchronous Disk I/O
- Disk I/O Slaves
- RATE Channel Parameter

Allocation of Input Disk Buffers

During a backup, an RMAN channel reads the blocks from the input files into I/O disk buffers. The database files on the disk subsystem can be managed by either **Automatic Storage Management (ASM)** or an alternative volume manager or file system. The considerations for backup tuning change depending on whether you manage database files with ASM.

The allocation of the input buffers depends on how the files are multiplexed. Backup **multiplexing** is RMAN's ability to read a number of files in a backup simultaneously from different sources and then write them to a single backup piece. The **level of multiplexing**, which is the number of input files simultaneously read and then written into the same **backup piece**, is determined by the algorithm described in "Multiplexed Backup Sets" on page 7-6. Review this section before proceeding.

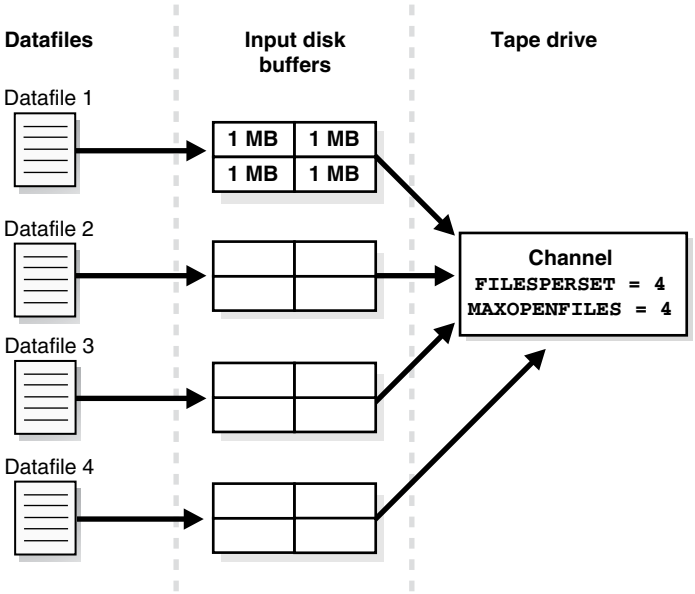
When an RMAN channel backs up files from disk, it uses the rules described in the following table to determine how large to make the input disk buffers.

Table 21–1 Datafile Read Buffer Sizing Algorithm

Level of Multiplexing	Input Disk Buffer Size
Less than or equal to 4	The RMAN channel allocates 16 buffers of size 1 MB so that the total buffer size for all the input files is 16 MB.
Greater than 4 but less than or equal to 8	The RMAN channel allocates a variable number of disk buffers of size 512 KB so that the total buffer size for all the input files is less than 16 MB.
Greater than 8	The RMAN channel allocates 4 disk buffers of 128 KB for each file, so that the total buffer size for each input file is 512 KB.

In the example shown in Figure 21–3, one channel is backing up four datafiles. `MAXOPENFILES` is set to 4 and `FILESPERSET` is set to 4. Thus, the level of multiplexing is 4. So, the total size of the buffers for each datafile is 4 MB. The combined size of all the buffers is 16 MB.

Figure 21–3 Disk Buffer Allocation




```
SGA if BACKUP_DISK_IO_SLAVES ≠ 0
                or
PGA if BACKUP_DISK_IO_SLAVES = 0
```

If a channel is backing up files stored in ASM, then the number of input disk buffers equals the number of physical disks in the ASM disk group. For example, if a datafile is stored in an ASM disk group that contains 16 physical disks, then the channel allocates 16 input buffers for the datafile backup.

If a channel is restoring a backup from disk, then 4 buffers are allocated. The size of the buffers is dependent on the operating system.

Synchronous and Asynchronous Disk I/O

When a channel reads from or writes to disk, the I/O is either **synchronous I/O** or **asynchronous I/O**. When the disk I/O is synchronous, a server process can perform only one task at a time. When the disk I/O is asynchronous, a server process can begin an I/O and then perform other work while waiting for the I/O to complete. RMAN can also begin multiple I/O operations before waiting for the first to complete.

When reading from an ASM **disk group**, you should use asynchronous disk I/O if possible. Also, if a channel reads from a **raw device** managed with a volume manager, then asynchronous disk I/O also works well. Some operating systems support native asynchronous disk I/O. The database takes advantage of this feature if it is available.

Disk I/O Slaves

On operating systems that do not support native asynchronous I/O, the database can simulate it with special I/O slave processes. These processes are dedicated to performing I/O on behalf of another process.

You can control disk I/O slaves by setting the `DBWR_IO_SLAVES` initialization parameter, which is not dynamic. The parameter specifies the number of I/O server processes used by the database writer process (DBWR). By default, the value is 0 and I/O server processes are not used. If asynchronous I/O is disabled, then RMAN allocates four backup disk I/O slaves for any nonzero value of `DBWR_IO_SLAVES`.

When attempting to get shared buffers for I/O slaves, the database does the following:

- If `LARGE_POOL_SIZE` is set, and if the `DBWR_IO_SLAVES` parameter is set to a nonzero value, then the database attempts to get memory from the large pool. If this value is not large enough, then an error is recorded in the alert log, the database does not try to get buffers from the shared pool, and asynchronous I/O is not used.
- If `LARGE_POOL_SIZE` is not set or is set to zero, then the database attempts to get memory from the shared pool.
- If the database cannot get enough memory, then it obtains I/O buffer memory from the PGA and writes a message to the `alert.log` file indicating that synchronous I/O is used for this backup.

The memory from the large pool is used for many features, including the shared server, parallel query, and RMAN I/O slave buffers. Configuring the large pool prevents RMAN from competing with other subsystems for the same memory.

Requests for contiguous memory allocations from the shared pool are usually small (under 5 KB) in size. However, it is possible that a request for a large contiguous memory allocation can either fail or require significant memory housekeeping to release the required amount of contiguous memory. Although the shared pool may be unable to satisfy this memory request, the large pool is able to do so. The large pool does not have a least recently used (LRU) list; the database does not attempt to age memory out of the large pool.

RATE Channel Parameter

In the `ALLOCATE` or `CONFIGURE CHANNEL` commands, the `RATE` parameter specifies the bytes/second that are read on a channel. You can use this parameter to set an upper limit for bytes read so that RMAN does not consume excessive disk bandwidth and degrade online performance. Essentially, `RATE` serves as a backup throttle. For

example, if you set `RATE 1500K`, and if each disk drive delivers 3 MB/second, then the channel leaves some disk bandwidth available to the online system.

Copy Phase

In this phase, a channel copies blocks from the input to the output buffers and performs additional processing. For example, if a channel reads data from disk and backs up to tape, then the channel copies the data from the disk buffers to the output tape buffers.

The copy phase involves the following types of processing:

- Validation
- Compression
- Encryption

When performing **validation** of the blocks, RMAN checks them for corruption. Validation is explained in Chapter 15, "Validating Database Files and Backups". Typically, this processing is not CPU-intensive.

When performing **binary compression**, RMAN applies a compression algorithm to the data in backup sets. Binary compression can be CPU-intensive. You can choose which compression algorithm that RMAN uses for backups. By default, RMAN uses `BZIP2`, which has a very good compression ratio. `ZLIB` compression, which requires a `COMPATIBLE` setting of 11.0.0 or higher, is very fast but has a lower compression ratio than other algorithms. Binary compression is explained in "Making Compressed Backups" on page 8-6.

When performing **backup encryption**, RMAN encrypts backup sets by using one of the algorithms listed in `V$RMAN_ENCRYPTION_ALGORITHMS`. RMAN offers three modes of encryption: transparent, password-protected, and dual-mode. Backup encryption is explained in "Encrypting RMAN Backups" on page 9-10. Backup encryption can be CPU-intensive.

Write Phase for SBT

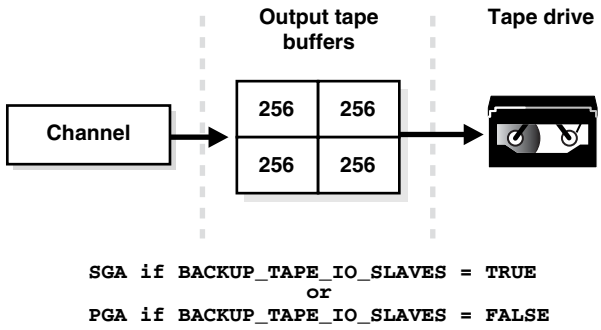
When backing up to SBT, RMAN gives the **media manager** a stream of bytes and associates a unique name with this stream. All details of how and where that stream is stored are handled entirely by the media manager. Thus, a backup to tape involves the interaction of both RMAN and the media manager.

RMAN Component of Write Phase for SBT

The RMAN-specific factors affecting the SBT write phase are analogous to the factors affecting disk reads. In both cases, the buffer allocation, slave processes, and synchronous or asynchronous I/O affect performance.

Allocation of Tape Buffers If you back up to or restore from an **SBT** device, then by default the database allocates four buffers for each channel for the tape writers (or reads if restoring data). The size of the tape I/O buffers is platform-dependent. You can change this value with the `PARMS` and `BLKSIZE` parameters of the `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` command.

Figure 21–4 Allocation of Tape Buffers



Tape I/O Slaves RMAN allocates the tape buffers in the SGA or the PGA, depending on whether I/O slaves are used. If you set the initialization parameter `BACKUP_TAPE_IO_SLAVES=true`, then RMAN allocates tape buffers from the SGA. Tape devices can only be accessed by one process at a time, so RMAN starts as many slaves as necessary for the number of tape devices. If the `LARGE_POOL_SIZE` initialization parameter is also set, then RMAN allocates buffers from the large pool. If you set `BACKUP_TAPE_IO_SLAVES=false`, then RMAN allocates the buffers from the PGA.

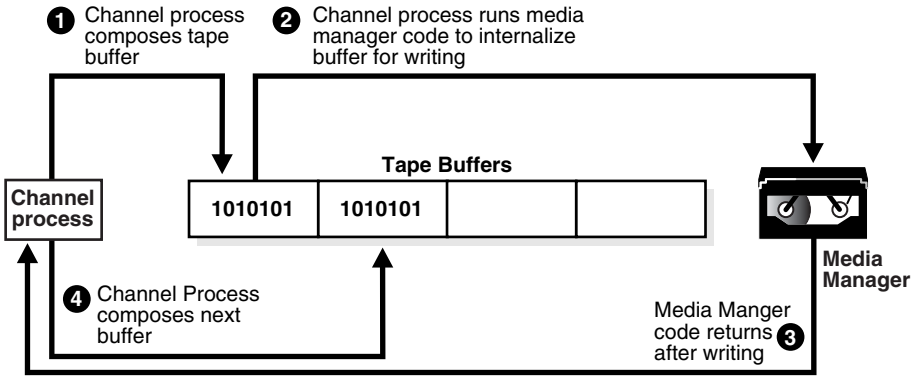
If you use I/O slaves, then set the `LARGE_POOL_SIZE` initialization parameter to dedicate SGA memory to holding these large memory allocations. This parameter prevents RMAN I/O buffers from competing with the library cache for SGA memory. If I/O slaves for tape I/O were requested but there is not enough space in the SGA for them, slaves are not used, and a message appears in the alert log.

Note that `BACKUP_TAPE_IO_SLAVES` specifies whether RMAN uses slave processes, not the number of slave processes. Tape devices can only be accessed by one process at a time, and RMAN uses the number of slaves necessary for the number of tape devices.

Synchronous and Asynchronous I/O When an SBT channel reads or writes data to tape, the I/O is always synchronous. For tape I/O, each channel allocated (whether manually or automatically) corresponds to a server process, called here a channel process.

The following figure shows synchronous I/O in a backup to tape.

Figure 21–5 Synchronous Tape I/O



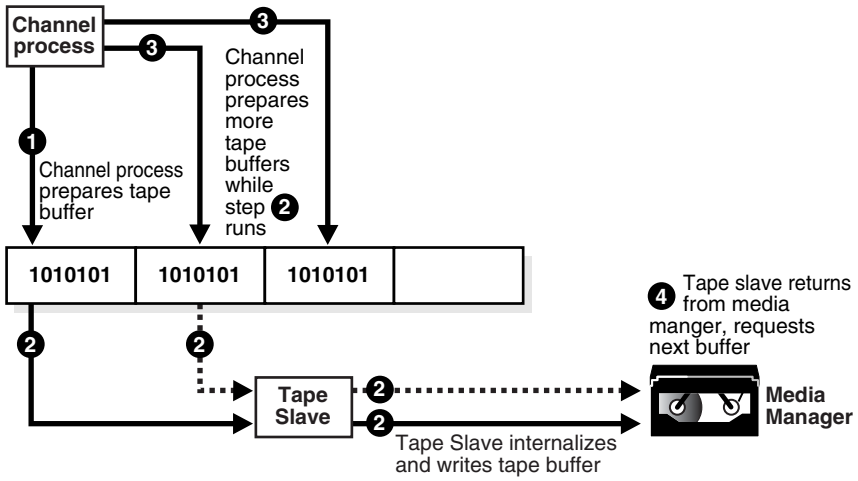
The following steps occur:

1. The channel process composes a tape buffer.

2. The channel process executes media manager code that processes the tape buffer and internalizes it for further processing and storage by the media manager.
3. The media manager code returns a message to the server process stating that it has completed writing.
4. The channel process can initiate a new task.

The following figure shows asynchronous I/O in a tape backup. Asynchronous I/O to tape is simulated by using tape slaves. In this case, each allocated channel corresponds to a server process, which in the explanation which follows is identified as a channel process. For each channel process, one tape slave is started (or more than one, in the case of multiple copies).

Figure 21–6 Asynchronous Tape I/O



The following steps occur:

1. A channel process writes blocks to a tape buffer.
2. The channel process sends a message to the tape slave process to process the tape buffer. The tape slave process executes media manager code that processes the tape buffer and internalizes it so that the media manager can process it.
3. While the tape slave process is writing, the channel process is free to read data from the datafiles and prepare more output buffers.
4. After the tape slave channel returns from the media manager code, it requests a new tape buffer, which usually is ready. Thus waiting time for the channel process is reduced, and the backup is completed faster.

Media Manager Component of Write Phase for SBT

The following factors affect the speed of the backup to tape:

- Network Throughput
- Native Transfer Rate
- Tape Compression
- Tape Streaming
- Physical Tape Block Size

Network Throughput If the tape device is remote, then the media manager needs to transfer data over the network. For example, an administrative domain in Oracle Secure Backup can contain multiple networked client hosts, media servers, and tape devices. If the database is on one host, but the output tape drive is attached to a different host, then Oracle Secure Backup manages the data transfer over the network. The network throughput is the upper limit for backup performance.

Native Transfer Rate The tape **native transfer rate** is the speed of writing to a tape without compression. This speed represents the upper limit of the backup rate. The upper limit of your backup performance should be the aggregate transfer rate of all of your tape drives. If your backup is already performing at that rate, and if it is not using an excessive amount of CPU, then RMAN performance tuning will not help.

Tape Compression The level of tape compression is very important for backup performance. If the tape has good compression, then the sustained backup rate is faster. For example, if the compression ratio is 2:1 and native transfer rate of the tape drive is 6 MB/s, then the resulting backup speed is 12 MB/s. In this case, RMAN must be able to read disks with a throughput of more than 12 MB/s or the disk becomes the bottleneck for the backup.

> **Note:** You should not use both tape compression provided by the media manager and binary compression provided by RMAN. If the media manager compression is efficient, then it is usually the better choice. Using RMAN-compressed backup sets can be an effective alternative to reduce bandwidth used to move uncompressed backup sets over a network to the media manager, if the CPU overhead required to compress the data in RMAN is acceptable.

Tape Streaming Tape **streaming** during write operations has a major impact on tape backup performance. Almost all tape drives currently on the market are fixed-speed, streaming tape drives. Because such drives can only write data at one speed, when they run out of data to write to tape, the tape must slow down and stop. Typically, when the drive's buffer empties, the tape is moving so quickly that it actually overshoots; to continue writing, the drive must rewind the tape to locate the point where it stopped writing.

Physical Tape Block Size The physical tape block size can affect backup performance. The block size is the amount of data written by media management software to a tape in one write operation. In general, the larger the tape block size, the faster the backup. Note that physical tape block size is not controlled by RMAN or the Oracle database server, but by media management software. See your media management software's documentation for details.

Write Phase for Disk

The principal factor affecting the write phase for disk is the buffer size. When the output of the backup resides on disk, each channel allocates 4 output buffers of 1 MB each. The disk channel writes the blocks to the disk subsystem. Note that the read phase when restoring files is just like the write phase when backing up files, except the blocks move in the opposite direction.

If RMAN reads from a disk asynchronously, then it writes to the disk asynchronously. When writing to disk, you can make use of disk I/O slaves just as when reading.

If RMAN is backing up files to a disk-based output destination striped over multiple disks, then you can allocate multiple channels. The number of channels is limited only to the number of disks over which the destination is striped. ASM is one example a destination striped over multiple disks.

Using V$ Views to Diagnose RMAN Performance Problems

Typically, you begin the tuning process by using V$ views to determine where RMAN backup and restore operations are encountering problems.

Monitoring RMAN Job Progress with V$SESSION_LONGOPS

You can monitor the progress of backups and restore jobs by querying the view V$SESSION_LONGOPS. RMAN uses two types of rows in V$SESSION_LONGOPS: detail and aggregate rows.

Detail rows describe the files being processed by one job step, while aggregate rows describe the files processed by all job steps in an RMAN command. A job step is the creation or restore of one backup set or datafile copy. Detail rows are updated with every buffer that is read or written during the backup step, so their granularity of update is small. Aggregate rows are updated when each job step completes, so their granularity of update is large.

Table 21–2 describes the columns in V$SESSION_LONGOPS that are most relevant for RMAN. Typically, you will view the detail rows rather than the aggregate rows to determine the progress of each backup set.

Table 21–2 Columns of V$SESSION_LONGOPS Relevant for RMAN

Column	Description for Detail Rows
SID	The server session ID corresponding to an RMAN channel.
SERIAL#	The server session serial number. This value changes each time a server session is reused.
OPNAME	A text description of the row. Examples of details rows include RMAN: datafile copy, RMAN: full datafile backup, and RMAN: full datafile restore. **Note:** RMAN: aggregate input and RMAN: aggregate output are the only aggregate rows.
CONTEXT	For backup output rows, this value is 2. For all other rows except proxy copy (which does not update this column), the value is 1.
SOFAR	The meaning of this column depends on the type of operation described by this row: ■ For image copies, the number of blocks that have been read. ■ For backup input rows, the number of blocks that have been read from the files being backed up. ■ For backup output rows, the number of blocks that have been written to the backup piece. ■ For restores, the number of blocks that have been processed to the files that are being restored in this one job step. ■ For proxy copies, the number of files that have been copied.

Table 21–2 (Cont.) Columns of V$SESSION_LONGOPS Relevant for RMAN

Column	Description for Detail Rows
TOTALWORK	The meaning of this column depends on the type of operation described by this row: ■ For image copies, the total number of blocks in the file. ■ For backup input rows, the total number of blocks to be read from all files processed in this job step. ■ For backup output rows, the value is 0 because RMAN does not know how many blocks that it will write into any backup piece. ■ For restores, the total number of blocks in all files restored in this job step. ■ For proxy copies, the total number of files to be copied in this job step.

Each server session performing a backup or restore job reports its progress compared to the total work required for a job step. For example, if you restore the database with two channels, and each channel has two backup sets to restore (a total of four sets), then each server session reports its progress through a single backup set. When a set is completely restored, RMAN begins reporting progress on the next set to restore.

To monitor RMAN job progress:

1. Before starting the RMAN job, create a script file (called, for this example, `longops`) containing the following SQL statement:

```
SELECT SID, SERIAL#, CONTEXT, SOFAR, TOTALWORK,
       ROUND(SOFAR/TOTALWORK*100,2) "%_COMPLETE"
FROM   V$SESSION_LONGOPS
WHERE  OPNAME LIKE 'RMAN%'
AND    OPNAME NOT LIKE '%aggregate%'
AND    TOTALWORK != 0
AND    SOFAR <> TOTALWORK;
```

2. Start RMAN and connect to the target database and recovery catalog (if used).
3. Start an RMAN job. For example, enter:

```
RMAN> RESTORE DATABASE;
```

4. While the RMAN job is running, start SQL*Plus and connect to the target database, and execute the `longops` script to check the progress of the RMAN job. If you repeat the query while the RMAN job progresses, then you see output such as the following:

```
SQL> @longops
       SID    SERIAL#    CONTEXT      SOFAR  TOTALWORK %_COMPLETE
---------- ---------- ---------- ---------- ---------- ----------
         8         19          1      10377      36617      28.34

SQL> @longops
       SID    SERIAL#    CONTEXT      SOFAR  TOTALWORK % COMPLETE
---------- ---------- ---------- ---------- ---------- ----------
         8         19          1      21513      36617      58.75

SQL> @longops
       SID    SERIAL#    CONTEXT      SOFAR  TOTALWORK % COMPLETE
---------- ---------- ---------- ---------- ---------- ----------
         8         19          1      29641      36617      80.95
```

```
SQL> @longops
       SID    SERIAL#    CONTEXT      SOFAR  TOTALWORK % COMPLETE
---------- ---------- ---------- ---------- ---------- ----------
         8         19          1      35849      36617       97.9

SQL> @longops
no rows selected
```

5. If you run the `longops` script at intervals of two minutes or more and the `%_COMPLETE` column does not increase, then RMAN is encountering a problem. Refer to "Monitoring RMAN Interaction with the Media Manager" on page 22-8 to obtain more information.

If you frequently monitor the execution of long-running tasks, then you could create a shell script or batch file under your host operating system that runs SQL*Plus to execute this query repeatedly.

Identifying Bottlenecks with V$BACKUP_SYNC_IO and V$BACKUP_ASYNC_IO

You can use the `V$BACKUP_SYNC_IO` and `V$BACKUP_ASYNC_IO` views to determine the source of backup or restore bottlenecks and to see detailed progress of backup jobs.

`V$BACKUP_SYNC_IO` contains rows when the I/O is synchronous to the process (or thread on some platforms) performing the backup. `V$BACKUP_ASYNC_IO` contains rows when the I/O is asynchronous. Asynchronous I/O is obtained either with I/O processes or because it is supported by the underlying operating system.

The results of a backup or restore job remain in memory until the database instance shuts down. Thus, you can query the views after the job completes.

To determine whether the tape is streaming when the I/O is synchronous:

1. Start SQL*Plus and connect to the target database.
2. Query the `EFFECTIVE_BYTES_PER_SECOND` column in the `V$BACKUP_SYNC_IO` or `V$BACKUP_ASYNC_IO` view.

   If `EFFECTIVE_BYTES_PER_SECOND` is less than the raw capacity of the hardware, then the tape is not streaming. If `EFFECTIVE_BYTES_PER_SECOND` is greater than the raw capacity of the hardware, the tape may or may not be streaming. Compression may cause the `EFFECTIVE_BYTES_PER_SECOND` to be greater than the speed of real I/O.

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

Identifying Bottlenecks with Synchronous I/O

With synchronous I/O, it is difficult to identify specific bottlenecks because all synchronous I/O is a bottleneck to the process. The only way to tune synchronous I/O is to compare the rate (in bytes/second) with the device's maximum throughput rate. If the rate is lower than the rate that the device specifies, then consider tuning this aspect of the backup and restore process.

To determine the rate of synchronous I/O:

1. Start SQL*Plus and connect to the target database.
2. Query the `DISCRETE_BYTES_PER_SECOND` column in the `V$BACKUP_SYNC_IO` view to display the I/O rate.

If you see data in V$BACKUP_SYNC_IO, then the problem is that you have not enabled asynchronous I/O or you are not using disk I/O slaves.

Identifying Bottlenecks with Asynchronous I/O

Long waits are the number of times the backup or restore process told the operating system to wait until an I/O was complete. **Short waits** are the number of times the backup or restore process made an operating system call to poll for I/O completion in a nonblocking mode. **Ready** indicates the number of times when I/O was already ready for use, so there was no need to make an operating system call to poll for I/O completion.

To determine the rate of asynchronous I/O:

1. Start SQL*Plus and connect to the target database.
2. Query the LONG_WAITS and IO_COUNT columns in the V$BACKUP_SYNC_IO view to display the I/O rate.

   The simplest way to identify the bottleneck is to find the datafile that has the largest ratio for LONG_WAITS divided by IO_COUNT. For example, you can use the following query:

```
SELECT   LONG_WAITS/IO_COUNT, FILENAME
FROM     V$BACKUP_ASYNC_IO
WHERE    LONG_WAITS/IO_COUNT > 0
ORDER BY LONG_WAITS/IO_COUNT DESC;
```

Note: If you have synchronous I/O but you have set BACKUP_DISK_IO_SLAVES, then the I/O will be displayed in V$BACKUP_ASYNC_IO.

See Also: *Oracle Database Reference* for descriptions of the V$BACKUP_SYNC_IO and V$BACKUP_ASYNC_IO views

Tuning RMAN Backup Performance

Many factors can affect backup performance. Often, finding the solution to a slow backup is a process of trial and error. To obtain the best performance for a backup, follow the steps in this section in sequential order.

This section contains the following steps:

- Step 1: Remove the RATE Parameter from Channel Settings
- Step 2: If You Use Synchronous Disk I/O, Set DBWR_IO_SLAVES
- Step 3: If You Fail to Allocate Shared Memory, Set LARGE_POOL_SIZE
- Step 4: Tune the Read, Write, and Copy Phases

Step 1: Remove the RATE Parameter from Channel Settings

As explained in "RATE Channel Parameter" on page 21-5, the RATE parameter on a channel is intended to reduce, rather than increase, backup throughput so that more disk bandwidth is available for other database operations. If the backup is not streaming to tape, then make sure that the RATE parameter is not set.

To remove the RATE parameter:

1. Examine your backup script.
2. Do one of the following:
   - If the backup is in a `RUN` command, then remove the `RATE` parameter, if it is specified, from the `ALLOCATE` command. Skip the remaining steps.
   - If the backup is not in a `RUN` command, then start RMAN, connect to the target database, and proceed to the next step.
3. Execute the `SHOW ALL` command to show the currently configured settings.
4. Remove the `RATE` parameter, if it is set, from the `CONFIGURE CHANNEL` command.

Step 2: If You Use Synchronous Disk I/O, Set DBWR_IO_SLAVES

As explained in "Synchronous and Asynchronous Disk I/O" on page 21-5, some operating systems support native asynchronous I/O. If and only if your disk does *not* support asynchronous I/O, then set `DBWR_IO_SLAVES`. Any nonzero value for `DBWR_IO_SLAVES` causes a fixed number of disk I/O slaves to be used for backup and restore, which simulates asynchronous I/O.

To enable disk I/O slaves:

1. Start SQL*Plus and connect to the target database.
2. Shut down the database.
3. Set `DBWR_IO_SLAVES` initialization parameter to a nonzero value.

   By setting `DBWR_IO_SLAVES`, the database writer processes will use slaves. Thus, you may need to increase the value of the `PROCESSES` initialization parameter.
4. Restart the database.
5. Restart the RMAN backup.

Step 3: If You Fail to Allocate Shared Memory, Set LARGE_POOL_SIZE

Set the `LARGE_POOL_SIZE` initialization parameter if the database reports an error in the alert log stating that it does not have enough memory and that it will not start I/O slaves. The message should resemble the following:

```
ksfqxcre: failure to allocate shared memory means sync I/O will be used whenever
async I/O to file not supported natively
```

The large pool is used for RMAN and for other purposes, so its total size should accommodate all uses. This is especially true if `DBWR_IO_SLAVES` has been set and the DBWR process needs buffers.

To set the large pool size:

1. Start SQL*Plus and connect to the target database.
2. Optionally, query `V$SGASTAT.POOL` to determine which pool (shared pool or large pool) the memory for an object resides.
3. Set the `LARGE_POOL_SIZE` initialization parameter in the target database.

   You can execute an `ALTER SYSTEM SET` statement to set the parameter dynamically. The formula for setting `LARGE_POOL_SIZE` is as follows:

```
LARGE_POOL_SIZE =  number_of_allocated_channels *
                   (16 MB + ( 4 *  size_of_tape_buffer ) )
```

4. Restart the RMAN backup.

> **See Also:** *Oracle Database Concepts* for more information about the large pool, and *Oracle Database Reference* for complete information about initialization parameters

Step 4: Tune the Read, Write, and Copy Phases

There are several tasks you can perform to identify and remedy bottlenecks that affect backup performance.

Using Backup Validation To Distinguish Between Read and Write Bottlenecks

One reliable way to determine whether the output device or input disk I/O is the bottleneck in a given backup job is to compare the time required to run backup tasks with the time required to run `BACKUP VALIDATE` of the same tasks. `BACKUP VALIDATE` of a backup performs the same disk reads as a real backup but performs no I/O to an output device.

To compare backup and validation times:

1. Make sure your NLS environment date format variable is set to show the time. For example, set the NLS variables as follows:

```
setenv NLS_LANG  AMERICAN_AMERICA.WE8DEC;
setenv NLS_DATE_FORMAT "MM/DD/YYYY HH24:MI:SS"
```

2. Edit your backup script to use the `BACKUP VALIDATE` command instead of the `BACKUP` command.
3. Run the backup script.
4. Examine the RMAN output and calculate the difference between the times displayed in the `Starting backup at` and `Finished backup at` messages.
5. Edit the backup script to use the `BACKUP` command instead of the `BACKUP VALIDATE` command.
6. Run the backup script.
7. Examine the RMAN output and calculate the difference between the times displayed in the `Starting backup at` and `Finished backup at` messages.
8. Compare the backup times for the validation and real backup.

   If the time for the `BACKUP VALIDATE` to tape is about the same as the time for a real backup to tape, then reading from disk is the likely bottleneck. See "Tuning the Read Phase" on page 21-15.

   If the time for the `BACKUP VALIDATE` to tape is significantly less than the time for a real backup to tape, then writing to the output device is the likely bottleneck. See "Tuning the Copy and Write Phases" on page 21-16.

Tuning the Read Phase

RMAN may not be able to send data blocks to the output device fast enough to keep it occupied. For example, during an **incremental backup**, RMAN only backs up blocks changed since a previous datafile backup as part of the same strategy. If you do not turn on **block change tracking**, then RMAN must scan whole datafiles for changed

blocks, and fill output buffers as it finds such blocks. If few blocks changed, and if RMAN is making an SBT backup, then RMAN may not fill output buffers fast enough to keep the tape drive streaming.

You can improve backup performance by adjusting the **level of multiplexing**, which is number of input files simultaneously read and then written into the same RMAN backup piece. The level of multiplexing is the minimum of the `MAXOPENFILES` setting on the channel and the number of input files placed in each backup set. The following table makes recommendations for adjusting the level of multiplexing.

Table 21–3 Adjusting the Level of Multiplexing

ASM	Striped Disk	Recommendation
No	Yes	Increase the level of multiplexing. Determine which is the minimum, `MAXOPENFILES` or the number of files in each backup set, and then increase this value. In this way, you increase the rate at which RMAN fills tape buffers, which makes it more likely that buffers are sent to the media manager fast enough to maintain streaming.
No	No	Increase the `MAXOPENFILES` setting on the channel.
Yes	n/a	Set the `MAXOPENFILES` parameter on the channel to 1 or 2.

See Also:

- "Multiplexed Backup Sets" on page 7-6 to learn how the `MAXOPENFILES` and `FILESPERSET` settings affect the level of multiplexing
- "Incremental Backups" on page 7-13 for a conceptual overview

Tuning the Copy and Write Phases

If the read phase is performing well, then the copy or write phases are probably the bottleneck. In particular, if RMAN is sending data blocks to the tape drive fast enough to support streaming, but the tape is not streaming, then the SBT write phase is the bottleneck. Try to improve performance as follows:

- If the backup is a **full backup**, then consider using incremental backups.

  Incremental level 1 backups write only the changed blocks from datafiles to tape, so that any bottleneck on writing to tape has less impact on your overall backup strategy. In particular, if tape drives are not locally attached to the node of the database being backed up, then incremental backups can be faster. See "Making and Updating Incremental Backups" on page 8-14.

- If the backup uses the `BZIP2` compression algorithm, which is the default, then change the compression algorithm from `BZIP2` to `ZLIB`.

  `ZLIB` is less CPU-intensive than `BZIP2`. See "Configuring the Backup Compression Algorithm" on page 6-6.

- If the database host uses multiple CPUs, and if the backup uses binary compression, then increase the number of channels.

- If the backup is encrypted, then change the encryption algorithm to `AES128`.

  The `AES128` algorithm is the least CPU-intensive. See "Configuring the Backup Encryption Algorithm" on page 6-10.

- If RMAN is backing up to tape, then try the following adjustments:

  - Adjust the size of the tape I/O buffers.

    Use the `PARMS` and `BLKSIZE` parameters of the `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` command to set the size. The size of the tape I/O buffers is platform-dependent. The `BLKSIZE` setting overrides the default.

  - Adjust settings in the media management software.

    A number of media manager settings, including the tape block size, may affect backup performance.

- If RMAN is backing up files to ASM, then increase the number of channels.

  For example, if RMAN is backing up the database to a single disk group with 16 physical disks, then allocate or configure at least 4 disk channels, up to a maximum of 16.

22

Troubleshooting RMAN Operations

This chapter describes how to troubleshoot Recovery Manager. This chapter contains these topics:

- Interpreting RMAN Message Output
- Using V$ Views for RMAN Troubleshooting
- Testing the Media Management API
- Terminating an RMAN Command

Interpreting RMAN Message Output

Recovery Manager provides detailed error messages that can aid in troubleshooting problems. Also, the Oracle database server and third-party media vendors generate useful debugging output of their own. The discussion which follows explains how to identify and interpret the different errors you may encounter.

Identifying Types of Message Output

Output that is useful for troubleshooting failed or hung RMAN jobs is located in several different places, as explained in the following table.

Table 22–1 Types of Message Output

Type of Output	Produced By	Location	Description
RMAN messages	RMAN	Completed job information is in `V$RMAN_STATUS` and `RC_RMAN_STATUS`. Current job information is in `V$RMAN_OUTPUT`. When running RMAN from the command line, you can direct output to the following places: ▪ Standard output ▪ A log file specified by `LOG` on the command line or the `SPOOL LOG` command ▪ A file created by redirecting RMAN output (for example, UNIX > operator)	Contains actions relevant to the RMAN job as well as error messages generated by RMAN, the database server, and the media vendor. RMAN error messages have an `RMAN-`*xxxxx* prefix. Normal action descriptions do not have a prefix. Note that you can execute the following PL/SQL to remove all entries from `V$RMAN_STATUS`: `SYS.DBMS_BACKUP_RESTORE.resetCfileSection(28);` The preceding function removes all job-related entries. No rows will be visible until new backup jobs are shown in `V$RMAN_BACKUP_JOB_DETAILS`.
`alert_`*SID*`.log`	Oracle Database	The `alert` subdirectory of the **Automatic Diagnostic Repository (ADR)** home.	Contains a chronological log of errors, initialization parameter settings, and administration operations. Records values for overwritten control file records.

Table 22–1 (Cont.) Types of Message Output

Type of Output	Produced By	Location	Description
Oracle trace file	Oracle Database	The `trace` subdirectory of the ADR home.	Contains detailed output generated by Oracle server processes. This file is created when an `ORA-600` or `ORA-3113` error message occurs, whenever RMAN cannot allocate a channel, and when the database fails to load the media management library.
`sbtio.log`	Third-party media management software	The `trace` subdirectory of the ADR home.	Contains vendor-specific information written by the media management software. This log does not contain Oracle server or RMAN errors.
Media manager log file	Third-party media management software	The filenames for any media manager logs other than `sbtio.log` are determined by the media management software.	Contains information on the functioning of the media management device.

Recognizing RMAN Error Message Stacks

RMAN reports errors as they occur. If an error is not retrievable, that is, RMAN cannot perform failover to another channel to complete a particular job step, then RMAN also reports a summary of the errors after all job sets complete. This feature is known as **deferred error reporting**.

One way to determine whether RMAN encountered an error is to examine its return code, as described in "Identifying RMAN Return Codes" on page 22-7. A second way is to search the RMAN output for the string `RMAN-00569`, which is the message number for the error stack banner. All RMAN errors are preceded by this error message. If you do not see an `RMAN-00569` message in the output, then there are no errors. The following example shows a syntax error.

Example 22–1 RMAN Syntax Error

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-00558: error encountered while parsing input commands
RMAN-01005: syntax error: found ")": expecting one of: "archivelog, backup,
backupset, controlfilecopy, current, database, datafile, datafilecopy, (, plus, ;,
tablespace"
RMAN-01007: at line 1 column 18 file: standard input
```

Identifying Error Codes

Typically, you find the following types of error codes in RMAN message stacks:

- Errors prefixed with `RMAN-`
- Errors prefixed with `ORA-`
- Errors preceded by the line `Additional information:`

> **See Also:** *Oracle Database Error Messages* for explanations of `RMAN` and `ORA` error codes

RMAN Error Message Numbers

Table 22–2 indicates the error ranges for common RMAN error messages, all of which are described in *Oracle Database Error Messages*.

Table 22–2 RMAN Error Message Ranges

Error Range	Cause
0550-0999	Command-line interpreter
1000-1999	Keyword analyzer
2000-2999	Syntax analyzer
3000-3999	Main layer
4000-4999	Services layer
5000-5499	Compilation of `RESTORE` or `RECOVER` command
5500-5999	Compilation of `DUPLICATE` command
6000-6999	General compilation
7000-7999	General execution
8000-8999	PL/SQL programs
9000-9999	Low-level keyword analyzer
10000-10999	Server-side execution
11000-11999	Interphase errors between PL/SQL and RMAN
12000-12999	Recovery catalog packages

ORA-19511: Media Manager Errors

In the event of a media manager error, `ORA-19511` is signalled, and the media manager is expected to provide RMAN a descriptive error. RMAN will display the error passed back to it by the media manager. For example, you might see this:

```
ORA-19511: Error received from media manager layer, error text:
   sbtpvt_open_input: file .* does not exist or cannot be accessed, errno = 2
```

The message from the media manager should provide you with enough information to let you fix the root problem. If it does not, you should refer to the documentation for your media manager or contact your media management vendor support representative for further information. `ORA-19511` errors originate with the media manager, not the Oracle database. The database merely passes the message on from the media manager. The cause can only be addressed by the media management vendor.

Note that if you are still using an **SBT** 1.1-compliant media management layer, you may see some additional error message text. Output from an SBT 1.1-compliant media management layer is similar to the following:

```
ORA-19507: failed to retrieve sequential file, handle="c-140148591-20031014-06",
parms=""
ORA-27007: failed to open file
Additional information: 7000
Additional information: 2
ORA-19511: Error received from media manager layer, error text:
   SBT error = 7000, errno = 0, sbtopen: backup file not found
```

The "Additional information" provided uses error codes specific to SBT 1.1. The values displayed correspond to the media manager message numbers and error text listed in Table 22–3. RMAN re-signals the error, as an ORA-19511 `Error received from media manager layer` error, and a general error message related to the error code

returned from the media manager and including the SBT 1.1 error number is then displayed.

The SBT 1.1 error messages are listed here for your reference. Table 22–3 lists media manager message numbers and their corresponding error text. In the error codes, `O/S` stands for *operating system*. The errors marked with an asterisk are internal and should not typically be seen during normal operation.

Table 22–3 Media Manager Error Message Ranges

Cause	No.	Message
sbtopen	7000	Backup file not found (only returned for read)
	7001	File exists (only returned for write)
	7002*	Bad mode specified
	7003	Invalid block size specified
	7004	No tape device found
	7005	Device found, but busy; try again later
	7006	Tape volume not found
	7007	Tape volume is in-use
	7008	I/O Error
	7009	Can't connect with Media Manager
	7010	Permission denied
	7011	O/S error for example malloc, fork error
	7012*	Invalid argument(s) to sbtopen
sbtclose	7020*	Invalid file handle or file not open
	7021*	Invalid flags to sbtclose
	7022	I/O error
	7023	O/S error
	7024*	Invalid argument(s) to sbtclose
	7025	Can't connect with Media Manager
sbtwrite	7040*	Invalid file handle or file not open
	7041	End of volume reached
	7042	I/O error
	7043	O/S error
	7044*	Invalid argument(s) to sbtwrite
sbtread	7060*	Invalid file handle or file not open
	7061	EOF encountered
	7062	End of volume reached
	7063	I/O error
	7064	O/S error
	7065*	Invalid argument(s) to sbtread

Table 22–3 (Cont.) Media Manager Error Message Ranges

Cause	No.	Message
sbtremove	7080	Backup file not found
	7081	Backup file in use
	7082	I/O Error
	7083	Can't connect with Media Manager
	7084	Permission denied
	7085	O/S error
	7086*	Invalid argument(s) to sbtremove
sbtinfo	7090	Backup file not found
	7091	I/O Error
	7092	Can't connect with Media Manager
	7093	Permission denied
	7094	O/S error
	7095*	Invalid argument(s) to sbtinfo
sbtinit	7110*	Invalid argument(s) to sbtinit
	7111	O/S error

Interpreting RMAN Error Stacks

Sometimes you may find it difficult to identify the useful messages in the RMAN error stack. Note the following tips and suggestions:

- Read the messages from the bottom up, because this is the order in which RMAN issues the messages. The last one or two errors displayed in the stack are often the most informative.
- When using an SBT 1.1 media management layer and presented with SBT 1.1 style error messages containing the "`Additional information:`" numeric error codes, look for the `ORA-19511` message that follows for the text of error messages passed back to RMAN by the media manager. These should identify the real failure in the media management layer.
- Look for the `RMAN-03002` or `RMAN-03009` message (`RMAN-03009` is the same as `RMAN-03002` but includes the channel ID), immediately following the error banner. These messages indicate which command failed. Syntax errors generate `RMAN-00558`.
- Identify the basic type of error according to the error range chart in Table 22–2 and then refer to *Oracle Database Error Messages* for information on the most important messages.

Interpreting RMAN Errors: Example

You attempt a backup of tablespace `users` and receive the following message:

```
Starting backup at 29-AUG-02
using channel ORA_DISK_1
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of backup command at 08/29/2002 15:14:03
RMAN-20202: tablespace not found in the recovery catalog
RMAN-06019: could not translate tablespace name "USESR"
```

The RMAN-03002 error indicates that the BACKUP command failed. You read the last two messages in the stack first and immediately see the problem: no tablespace user appears in the recovery catalog because you mistyped the name.

Interpreting Server Errors: Example

Assume that you attempt to recover a tablespace and receive the following errors:

```
RMAN> RECOVER TABLESPACE users;

Starting recover at 29-AUG-01
using channel ORA_DISK_1

starting media recovery
media recovery failed
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of recover command at 08/29/2007 15:18:43
RMAN-11003: failure during parse/execution of SQL statement: alter database
recover if needed tablespace USERS
ORA-00283: recovery session canceled due to errors
ORA-01124: cannot recover data file 8 - file is in use or recovery
ORA-01110: data file 8: '/oracle/oradata/trgt/users01.dbf'
```

As suggested, you start reading from the bottom up. The ORA-01110 message explains there was a problem with the recovery of datafile users01.dbf. The second error indicates that the database cannot recover the datafile because it is in use or already being recovered. The remaining RMAN errors indicate that the recovery session was cancelled due to the server errors. Hence, you conclude that because you were not already recovering this datafile, the problem must be that the datafile is online and you need to take it offline and restore a backup.

Interpreting SBT 2.0 Media Management Errors: Example

Assume that you use a tape drive and see the following output during a backup job:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
ORA-19624: operation failed, retry possible
ORA-19507: failed to retrieve sequential file, handle="/tmp/mydir", parms=""
ORA-27029: skgfrtrv: sbtrestore returned error
ORA-19511: Error received from media manager layer, error text:
  sbtpvt_open_input:file /tmp/mydir does not exist or cannot be accessed, errno=2
```

The error text displayed following the ORA-19511 error is generated by the media manager and describes the real source of the failure. Refer to the media manager documentation to interpret this error.

Interpreting SBT 1.1 Media Management Errors: Example

Assume that you use a tape drive and see the following output during a backup job:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03009: failure of backup command on c1 channel at 09/04/2007 13:18:19
ORA-19506: failed to create sequential file, name="07d36ecp_1_1", parms=""
ORA-27007: failed to open file
```

```
SVR4 Error: 2: No such file or directory
Additional information: 7005
Additional information: 1
ORA-19511: Error received from media manager layer, error text:
   SBT error = 7005, errno = 2, sbtopen: system error
```

The main information of interest returned by SBT 1.1 media managers is the error code in the "Additional information" line:

```
Additional information: 7005
```

Referring to Table 22–3, " Media Manager Error Message Ranges", you discover that error `7005` means that the media management device is busy. So, the media management software is not able to write to the device because it is in use or there is a problem with it.

> **Note:** The `sbtio.log` contains information written by the media management software, *not* the Oracle database. Thus, you must consult your media vendor documentation to interpret the error codes and messages. If no information is written to the `sbtio.log`, then contact your media manager support to ask whether they are writing error messages in some other location, or whether there are steps you need to take to have the media manager errors appear in `sbtio.log`.

Identifying RMAN Return Codes

One way to determine whether RMAN encountered an error is to examine its return code or exit status. The RMAN client returns 0 to the shell from which it was invoked if no errors occurred, and a nonzero error value otherwise.

How you access this return code depends upon the environment from which you invoked the RMAN client. For example, if you are running UNIX with the C shell, then, when RMAN completes, the return code is placed in a shell variable called `$status`. The method of returning exit status is a detail specific to the host operating system rather than the RMAN client.

Using V$ Views for RMAN Troubleshooting

When `LIST`, `REPORT` and `SHOW` do not provide all the information you need for RMAN operations, a number of useful `V$` views can provide more details.

Sometimes it is useful to identify exactly what a server session performing a backup and recovery job is doing. The views described in Table 22–4 are useful for obtaining information about RMAN jobs.

Table 22–4 Useful V$ Views for Troubleshooting

View	Description
`V$PROCESS`	Identifies currently active processes.
`V$SESSION`	Identifies currently active sessions. Use this view to determine which database server sessions correspond to which RMAN allocated channels.
`V$SESSION_WAIT`	Lists the events or resources for which sessions are waiting.

You can use the preceding views to perform the following tasks:

- Monitoring RMAN Interaction with the Media Manager
- Correlating Server Sessions with RMAN Channels

Monitoring RMAN Interaction with the Media Manager

You can use the event names in the dynamic performance event views to monitor RMAN calls to the media management API. The event names have one-to-one correspondence with SBT functions, as shown in the following examples:

```
Backup: sbtinit
Backup: ssbtopen
Backup: ssbtread
Backup: ssbtwrite
Backup: ssbtbackup
.
.
.
```

To obtain the complete list of SBT events, you can use the following query:

```
SELECT NAME
FROM   V$EVENT_NAME
WHERE  NAME LIKE '%sbt%';
```

Before making a call to any of functions in the media management API, the server adds a row in `V$SESSION_WAIT`, with the `STATE` column including the string `WAITING`. The `V$SESSION_WAIT.SECONDS_IN_WAIT` column shows the number of seconds that the server has been waiting for this call to return. After an SBT function is returned from the media manager, this row disappears.

A row in `V$SESSION_WAIT` corresponding to an SBT event name does not indicate a problem, because the server updates these rows at runtime. The rows appear and disappear as calls are made and returned. However, if the `SECONDS_IN_WAIT` column is high, then the media manager may be hung.

To monitor the SBT events, you can run the following SQL query:

```
COLUMN EVENT FORMAT a10
COLUMN SECONDS_IN_WAIT FORMAT 999
COLUMN STATE FORMAT a20
COLUMN CLIENT_INFO FORMAT a30

SELECT p.SPID, EVENT, SECONDS_IN_WAIT AS SEC_WAIT,
       sw.STATE, CLIENT_INFO
FROM   V$SESSION_WAIT sw, V$SESSION s, V$PROCESS p
WHERE  sw.EVENT LIKE 's%bt%'
AND    s.SID=sw.SID
AND    s.PADDR=p.ADDR;
```

Examine the SQL output to determine which SBT functions are waiting. For example, the following output indicates that RMAN has been waiting for the `sbtbackup` function to return for ten minutes:

```
SPID EVENT             SEC_WAIT   STATE               CLIENT_INFO
---- ----------------- ---------- ------------------- ------------------------------
8642 Backup: sbtbackup 600        WAITING             rman channel=ORA_SBT_TAPE_1
```

> **Note:** The `V$SESSION_WAIT` view shows only database events, not media manager events.

See Also: *Oracle Database Reference* for descriptions of V$SESSION_WAIT

Correlating Server Sessions with RMAN Channels

To identify which server sessions correspond to which RMAN channels, you can query V$SESSION and V$PROCESS. The SPID column of V$PROCESS identifies the operating system ID number for the process or thread. For example, on UNIX the SPID column shows the process ID, whereas on Windows the SPID column shows the thread ID. You have two basic methods for obtaining this information, depending on whether you have multiple RMAN sessions active concurrently.

Matching Server Sessions with Channels When One RMAN Session Is Active

When only one RMAN session is active, the easiest method for determining the server session ID for an RMAN channel is to execute the following query on the target database while the RMAN job is executing:

```
COLUMN CLIENT_INFO FORMAT a30
COLUMN SID FORMAT 999
COLUMN SPID FORMAT 9999

SELECT s.SID, p.SPID, s.CLIENT_INFO
FROM   V$PROCESS p, V$SESSION s
WHERE  p.ADDR = s.PADDR
AND    CLIENT_INFO LIKE 'rman%';
```

The following shows sample output:

```
 SID SPID         CLIENT_INFO
---- ------------ ------------------------------
  14 8374         rman channel=ORA_SBT_TAPE_1
```

If you set an ID using the RMAN SET COMMAND ID command instead of using the system-generated default ID, then search for that value in the CLIENT_INFO column instead of 'rman%'.

Matching Server Sessions with Channels in Multiple RMAN Sessions

If more than one RMAN session is active, then it is possible for the V$SESSION.CLIENT_INFO column to yield the same information for a channel in each session. For example:

```
 SID SPID         CLIENT_INFO
---- ------------ ------------------------------
  14 8374         rman channel=ORA_SBT_TAPE_1
   9 8642         rman channel=ORA_SBT_TAPE_1
```

In this case, you have the following methods for determining which channel corresponds to which SID value.

Obtaining the Channel ID from the RMAN Output In this method, you must first obtain the sid values from the RMAN output and then use these values in your SQL query.

To correlate a process with a channel during a backup:

1. In one of the active sessions, run the RMAN job as normal and examine the output to get the sid for the channel. For example, the output may show:

   ```
   Starting backup at 21-AUG-01
   allocated channel: ORA_SBT_TAPE_1
   ```

```
channel ORA_SBT_TAPE_1: sid=14 devtype=SBT_TAPE
```

2. Start a SQL*Plus session and then query the joined V$SESSION and V$PROCESS views *while the RMAN job is executing*. For example, enter:

```
COLUMN CLIENT_INFO FORMAT a30
COLUMN SID FORMAT 999
COLUMN SPID FORMAT 9999

SELECT s.SID, p.SPID, s.CLIENT_INFO
FROM   V$PROCESS p, V$SESSION s
WHERE  p.ADDR = s.PADDR
AND    CLIENT_INFO LIKE 'rman%'
/
```

   Use the sid value obtained from the first step to determine which channel corresponds to which server session:

```
       SID SPID         CLIENT_INFO
---------- ------------ ------------------------------
        14 2036         rman channel=ORA_SBT_TAPE_1
        12 2066         rman channel=ORA_SBT_TAPE_1
```

Correlating Server Sessions with Channels by Using SET COMMAND ID In this method, you specify a command ID string in the RMAN backup script. You can then query V$SESSION.CLIENT_INFO for this string.

To correlate a process with a channel during a backup:

1. In each session, set the COMMAND ID to a different value *after* allocating the channels and then back up the desired object. For example, enter the following in session 1:

```
RUN
{
  ALLOCATE CHANNEL c1 TYPE disk;
  SET COMMAND ID TO 'sess1';
  BACKUP DATABASE;
}
```

   Set the command ID to a string such as sess2 in the job running in session 2:

```
RUN
{
  ALLOCATE CHANNEL c1 TYPE sbt;
  SET COMMAND ID TO 'sess2';
  BACKUP DATABASE;
}
```

2. Start a SQL*Plus session and then query the joined V$SESSION and V$PROCESS views *while the RMAN job is executing*. For example, enter:

```
SELECT SID, SPID, CLIENT_INFO
FROM   V$PROCESS p, V$SESSION s
WHERE  p.ADDR = s.PADDR
AND    CLIENT_INFO LIKE '%id=sess%';
```

   If you run the SET COMMAND ID command in the RMAN job, then the CLIENT_INFO column displays in the following format:

```
id=command_id,rman channel=channel_id
```

For example, the following shows sample output:

```
 SID SPID         CLIENT_INFO
---- ------------ ------------------------------
  11 8358         id=sess1
  15 8638         id=sess2
  14 8374         id=sess1,rman channel=c1
   9 8642         id=sess2,rman channel=c1
```

The rows that contain the string `rman channel` show the channel performing the backup. The remaining rows are for the connections to the target database.

> **See Also:** *Oracle Database Backup and Recovery Reference* for `SET COMMAND ID` syntax, and *Oracle Database Reference* for more information on `V$SESSION` and `V$PROCESS`

Testing the Media Management API

On some platforms, Oracle provides a diagnostic tool called `sbttest`. This utility performs a simple test of the media management software by attempting to communicate with the media manager as the Oracle database server would.

Obtaining the sbttest Utility

On UNIX, the `sbttest` utility is typically located in `$ORACLE_HOME/bin`. If for some reason the utility is not included with your platform, then contact Oracle Support to obtain the C version of the program. You can compile this version of the program on all UNIX platforms.

Note that on platforms such as Solaris, you do not have to relink when using `sbttest`. On other platforms, relinking may be necessary.

Obtaining Online Documentation for the sbttest Utility

For online documentation of `sbttest`, issue the following on the command line:

```
% sbttest
```

The program displays the list of possible arguments for the program:

```
Error: backup file name must be specified
Usage: sbttest backup_file_name # this is the only required parameter
               <-dbname database_name>
               <-trace trace_file_name>
               <-remove_before>
               <-no_remove_after>
               <-read_only>
               <-no_regular_backup_restore>
               <-no_proxy_backup>
               <-no_proxy_restore>
               <-file_type n>
               <-copy_number n>
               <-media_pool n>
               <-os_res_size n>
               <-pl_res_size n>
               <-block_size block_size>
               <-block_count block_count>
               <-proxy_file os_file_name bk_file_name
                           [os_res_size pl_res_size block_size block_count]>
               <-libname sbt_library_name>
```

The display also indicates the meaning of each argument. For example, following is the description for two optional parameters:

```
Optional parameters:
  -dbname  specifies the database name which will be used by SBT
           to identify the backup file. The default is "sbtdb"
  -trace   specifies the name of a file where the Media Management
           software will write diagnostic messages.
```

Using the sbttest Utility

Use `sbttest` to perform a quick test of the media manager.

If `sbttest` returns 0, then the test ran without error, which means that the media manager is correctly installed and can accept a data stream and return the same data when requested. If `sbttest` returns a nonzero value, then either the media manager is not installed or it is not configured correctly.

To use sbttest:

1. Make sure the program is installed and included in the system path by typing `sbttest` at the command line:

   ```
   % sbttest
   ```

   If the program is operational, then you should see a display of the online documentation.

2. Execute the program, specifying any of the arguments described in the online documentation. For example, enter the following to create test file `some_file.f` and write the output to `sbtio.log`:

   ```
   % sbttest some_file.f -trace sbtio.log
   ```

   You can also test a backup of an existing datafile. For example, this command tests datafile `tbs_33.f` of database `prod`:

   ```
   % sbttest tbs_33.f -dbname prod
   ```

3. Examine the output. If the program encounters an error, then it provides messages describing the failure. For example, if the database cannot find the library, you see:

   ```
   libobk.so could not be loaded. Check that it is installed properly, and that
    LD_LIBRARY_PATH environment variable (or its equivalent on your platform)
    includes the directory where this file can be found. Here is some additional
    information on the cause of this error:
   ld.so.1: sbttest: fatal: libobk.so: open failed: No such file or directory
   ```

Note that in some cases `sbttest` can work but an RMAN backup does not. The reasons can be the following:

- The user who starts `sbttest` is not the owner of the Oracle processes.
- If the database server is not linked with the media management library or cannot load it dynamically when needed, then RMAN backups to the media manager fail, but `sbttest` may still work.
- The `sbttest` program passes all environment parameters from the shell but RMAN does not.

Terminating an RMAN Command

There are several ways to terminate an RMAN command in the middle of execution:

- The preferred method is to press CTRL+C (or the equivalent "attention" key combination for your system) in the RMAN interface. This also terminates allocated channels, unless they are hung in the media management code, as happens when, for example, when they are waiting for a tape to be mounted.
- You can kill the server session corresponding to the RMAN channel by running the SQL ALTER SYSTEM KILL SESSION statement.
- You can terminate the server session corresponding to the RMAN channel on the operating system.

Terminating the Session with ALTER SYSTEM KILL SESSION

You can identify the Oracle session ID for an RMAN channel by looking in the RMAN log for messages with the format shown in the following example:

```
channel ch1: sid=15 devtype=SBT_TAPE
```

The sid and devtype are displayed for each allocated channel. Note that the Oracle sid is different from the operating system process ID. You can kill the session using a SQL ALTER SYSTEM KILL SESSION statement.

ALTER SYSTEM KILL SESSION takes two arguments, the sid printed in the RMAN message and a serial number, both of which can be obtained by querying V$SESSION. For example, run the following statement, where *sid_in_rman_output* is the number from the RMAN message:

```
SELECT SERIAL#
FROM   V$SESSION
WHERE  SID=sid_in_rman_output;
```

Then, run the following statement, substituting the *sid_in_rman_output* and serial number obtained from the query:

```
ALTER SYSTEM KILL SESSION 'sid_in_rman_output,serial#';
```

Note that this will not unhang the session if the session is hung in media manager code.

Terminating the Session at the Operating System Level

Finding and killing the processes that are associated with the server sessions is operating system specific. On some platforms the server sessions are not associated with any processes at all. Refer to your operating system specific documentation for more information.

Terminating an RMAN Session That Is Hung in the Media Manager

You may sometimes need to kill an RMAN job that is hung in the media manager. The best way to terminate RMAN when the channel connections are hung in the media manager is to kill the session in the media manager. If this action does not solve the problem, then on some platforms, such as Unix, you may be able to kill the Oracle processes of the connections. (Note that killing the Oracle processes may cause problems from the media manager. See your media manager documentation for details.)

Components of an RMAN Session

The nature of an RMAN session depends on the operating system. In UNIX, an RMAN session has the following processes associated with it:

- The **RMAN client process** itself
- The **default channel**, the initial connection to the target database
- One **target connection** to the target database corresponding to each allocated channel
- The **catalog connection** to the recovery catalog database, if you use a recovery catalog
- An **auxiliary connection** to an auxiliary instance, during `DUPLICATE` or TSPITR operations
- A **polling connection** to the target database, used for monitoring RMAN command execution on the various allocated channels. By default, RMAN makes one polling connection. RMAN makes additional polling connections if you use different connect strings in the `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` commands. One polling connection exists for each distinct connect string used in the `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` command.

Process Behavior During a Hung Job

RMAN usually hangs because one of the channel connections is waiting in the media manager code for a tape resource. The catalog connection and the default channel appear to hang, because they are waiting for RMAN to tell them what to do. Polling connections seem to be in an infinite loop while polling the RPC under the control of the RMAN process.

If you kill the RMAN process itself, then you also kill the catalog connection, the auxiliary connection, the default channel, and the polling connections. If target and auxiliary connections are not hung in the media manager code, they also terminate. If either the target connection or any of the auxiliary connections are executing in the media management layer, then they will not terminate until the processes are manually killed at the operating system level.

Not all media managers can detect the termination of the Oracle process. Those which cannot may keep resources busy or continue processing. Consult your media manager documentation for details.

Terminating the catalog connection does not cause the RMAN process to terminate because RMAN is not performing catalog operations while the backup or restore is in progress. Removing default channel and polling connections causes the RMAN process to detect that one of the channels has died and then proceed to exit. In this case, the connections to the hung channels remain active as described previously.

Terminating an RMAN Session: Basic Steps

After the hung channels in the media manager code are killed, the RMAN process detects this termination and proceed to exit, removing all connections except target connections that are still operative in the media management layer. The warning about the media manager resources still applies in this case.

To terminate an Oracle process that is hung in the media manager:

1. Query `V$SESSION` and `V$SESSION_WAIT` as described in "Using V$ Views for RMAN Troubleshooting" on page 22-7. For example, execute the following query:

```
COLUMN EVENT FORMAT a10
```

```
COLUMN SECONDS_IN_WAIT FORMAT 999
COLUMN STATE FORMAT a20
COLUMN CLIENT_INFO FORMAT a30

SELECT p.SPID, s.EVENT, s.SECONDS_IN_WAIT AS SEC_WAIT,
       sw.STATE, s.CLIENT_INFO
FROM   V$SESSION_WAIT sw, V$SESSION s, V$PROCESS p
WHERE  sw.EVENT LIKE 'sbt%'
AND    s.SID=sw.SID
AND    s.PADDR=p.ADDR;
```

Examine the SQL output to determine which SBT functions are waiting. For example, the output may be as follows:

```
SPID EVENT        SEC_WAIT STATE               CLIENT_INFO
---- ---------- ---------- ------------------- -------------
8642 sbtwrite2         600 WAITING             rman channel=ORA_SBT_TAPE_1
8374 sbtwrite2         600 WAITING             rman channel=ORA_SBT_TAPE_2
```

2. Using operating system-level tools appropriate to your platform, kill the hung sessions. For example, on Linux execute a `kill -9` command:

   ```
   % kill -9 8642 8374
   ```

   Some platforms include a command-line utility called `orakill` that enables you to kill a specific thread. From a command prompt, run the following command, where *`sid`* identifies the database instance to target, and the *`thread_id`* is the `SPID` value from the query in step 1:

   ```
   orakill sid thread_id
   ```

3. Check that the media manager also clears its processes. If any remain, the next backup or restore operation may hang again, due to the previous hang. In some media managers, the only solution is to shut down and restart the media manager. If the documentation from the media manager does not provide the needed information, contact technical support for the media manager.

   See Also: Your operating system specific documentation for the relevant commands

Part VII

Transferring Data with RMAN

The following chapters describe how to use RMAN for database and tablespace transport and migration. This part of the book contains these chapters:

- Chapter 23, "Duplicating a Database"
- Chapter 24, "Creating Transportable Tablespace Sets"
- Chapter 25, "Transporting Data Across Platforms"
- Chapter 26, "Performing ASM Data Migration"

23

Duplicating a Database

This chapter describes how to use the DUPLICATE command to create a duplicate database. This chapter contains these topics:

- Overview of RMAN Database Duplication
- Making Backups and Archived Logs Accessible to the Duplicate Instance
- Choosing a Strategy for Naming Duplicate Files
- Preparing the Auxiliary Instance
- Starting and Configuring RMAN Before Duplication
- Duplicating a Database
- Naming Duplicate Files with Alternative Techniques
- RMAN Duplication Scenarios

Overview of RMAN Database Duplication

This section explains the basic concepts and tasks involved in database duplication.

Purpose of Database Duplication

The goal of database duplication is the creation of a **duplicate database**, which is a separate database that contains all or a subset of the data in the **source database**. A duplicate database is useful for a variety of purposes, most of which involve testing. You can perform the following tasks in a duplicate database:

- Test backup and recovery procedures
- Test an upgrade to a new release of Oracle Database
- Test the effect of applications on database performance
- Generate reports
- Export data such as a table that was inadvertently dropped from the production database, and then import it back into the production database

For example, you can duplicate the production database on host1 to host2, and then use the duplicate database on host2 to practice restoring and recovering this database while the production database on host1 operates as usual.

If you copy a database by means of operating system utilities rather than with DUPLICATE, then the DBID of the copied database remains the same as the original database. To register the copy database in the same recovery catalog with the original, you must change the DBID with the DBNEWID utility (refer to *Oracle Database*

Utilities). In contrast, the DUPLICATE command automatically assigns the copied database a different DBID so that it can be registered in the same recovery catalog as the database from which it was copied.

A duplicate database serves a different purpose from a **physical standby database**, although both are created with the DUPLICATE command. A standby database is a copy of the primary database that you update continually with archived logs from the primary database. If the primary database is inaccessible, then you can perform failover to the standby database and make it the new primary database. A duplicate database, on the other hand, cannot be used in this way: it is not intended for failover scenarios and does *not* support the various standby recovery and failover options.

> **See Also:** *Oracle Data Guard Concepts and Administration* to learn how to create a standby database with the DUPLICATE command

Basic Concepts of Database Duplication

You create a duplicate database by using the RMAN DUPLICATE command. The duplicate database is the copied database, whereas the source database is the database that you are copying. The duplicate database has a different DBID from the source database and functions entirely independently.

The **source host** is the computer that hosts the source database, whereas the **destination host** is the computer that hosts the duplicate database. The source host and destination host can be the same or different. The source database instance is the Oracle instance associated with the source database. The instance associated with the duplicate database is called the **auxiliary instance**.

A duplicate database can include the same contents as the source database or contain only a subset of the tablespaces in the source database. For example, you can use the TABLESPACE option to duplicate only specified tablespaces, or the SKIP READONLY option to exclude read-only tablespaces from the duplicate database.

You can use either of the following techniques: **active database duplication** or **backup-based duplication**. Active database duplication copies the live source database over the network to the duplicate database instance, whereas backup-based duplication uses pre-existing RMAN backups and copies.

The principal work of the duplication is performed by an **auxiliary channel**. This channel corresponds to a server session on the auxiliary instance on the destination host. As part of the duplicating operation, RMAN automates the following steps:

1. Creates a control file for the duplicate database
2. Restarts the auxiliary instance and mounts the duplicate control file
3. Creates the duplicate datafiles and recovers them with incremental backups and archived redo logs

   RMAN must perform incomplete recovery because the online redo log files in the source database are not backed up and cannot be applied to the duplicate database. The farthest that RMAN can go in recovery of the duplicate database is the most recent redo log archived by the source database.
4. Opens the duplicate database with the RESETLOGS option

> **See Also:** The DUPLICATE entry in *Oracle Database Backup and Recovery Reference* for a complete list of which files are copied to the duplicate database

Basic Steps of Database Duplication

Before duplicating a database you must meet a number of prerequisites. For example, the source and duplicate databases must be on the same platform. Review the "Prerequisites" section of the DUPLICATE command entry in the *Oracle Database Backup and Recovery Reference* for a complete list.

To duplicate a database:

1. If you use backup-based duplication, then make database backups and archived redo logs available to the auxiliary instance on the destination host. If you use active database duplication, then skip this step.

   This task is described in "Making Backups and Archived Logs Accessible to the Duplicate Instance" on page 23-3.

2. Decide how to provide names for the duplicate control files, datafiles, online redo logs, and tempfiles. If you are duplicating to a different host that uses the same directory structure as the source host, and if you want to name the duplicate files the same as the source database files, then skip to the next step.

   This task is described in "Choosing a Strategy for Naming Duplicate Files" on page 23-6.

3. Create an initialization parameter file for use by the auxiliary instance on the destination host, and then start the auxiliary instance.

   This task is described in "Preparing the Auxiliary Instance" on page 23-7.

4. Start RMAN, connect as TARGET to the source database, as AUXILIARY to the instance of the duplicate database, and optionally as CATALOG to the recovery catalog database. If necessary configure channels for use by the DUPLICATE command.

   This task is described in "Starting and Configuring RMAN Before Duplication" on page 23-10.

5. Execute the DUPLICATE command.

   This task is described in "Duplicating a Database" on page 23-11.

Making Backups and Archived Logs Accessible to the Duplicate Instance

If you are using active database duplication, then skip this section and proceed to "Choosing a Strategy for Naming Duplicate Files" on page 23-6.

RMAN uses the metadata in the duplicate control file to locate backups and archived logs needed for duplication. You must ensure that the auxiliary channels can access all datafile backups and archived redo logs required to restore and recover the duplicate database. Otherwise, the duplicate operation fails.

The database backup need not have been generated with BACKUP DATABASE. You can use a mix of full and incremental backups of individual datafiles. For example, suppose you have taken a full backup of datafiles 1, 2, and 3. Also, you have taken a level 0 and level 1 backup of datafiles 4, 5, 6. In this case, DUPLICATE can restore files 1, 2, and 3 from the full backup and files 4, 5, and 6 from the level 0 backup. RMAN can apply the incremental level 1 incremental backup to 4, 5, and 6 and apply archived redo logs to all six datafiles.

When using backup-based duplication, decide how to make database backups available to the auxiliary instance. Also, archived redo log files required to recover the duplicate database to the desired point in time must be accessible at the same path by

the host where the duplicate database is to be created. The logs can be available either as backups (for instance, on a media manager) or as image copies (or the actual archived redo log files).

Making SBT Backups Accessible to the Duplicate Instance

When using SBT backups, make the tapes with the backups accessible to the destination host. If media management software is installed on the destination host, then you can physically move the tape to a drive attached to the remote host or use a network-accessible tape server. Depending on the media manager, you typically must inform the remote media management software about the existence of the tapes.

Making Disk Backups Accessible to the Duplicate Instance

When using disk backups, the simplest scenario is when the file system of the source host and destination hosts are identical. For example, assume that the backups of the source database are stored in `/dsk1/bkp`. In this case, you can make disk backups accessible to the destination host in either of the following ways:

- Manually transfer the backups and copies from the source host to the destination host to an identical path. For example, if the backups are in `/dsk1/bkp` on the source host, then FTP them to `/dsk1/bkp` on the destination host.
- Use NFS or shared disks and make sure that the same path is accessible in the destination host. For example, use NFS to mount `/dsk1/bkp` on the destination host and use `/dsk1/bkp` as the mount point name.

A more complicated scenario occurs when you *cannot* use the same directory name on the destination host as you use on the source host. As explained in the following sections, the technique for making backups accessible in this scenario depends on whether you use shared disk.

Making Disk Backups Accessible Without Using Shared Disk

When NFS or shared disk is not an option, then the path that stores the backups must exist on both the source and destination hosts. Assume that you maintain two hosts, `srchost` and `dsthost`. The database on `srchost` is `srcdb`. The RMAN backups of `srcdb` reside in `/dsk1/bkp` on host `srchost`. The directory `/dsk1/bkp` is already in use on the destination host, so you intend to store backups in `/dsk2/dup` on the destination host. You must update the controlfile on the duplicate site with the new location of the dumpfiles.

Follow these steps to transfer the backups from the source host to the destination host:

1. Create a new directory in the source host that has the same name as the directory on the destination host that will contain the backups.

   For example, if you intend to store the RMAN backups in `/dsk2/dup` on the destination host, then create `/dsk2/dup` on the source host.

2. On the source host, copy the backups to the directory that you created in the previous step, and then catalog the backups. You can use either of the following techniques:

   - Connect RMAN to the source database as `TARGET` and use the `BACKUP` command to back up the backups, as explained in "Backing Up RMAN Backups" on page 8-26. For example, use the `BACKUP COPY OF DATABASE` command to copy the backups in `/dsk1/bkp` on the source host to `/dsk2/dup` on the source host. In this case, RMAN automatically catalogs the backups in the new location.

- Use an operating system utility to copy the backups in `/dsk1/bkp` on the source host to `/dsk2/dup` on the source host. Afterward, connect RMAN to the source database as `TARGET` and use the `CATALOG` command to update the source database control file with the location of the manually transferred backups.

3. Manually transfer the backups in the new directory on the source host to the identically named directory on the destination host.

   For example, use FTP to transfer the backups in `/dsk2/dup` on the source host to `/dsk2/dup` on the destination host.

As part of the duplication, RMAN will create a duplicate control file listing backups in the `/dsk2/dup` path. The auxiliary channel can then search for backups in `/dsk2/dup` on the destination host and restore them.

Making Disk Backups Accessible Using Shared Disk

Assume that there are two hosts, `srchost` and `dsthost`. The database on `srchost` is called `srcdb`. The backups of `srcdb` reside in `/dsk1/bkp` on host `srchost`. The directory `/dsk1/bkp` is already in use on the destination host, but the directory `/dsk2/dup` is not in use in either host.

Follow these steps to transfer the backups from the source host to the destination host:

1. Create the new backup storage directory in either the source or destination host.

   For example, create backup directory `/dsk2/dup` on the destination host.

2. Mount the directory created in the previous step on the other host, making sure that the directory and the mount point names are the same.

   For example, if you created `/dsk2/dup` on the destination host, then use NFS to mount this directory as `/dsk2/dup` on the source host.

3. Make the backups available in the new location on the destination host. You can use either of the following techniques:

   - Connect RMAN to the source database as `TARGET` and use the `BACKUP` command to back up the backups, as explained in "Backing Up RMAN Backups" on page 8-26. For example, use the `BACKUP COPY OF DATABASE` command to copy the backups in `/dsk1/bkp` on the source host to `/dsk2/dup` on the source host. In this case, RMAN automatically catalogs the backups in the new location.
   - Use an operating system utility to transfer the backups to the new location. For example, FTP the backups from `/dsk1/bkp` on the source host to `/dsk2/dup` on the destination host, or use the `cp` command to copy the backups from `/dsk1/bkp` on the source host to `/dsk2/dup` on the source host. Afterward, connect RMAN to the source database as `TARGET` and use the `CATALOG` command to update the source database control file with the location of the manually transferred backups.

As part of the duplication, RMAN will create a duplicate control file listing backups in the `/dsk2/dup` path. The auxiliary channel can then search for backups in `/dsk2/dup` on the destination host and restore them.

> **See Also:** *Oracle Database Backup and Recovery Reference* for information about the `BACKUP` command

Choosing a Strategy for Naming Duplicate Files

When duplicating a database, RMAN generates names for the duplicate database files. If the destination host uses the same directory structure as the source host, then you can use the same names for the duplicate database files that you used for the source database files. In this case, you do not need to rename the duplicate files, but you do have to specify the NOFILENAMECHECK option on the DUPLICATE command.

If the hosts use different directory structures, or if they use the same structure but you want to name the duplicate files differently, then decide how to generate the names for the duplicate database files. Specifically, decide how to name the control files, datafiles, online redo log files, and tempfiles.

RMAN supports the following strategies for generating names for duplicate files:

- Specify the SPFILE clause on the DUPLICATE command to set all necessary parameters involving filenames, with the exception of DB_FILE_NAME_CONVERT

  Oracle recommends this strategy because it is simplest. When you execute DUPLICATE ... SPFILE, RMAN either restores the server parameter file from a backup or copies it from an active database. RMAN updates the initialization parameter values in the copied server parameter file based on the SPFILE PARAMETER_VALUE_CONVERT and SPFILE SET values (in this order). RMAN then restarts the auxiliary instance with the server parameter file.

- Use one of the alternative techniques described in "Naming Duplicate Files with Alternative Techniques" on page 23-16

  You can use a text editor to set the parameters in the initialization parameter file on the duplicate instance, specify the LOGFILE and DB_FILE_NAME_CONVERT clauses of the DUPLICATE command, or issue SET and CONFIGURE commands. You can combine any of these options to produce the desired effect.

This section describes the SPFILE strategy only. The following list shows DUPLICATE options that you can use to name duplicate files. Refer to *Oracle Database Backup and Recovery Reference* for complete syntax and semantics of the DUPLICATE command.

- SPFILE ... PARAMETER_VALUE_CONVERT '*string_pattern*'

  Specifies conversion strings for initialization parameters whose values specify path names, with the exception of the DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT parameters. The primary purpose of PARAMETER_VALUE_CONVERT is so that you can set a collection of initialization parameters and avoid explicitly setting them one by one.

  > **Note:** PARAMETER_VALUE_CONVERT can update all string values, not just those containing path names. The values are case-sensitive.

- SPFILE ... SET '*string_pattern*'

  Sets the specified initialization parameters to the specified values. You can use SET to set the LOG_FILE_NAME_CONVERT parameter for the online redo logs.

- DB_FILE_NAME_CONVERT '*string_pattern*'

  Specifies a rule for creating the filenames for duplicate datafiles and tempfiles. Note that DB_FILE_NAME_CONVERT specified on the DUPLICATE command overrides the initialization parameter DB_FILE_NAME_CONVERT if it is set in the initialization parameter file.

- NOFILENAMECHECK

Prevents RMAN from checking whether the source database datafiles and online redo logs files share the same names as the duplicated files. This option is necessary when you are creating a duplicate database in a different host that has the same disk configuration, directory structure, and filenames as the host of the source database. If duplicating a database on the same host as the source database, then make sure that NOFILENAMECHECK is not set.

Example 23–1 shows a DUPLICATE command that uses the SPFILE clause to name duplicate files. The PARAMETER_VALUE_CONVERT option substitutes /disk2 for /disk1 in all initialization parameters that specify filenames (with the exception of DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT). The SET LOG_FILE_NAME_CONVERT clause substitutes /disk2 for /disk1 in the filenames of the online redo logs of the duplicate database. The DB_FILE_NAME_CONVERT option replaces /disk1 with /disk2 in the names of the duplicate datafiles and tempfiles.

Example 23–1 Using the SPFILE Clause to Name Duplicate Files

```
DUPLICATE TARGET DATABASE TO dup1
  FROM ACTIVE DATABASE
  DB_FILE_NAME_CONVERT '/disk1','/disk2'
  SPFILE
    PARAMETER_VALUE_CONVERT '/disk1', '/disk2'
    SET LOG_FILE_NAME_CONVERT '/disk1','/disk2'
    SET SGA_MAX_SIZE '200M'
    SET SGA_TARGET '125M';
```

See Also: *Oracle Database Reference* for information about the DB_FILE_NAME_CONVERT initialization parameter

Preparing the Auxiliary Instance

To prepare the auxiliary instance used by the DUPLICATE DATABASE command, perform the following tasks:

- Step 1: Create an Oracle Password File for the Auxiliary Instance
- Step 2: Establish Oracle Net Connectivity to the Auxiliary Instance
- Step 3: Create an Initialization Parameter File for the Auxiliary Instance
- Step 4: Start the Auxiliary Instance with SQL*Plus

Step 1: Create an Oracle Password File for the Auxiliary Instance

A **password file** is required for the auxiliary instance only if one of the following conditions is true:

- You use the RMAN client on a host other than the destination host.
- You duplicate from an active database.

Note: A password file is not required for backup-based duplication. You can use operating system authentication for the auxiliary connection when duplicating to the same host as the source database.

When using the FROM ACTIVE DATABASE option, the source database instance, which is the database instance to which RMAN is connected as TARGET, connects directly to the auxiliary database instance. This connection requires a password file with the same SYSDBA password. You can create the password file manually, making

sure to use the same SYSDBA password as the source database. You may want to create the password file with a single password so you can start the auxiliary instance and enable the source database to connect to it.

Additionally, you can specify the PASSWORD FILE option on the DUPLICATE command. In this case, RMAN copies the source database password file to the destination host and overwrites any existing password file for the auxiliary instance. This technique is useful if the source database password file has multiple passwords that you want to make available on the duplicate database.

If you are creating a standby database with active database duplication, then RMAN always copies the password file to the standby host. You do not need to specify the PASSWORD FILE option. RMAN overwrites any existing password file for the auxiliary instance.

To create a password file:

- Follow the instructions in *Oracle Database Administrator's Guide* to create a password file. Note that the types of filenames allowed for password file are operating system-specific.

> **See Also:** *Oracle Data Guard Concepts and Administration* to learn how to create a standby database with the DUPLICATE command

Step 2: Establish Oracle Net Connectivity to the Auxiliary Instance

The auxiliary instance must be available through Oracle Net if either of the following conditions is met:

- You use the RMAN client on a host other than the destination host.
- You duplicate from an active database.

When duplicating from an active database, you must first have connected as SYSDBA to the auxiliary instance by means of a net service name. This net service name must also be available on the source database instance. The source database instance, to which RMAN is connected as TARGET, uses this net service name to connect directly to the auxiliary database instance.

Step 3: Create an Initialization Parameter File for the Auxiliary Instance

Create a text-based initialization parameter file for the auxiliary instance.

If you are using the SPFILE technique for naming duplicate files described in "Choosing a Strategy for Naming Duplicate Files" on page 23-6, then the only necessary parameter is DB_NAME, which you can set to an arbitrary value. You do not need to set parameters such as DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT in the initialization parameter file because you can set these and all other parameters in the DUPLICATE command itself.

If you are not using the SPFILE technique, then you need to set initialization parameters to set in the initialization parameter file. Table 23–1 describes these initialization parameters.

Table 23–1 Auxiliary Instance Initialization Parameters

Initialization Parameter	Value	Status
DB_NAME	The same name used in the DUPLICATE command. The DB_NAME setting for the duplicate database must be unique among databases in its Oracle home.	Required
CONTROL_FILES	Control file locations. **See Also:** "Naming Duplicate Control Files" on page 23-17	Required (unless you set parameters for Oracle-managed files)
DB_BLOCK_SIZE	The block size for the duplicate database. This block size must match the block size of the source database. If the source database parameter file contains a value for the DB_BLOCK_SIZE initialization parameter, then you must specify the same value for the auxiliary instance. If no DB_BLOCK_SIZE is specified in the source database initialization parameter file, however, then do not specify DB_BLOCK_SIZE in the auxiliary instance. **See Also:** "Naming Duplicate Files with Alternative Techniques" on page 23-16	Required if this initialization parameter is set in source database
DB_FILE_NAME_CONVERT	Pairs of strings for converting the names of datafiles and tempfiles. Note that you can also specify DB_FILE_NAME_CONVERT on the DUPLICATE command itself. **See Also:** "Naming Duplicate Datafiles" on page 23-18 and "Naming Duplicate Tempfiles" on page 23-19	Optional
LOG_FILE_NAME_CONVERT	Pairs of strings for naming online redo log files. **See Also:** "Naming Duplicate Online Redo Log Files" on page 23-17	Optional
DB_CREATE_FILE_DEST	Location for Oracle-managed datafiles. **See Also:** "Naming Duplicate Datafiles" on page 23-17 and "Naming Duplicate Tempfiles" on page 23-17	Optional
DB_CREATE_ONLINE_LOG_DEST_*n*	Location for Oracle-managed online redo log files. **See Also:** "Naming Duplicate Online Redo Log Files" on page 23-17.	Optional
DB_RECOVERY_FILE_DEST	Location for flash recovery area. **See Also:** "Naming Duplicate Files with Alternative Techniques" on page 23-16	Optional

If necessary, set other initialization parameters, including the parameters that allow you to connect as SYSDBA through Oracle Net, as needed.

When duplicating to the same host or to a new host with a different file system, pay attention to all initialization parameters specifying path names. Verify that all paths are accessible on the host where the database is being duplicated.

Example 23–2 shows sample settings for a sample initialization parameter file.

Example 23–2 Sample Initialization Parameter File for the Auxiliary Instance

```
DB_NAME=dupdb
CONTROL_FILES=(/dup/oracle/oradata/prod/control01.ctl,
               /dup/oracle/oradata/prod/control02.ctl)
DB_FILE_NAME_CONVERT=(/oracle/oradata/prod/,/dup/oracle/oradata/prod/)
LOG_FILE_NAME_CONVERT=(/oracle/oradata/prod/redo,/dup/oracle/oradata/prod/redo)
```

Step 4: Start the Auxiliary Instance with SQL*Plus

Before beginning RMAN duplication, use SQL*Plus to connect to the auxiliary database instance with SYSOPER privileges. Start the instance in NOMOUNT mode, specifying the PFILE parameter if necessary:

```
SQL> STARTUP NOMOUNT
```

> **Note:** Because the auxiliary instance does not yet have a control file, you can only start the instance in NOMOUNT mode. Do not create a control file or try to mount or open the auxiliary instance.

RMAN shuts down and restarts the auxiliary instance as part of the duplication. Thus, it is a good idea to have a server-side initialization parameter file for the auxiliary instance in the default location on the destination host.

If you use the SPFILE clause of the DUPLICATE command, then RMAN copies or restores the source database server parameter file to the default location for the auxiliary instance on the destination host. If you do not use the SPFILE clause, then either copy the server parameter file to the destination host or specify a text-based initialization parameter file with the PFILE parameter on the DUPLICATE command. Note that the text-based initialization parameter file for the auxiliary instance must reside on the same host as the RMAN client used to perform the duplication.

Starting and Configuring RMAN Before Duplication

Before executing the DUPLICATE DATABASE command, perform the following tasks:

- Step 1: Start RMAN and Connect to the Database Instances
- Step 2: Mount or Open the Source Database
- Step 3: Configure RMAN Channels for Use in the Duplication

Step 1: Start RMAN and Connect to the Database Instances

Start RMAN and connect to the source database as TARGET, the duplicate database instance as AUXILIARY, and, if applicable, the recovery catalog database. You can start the RMAN client on any host so long as it can connect to all of the database instances. If the auxiliary instance requires a text-based initialization parameter file, then this file must exist on the same host that runs the RMAN client application.

To start RMAN and connect to the target and auxiliary instances:

1. Start the RMAN client.

   For example, enter the following command at the operating system prompt:

   ```
   % rman
   ```

2. At the RMAN prompt, run CONNECT commands for each database instance.

   In this example, a connection is established to three database instances, all through the use of net service names:

   ```
   RMAN> CONNECT TARGET SYS@prod     # source database

   target database Password: password
   connected to target database: PROD (DBID=39525561)
   ```

```
RMAN> CONNECT AUXILIARY SYS@dupdb   # duplicate database instance

auxiliary database Password: password
connected to auxiliary database: DUPDB (not mounted)

RMAN> CONNECT CATALOG rman@catdb     # recovery catalog database

recovery catalog database Password: password
connected to recovery catalog database
```

Step 2: Mount or Open the Source Database

Before beginning RMAN duplication, mount or open the source database it if it is not already mounted or open. If the source database is open, then archiving must be enabled. If the source database is not open, and if it is not a standby database, then it must have been shut down consistently.

Step 3: Configure RMAN Channels for Use in the Duplication

If necessary for backup-based duplication, configure channels to be used on the auxiliary database instance. Note that it is the channel on the auxiliary instance, not the source database instance, that restores backups.

RMAN can use the same channel configurations set on the source database for duplication even if they do not specify the `AUXILIARY` option. If the auxiliary channels need special parameters (for example, to point to a different media manager), however, then you can configure an automatic channel with the `AUXILIARY` option of the `CONFIGURE` command.

In backup-based duplication, the channel type (`DISK` or `sbt`) of the channel must match the media where the backups of the source database are located. If the backups reside on disk, then the more channels you allocate, the faster the duplication will be. For tape backups, limit the number of channels to the number of devices available.

In active database duplication, you do not have to change your source database channel configuration or configure `AUXILIARY` channels. However, you may want to increase the parallelism setting of your source database disk channels so that RMAN copies files over the network in parallel.

> **See Also:** *Oracle Database Backup and Recovery Reference* for information about the `CONFIGURE` command

Duplicating a Database

This section describes the most basic procedure to create a duplicate database. The procedure depends on how your databases and hosts are configured. This section contains the following topics:

- Duplicating a Database to a Remote Host with the Same Directory Structure
- Duplicating a Database to a Remote Host with a Different Directory Structure
- Creating a Duplicate Database on the Local Host
- Duplicating a Database with Oracle Managed Files or Automatic Storage Management

See Also: *Oracle Database Backup and Recovery Reference* for the complete set of options and clauses that you can specify on the DUPLICATE command

Duplicating a Database to a Remote Host with the Same Directory Structure

The simplest case is to use active database duplication to duplicate the database to a different host and use the same directory structure. If the source database uses a server parameter file (or a backup is available), then you can create a temporary initialization parameter file on the destination host and set only the DB_NAME parameter. You do not have to set additional parameters for specifying duplicate database filenames or transfer backups to the destination host.

To duplicate a database to a remote host with the same directory structure:

1. Follow the steps in "Preparing the Auxiliary Instance" on page 23-7.

   In this example, the initialization parameter file contains only DB_NAME set to an arbitrary value.

2. Follow the steps in "Starting and Configuring RMAN Before Duplication" on page 23-10.

3. Run the DUPLICATE command.

Example 23–3 illustrates how to use DUPLICATE for active duplication. This example requires the NOFILENAMECHECK option because the source database files have the same names as the duplicate database files.

Example 23–3 Duplicating to a Host with the Same Directory Structure (Active)

```
DUPLICATE TARGET DATABASE
  TO dupdb
  FROM ACTIVE DATABASE
  SPFILE
  NOFILENAMECHECK;
```

RMAN automatically copies the server parameter file to the destination host, starts the auxiliary instance with the server parameter file, copies all necessary database files and archived redo logs over the network to the destination host, and recovers the database. Finally, RMAN opens the database with the RESETLOGS option to create the online redo logs.

Assume a slightly different case in which you want to recover the duplicate database to one week ago in order to view the data in the source database as it appeared at that time. Example 23–4 uses backup-based duplication to create a duplicate of the source database as it appeared one week ago. The UNTIL clause is not supported in active database duplication.

Example 23–4 Duplicating a Database to a Past Point in Time (Backup Based)

```
DUPLICATE TARGET DATABASE
  TO dupdb
  PASSWORD FILE
  SPFILE
  NOFILENAMECHECK
  UNTIL TIME 'SYSDATE-7';
```

Note that the command in Example 23–4 specifies the PASSWORD FILE option to indicate that RMAN should duplicate the password file from the source database.

Duplicating a Database to a Remote Host with a Different Directory Structure

If you create the duplicate database on a host with a different directory structure, then you must use some technique to generate filenames for the duplicate database datafiles. The simplest technique is to use active database duplication and to use the SPFILE clause to rename files as explained in "Choosing a Strategy for Naming Duplicate Files" on page 23-6. If the source database uses a server parameter file (or a backup is available), then you can create a temporary initialization parameter file on the destination host and set only the DB_NAME parameter.

Assume that the source database datafiles reside in /oracle/oradata/prod/, and you want to duplicate them to /scratch/oracle/oradata/dupdb/. All of the source database online redo logs reside in /oracle/oradata/prod/redo/, and you want to duplicate them to /scratch/oracle/oradata/dupdb/redo/.

To duplicate a database on a remote host with a different directory structure:

1. Follow the steps in "Preparing the Auxiliary Instance" on page 23-7.

   In this example, you create a temporary initialization parameter file for the duplicate instance with DB_NAME as the only parameter. You can set this parameter to any arbitrary value.

2. Follow the steps in "Starting and Configuring RMAN Before Duplication" on page 23-10.

   For this example, assume that the source database is open and you have automatic channels configured. You start RMAN and connect to the source and auxiliary database instances as follows:

   ```
   RMAN> CONNECT TARGET SYS@prod

   target database Password: password
   connected to target database: PROD1 (DBID=39525561)

   CONNECT AUXILIARY SYS@dupdb

   auxiliary database Password: password
   connected to auxiliary database: DUPDB (not mounted)
   ```

3. Run the DUPLICATE command.

   Example 23–5 specifies names for the duplicate database files with the SPFILE clause in conjunction with the DB_FILE_NAME_CONVERT parameter. Note that the PARAMETER_FILE_CONVERT parameter does not affect LOG_FILE_NAME_CONVERT or DB_FILE_NAME_CONVERT, which is why these two parameters must be set separately.

Example 23–5 Duplicating to a Host with a Different Directory Structure (Active)

```
DUPLICATE TARGET DATABASE
  TO dupdb
  FROM ACTIVE DATABASE
  DB_FILE_NAME_CONVERT '/oracle/oradata/prod/','/scratch/oracle/oradata/dupdb/'
  SPFILE
    PARAMETER_VALUE_CONVERT '/oracle/oradata/prod/',
                            '/scratch/oracle/oradata/dupdb/'
    SET SGA_MAX_SIZE '300M'
    SET SGA_TARGET '250M'
    SET LOG_FILE_NAME_CONVERT '/oracle/oradata/prod/redo/',
                              '/scratch/oracle/oradata/dupdb/redo/';
```

RMAN automatically copies the server parameter file to the destination host and updates the server parameter file on the destination host based on values provided in `PARAMETER_VALUE_CONVERT` and `SET`. RMAN then starts the auxiliary instance with the server parameter file, copies all necessary database files and archived redo logs over the network to the destination host, and recovers the database. Finally, RMAN opens the database with the `RESETLOGS` option to create the online redo logs.

Creating a Duplicate Database on the Local Host

When creating a duplicate database on the same host as the source database, follow the same procedure as for duplicating to a remote host with a different directory structure as described in "Duplicating a Database to a Remote Host with a Different Directory Structure" on page 23-13.

You can duplicate the database to the same Oracle home as the source database, but you must use a different database name from the source database, and convert the filenames by means of the same methods used for conversion on a separate host.

Caution: Do not use the `NOFILENAMECHECK` option when duplicating to the same Oracle home as the primary database. If you do, then the `DUPLICATE` command may overwrite the datafiles of the source database.

Duplicating a Database with Oracle Managed Files or Automatic Storage Management

The following sections discuss requirements for creating a duplicate database when some or all files of the duplicate database are Oracle-managed files or use Automatic Storage Management (ASM):

- Setting Initialization Parameters for Oracle Managed Files
- Setting Initialization Parameters for ASM

> **See Also:** *Oracle Database Storage Administrator's Guide* for an introduction to ASM

Setting Initialization Parameters for Oracle Managed Files

When creating a duplicate database that uses Oracle-managed files, you must set initialization parameters in the auxiliary instance. If you use the `SPFILE` clause of `DUPLICATE` to name the files, then you can set initialization parameters in the `SPFILE` clause. The following table describes the relevant initialization parameters and recommended settings.

Table 23–2 Initialization Parameters for Oracle Managed Files

Initialization Parameter	Purpose	Recommendation
DB_CREATE_FILE_DEST	Specifies the default location for Oracle-managed datafiles. This location is also the default location for Oracle-managed control files and online logs if none of the DB_CREATE_ONLINE_LOG_DEST initialization parameters are specified.	Set this parameter to the location for the Oracle-managed files. Any database files for which no other location is specified are created in DB_CREATE_FILE_DEST by DUPLICATE. You can override the default for specific files using SET NEWNAME, as described in "Using SET NEWNAME to Name Oracle-Managed Files" on page 23-23.
DB_CREATE_ONLINE_LOG_DEST_n	Specifies the default location for Oracle-managed control files and online redo logs. If multiple parameters are set, then one control file and one online redo log is created in each location.	Set these parameters (_1, _2, and so on) only if you want to multiplex the control files and redo logs in multiple locations.
DB_RECOVERY_FILE_DEST	Specifies the default location for the flash recovery area. The flash recovery area contains multiplexed copies of current control files and online redo logs.	Set this parameter if you want a multiplexed copy of the control file and online redo log file in the recovery area.
CONTROL_FILES	Specifies one or more names of control files, separated by commas.	Do not set this parameter if you want the duplicate database control files in an Oracle Managed Files format. Oracle recommends that you use a server parameter file at the duplicate database when using control files in an OMF format.
DB_FILE_NAME_CONVERT	Converts the filename of a new datafile on the primary database to a filename on the duplicate database.	Do not set this parameter. Omitting this parameter enables the database to generate valid Oracle-managed filenames for the duplicate datafiles.
LOG_FILE_NAME_CONVERT	Converts the filename of a new log file on the primary database to the filename of a log file on the standby database.	Do not set this parameter. Omitting this parameter enables the database to generate valid Oracle-managed online redo log filenames. To direct duplicate database online redo log files to Oracle-managed storage, you can use the DB_CREATE_FILE_DEST, DB_RECOVERY_FILE_DEST or DB_CREATE_ONLINE_LOG_DEST_*n* initialization parameters to identify an Oracle-managed location for the online logs.

Setting Initialization Parameters for ASM

The procedure for creating a duplicate database to an ASM location is similar to the procedure described in "Setting Initialization Parameters for Oracle Managed Files" on page 23-14. The difference is that you must set the initialization parameters in the auxiliary instance that control the location where files are created at the duplicate to the ASM disk group. For example, set DB_CREATE_FILE_DEST, DB_CREATE_ONLINE_DEST_*n*, and CONTROL_FILES.

Duplicating a Database from a File System to ASM: Example In this example, you use active database duplication. If the source database uses a server parameter file (or a backup is

available), then you can just create a temporary initialization parameter file on the destination host and set only the DB_NAME parameter.

Assume that the source database prod is on host1 and stores its datafiles in a non-ASM file system. The control files for prod are located in /oracle/oradata/prod/. You want to duplicate the source database to database dupdb on remote host host2. You want to store the duplicate database files in ASM disk group +DISK1.

After connecting RMAN to the target, duplicate, and recovery catalog database, run the RMAN script shown in Example 23–6 to duplicate the database.

Example 23–6 Duplicating from a File System to ASM (Active)

```
DUPLICATE TARGET DATABASE
  TO dupdb
  FROM ACTIVE DATABASE
  SPFILE
    PARAMETER_VALUE_CONVERT '/oracle/oradata/prod/', '+DISK1'
    SET DB_CREATE_FILE_DEST +DISK1;
```

When the DUPLICATE command completes, the duplicate database is created, with datafiles, online redo logs, and control files in ASM disk group +DISK1.

Duplicating a Database from ASM to ASM: Example In this example, you use active database duplication. If the source database uses a server parameter file (or a backup is available), then you can just create a temporary initialization parameter file on the destination host and set only the DB_NAME parameter.

Assume that the source database prod is on host1 and stores its datafiles in ASM disk group +DISK1. You want to duplicate the target to database dupdb on remote host host2. You want to store the datafiles for dupdb in ASM. Specifically, you want to store the datafiles and control files in disk group +DISK2.

In the DUPLICATE command, set PARAMETER_VALUE_CONVERT to convert all directory locations from +DISK1 to +DISK2. The new filenames in +DISK2 are generated by ASM and do not match the original filenames in disk group +DISK1.

After connecting to the target, duplicate, and catalog databases, run the RMAN script shown in Example 23–7 to duplicate the database.

Example 23–7 Duplicating from ASM to ASM (Active)

```
DUPLICATE TARGET DATABASE
  TO dupdb
  FROM ACTIVE DATABASE
  SPFILE PARAMETER_VALUE_CONVERT '+DISK1','+DISK2'
     SET DB_FILE_NAME_CONVERT    '+DISK1','+DISK2'
     SET LOG_FILE_NAME_CONVERT   '+DISK1','+DISK2';
```

When the DUPLICATE command completes, the duplicate database is created, with datafiles, online redo logs, and control files in ASM disk group +DISK2.

Naming Duplicate Files with Alternative Techniques

This section assumes that you do not want to use the SPFILE clause technique for naming duplicate files, or you want to supplement the SPFILE technique with other naming techniques.

The following sections explain how to name the files in the duplicate database:

- Naming Duplicate Control Files
- Naming Duplicate Online Redo Log Files
- Naming Duplicate Datafiles
- Naming Duplicate Tempfiles

Naming Duplicate Control Files

When specifying names for the duplicate database control files, you must set initialization parameters in the auxiliary initialization parameter file. You can specify filenames in the CONTROL_FILES initialization parameter or use other parameters to specify the location for Oracle-managed files.

The rules of precedence for determining the names for the duplicate database control files are the same used by the SQL statement CREATE CONTROLFILE. When choosing names for the duplicate control files, make sure you set the parameters in the initialization parameter file of the auxiliary database correctly; otherwise, you could overwrite the control files of the source database.

> **See Also:** The "Semantics" section in the CREATE CONTROLFILE entry in *Oracle Database SQL Language Reference*

Naming Duplicate Online Redo Log Files

RMAN needs new names for the online redo log files of the duplicate database. You can either specify the names explicitly in the DUPLICATE command, or you can let RMAN generate them.

1. Specify the LOGFILE clause of DUPLICATE command.

   This option creates online redo logs in the duplicate database as specified.

2. Set the LOG_FILE_NAME_CONVERT initialization parameter.

   This parameter creates the duplicate log file names by using string substitution in the names of the source database files, for example, from log_* to duplog_*. Note that you can specify multiple conversion pairs.

   For details on the use of LOG_FILE_NAME_CONVERT with Oracle-managed files, see "Setting Initialization Parameters for Oracle Managed Files" on page 23-14.

   RMAN uses the REUSE parameter when creating the online redo logs. If an online redo log file already exists at the named location and is of the correct size, then it is reused for the duplicate.

3. Set one of the Oracle Managed Files initialization parameters DB_CREATE_FILE_DEST, DB_CREATE_ONLINE_DEST_*n*, or DB_RECOVERY_FILE_DEST.

   This parameter creates the duplicate log file names by using string substitution in the names of the source database files. The rules of precedence among these parameters are the same used by the SQL statement ALTER DATABASE ADD LOGFILE.

4. Do none of the preceding steps.

   Makes the duplicate filenames the same as the filenames from the source database. You must specify the NOFILENAMECHECK option when using this technique. The duplicate database should be in a different host so that the online redo logs of the duplicate do not conflict with the originals.

Rules higher in the order of precedence override rules lower in the list. For example, if you specify both the LOGFILE clause and the LOG_FILE_NAME_CONVERT parameter, then RMAN uses the LOGFILE clause.

Caution:

- If the source database and duplicate database are in the same host, then do not use the name of an online redo log currently in use by the source database.
- If the duplicate database is in a different host and NOFILENAMECHECK is not used, then do not use the name of an online redo log file currently used by any database on the destination host.

Naming Duplicate Datafiles

There are several means of specifying new names to be used for the datafiles of your duplicate database. Listed in order of precedence, they are:

1. Use the RMAN command SET NEWNAME FOR DATAFILE within a RUN block that encloses both the SET NEWNAME commands and the DUPLICATE command.
2. Use the RMAN command CONFIGURE AUXNAME to specify new names for existing datafiles. Run the CONFIGURE AUXNAME command before the DUPLICATE command.
3. Execute DUPLICATE ... DB_FILE_NAME_CONVERT to specify a rule for converting filenames for any datafiles not renamed with SET NEWNAME or CONFIGURE AUXNAME.

   Note: You cannot use the DB_FILE_NAME_CONVERT clause of the DUPLICATE command to control generation of new names for files at the duplicate instance which are Oracle Managed Files (OMF) at the source database instance. See *Oracle Database Backup and Recovery Reference* for details on this restriction.

4. Set the DB_FILE_NAME_CONVERT initialization parameter.

   Note: You cannot use the DB_FILE_NAME_CONVERT initialization parameter to control generation of new names for files at the duplicate instance which are Oracle Managed Files (OMF) at the source database instance. It is subject to the same semantics and limitations as the DB_FILE_NAME_CONVERT parameter to the DUPLICATE command. It See *Oracle Database Backup and Recovery Reference* for details .

5. Set the DB_CREATE_FILE_DEST initialization parameter to create Oracle Managed Files datafiles at the specified location.

If you do not use any of the preceding options, then the duplicate database reuses the original datafile locations from the source database.

Naming Duplicate Tempfiles

RMAN re-creates datafiles for temporary tablespaces as part of the process of duplicating a database. There are several means of specifying locations for duplicate database tempfiles. Listed in order of precedence, they are:

1. Use the `SET NEWNAME FOR TEMPFILE` command within a `RUN` block that encloses both the `SET NEWNAME` commands and the `DUPLICATE` command.
2. Specify the `DB_FILE_NAME_CONVERT` clause to the `DUPLICATE` command to specify a rule for converting tempfiles not renamed with `SET NEWNAME`.

   Note: You cannot use the `DB_FILE_NAME_CONVERT` clause to control generation of new names for files at the duplicate instance which are Oracle Managed Files (OMF) at the source database instance. See *Oracle Database Backup and Recovery Reference* for details on this restriction.

3. Set the `DB_FILE_NAME_CONVERT` initialization parameter.

   Note: The `DB_FILE_NAME_CONVERT` initialization parameter is subject to the same semantics and limitations as the `DB_FILE_NAME_CONVERT` parameter to the `DUPLICATE` command.See *Oracle Database Backup and Recovery Reference* for details.

4. Set the `DB_CREATE_FILE_DEST` initialization parameter to create Oracle Managed Files tempfiles.

RMAN Duplication Scenarios

This section presents several representative scenarios for using RMAN `DUPLICATE`:

- Duplicating a Subset of the Source Database Tablespaces
- Using DUPLICATE to Restore an Archival Backup
- Using SET NEWNAME to Name Duplicate Files
- Using CONFIGURE AUXNAME to Name Duplicate Files

Duplicating a Subset of the Source Database Tablespaces

It is not always necessary to duplicate all tablespaces of a database. For example, you may plan to generate reports at the duplicate that require only a subset of tablespaces from your source database. The `DUPLICATE DATABASE` command has options that enable you to skip read-only tablespaces or tablespaces that are currently `OFFLINE NORMAL`. You can also use the `TABLESPACE` clause to specify which tablespaces to include in the duplicate database. This section contains the following topics:

- Excluding Read-Only Tablespaces
- Excluding OFFLINE NORMAL Tablespaces
- Including and Excluding Specified Tablespaces

Excluding Read-Only Tablespaces

When you specify SKIP READONLY on the DUPLICATE command, RMAN does not duplicate the datafiles of read-only tablespaces. Example 23–8 is a variation of Example 23–3 except with read-only tablespace excluded.

Example 23–8 Excluding Read-Only Tablespaces

```
DUPLICATE TARGET DATABASE
  TO dupdb
  FROM ACTIVE DATABASE
  SKIP READONLY
  NOFILENAMECHECK;
```

Excluding OFFLINE NORMAL Tablespaces

When tablespaces are taken offline with the OFFLINE NORMAL option before a DUPLICATE operation, RMAN does not duplicate their datafiles and issues DROP TABLESPACE for these tablespaces on the duplicate database. Thus, you do not need to specify special syntax to exclude these tablespaces. After duplication, all datafiles and tablespaces are online.

> **Note:** RMAN duplicates tablespaces that are taken offline with the IMMEDIATE option because they require recovery. As with online tablespaces, RMAN requires a valid backup for these tablespaces when you use the backup-based duplication technique.

Including and Excluding Specified Tablespaces

You can use the SKIP TABLESPACE parameter to exclude specified tablespaces from the duplicate database. Note that you cannot exclude the SYSTEM and SYSAUX tablespaces, undo tablespaces, and tablespaces with rollback segments. You can use the TABLESPACE parameter to specify which tablespaces should be included in the specified database. Unlike SKIP TABLESPACE, which specifies which tablespaces should be excluded from the duplicate database, this option specified which tablespaces should be included and then skips the remaining tablespaces.

Example 23–9 is a variation of Example 23–3 except with the tools tablespace excluded.

Example 23–9 Excluding Specified Tablespaces

```
DUPLICATE TARGET DATABASE
  TO dupdb
  FROM ACTIVE DATABASE
  SKIP TABLESPACE tools
  NOFILENAMECHECK;
```

Example 23–10 is a variation of Example 23–3 except with the users tablespace included and all other tablespaces excluded, with the exception of the SYSTEM and SYSAUX tablespaces and tablespaces with undo or rollback segments.

Example 23–10 Including Specified Tablespaces

```
DUPLICATE TARGET DATABASE
  TO dupdb
  FROM ACTIVE DATABASE
  TABLESPACE users
  NOFILENAMECHECK;
```

Using DUPLICATE to Restore an Archival Backup

As explained in "Making Database Backups for Long-Term Storage" on page 8-23, you can make an **archival backup** that is all-inclusive in the sense that every file needed to restore and recover the database is included. The recommended technique for restoring an archival backup for testing is to create a temporary instance and use the DUPLICATE command. In this way, you avoid interfering with the production system.

Assume that you start RMAN and connect to a target and catalog database. Example 23–11 makes an archival backup on a temporary disk with the tag TESTDB.

Example 23–11 Creating a Temporary Archival Backup

```
BACKUP DATABASE
  FORMAT '/disk1/oraclebck/%U'
  TAG TESTDB
  KEEP UNTIL 'SYSDATE+1'
  RESTORE POINT TESTDB06;
```

Example 23–11 creates a restore point, which is a label for the time to which the backup should be recovered, that exists both in the recovery catalog and in the backup control file. Note that archived logs are only backed up if this is an online backup. Archived logs are not needed for offline backups and as such are not backed up.

The procedure for restoring the backup created in Example 23–11 is the same as for "Duplicating a Database" on page 23-11. The only additional requirement is that in the DUPLICATE command you must specify the restore point that was created with the archival backup.

To restore an archival backup:

1. Connect RMAN to the source database as TARGET, the duplicate database instance as AUXILIARY, and recovery catalog.
2. Run the LIST RESTORE POINT to display the available restore points (see "Listing Restore Points" on page 10-9 for instructions).

   ```
   LIST RESTORE POINT;
   ```

3. Follow all the steps for duplicating database up to (but not including) the issuing of the DUPLICATE command itself.
4. Execute the DUPLICATE command, specifying the restore point of the archival backup that you intend to restore.

   The following example assumes that you have created an auxiliary instance and are running the RMAN client on the test host. The UNTIL RESTORE POINT clause specifies TESTDB06, which is the restore point created in Example 23–11.

   ```
   DUPLICATE TARGET DATABASE
     TO mytest
     UNTIL RESTORE POINT TESTDB06
     DB_FILE_NAME_CONVERT '/prod/oracledb/','/test/oracledb'
     PFILE 'test/oracledb/init.ora';
   ```

   The preceding DUPLICATE command restores the whole database and renames it to mytest. The DUPLICATE command does not restore the original control file, but instead creates a new control file. Thus, you can only specify DUPLICATE ... UNTIL RESTORE POINT if RMAN is connected to a catalog or connected to the source database when the restore point still exists in the control file.

Using SET NEWNAME to Name Duplicate Files

In this scenario, you create a duplicate database by using backup-based duplication. The source database does not use a server parameter file, so you cannot use the SPFILE technique to specify names for the duplicate datafiles. You decide to use SET NEWNAME commands to specify the filenames because the duplicate datafiles will be spread out across several directories.

Assume that the source database prod is on host1 and contains eight datafiles, which are spread out over multiple directories. You want to duplicate the source database to database dupdb on remote host host2. The dupdb database should exclude tablespace tools, but keep all of the other tablespaces.

In this scenario, host1 and host2 have different directory structures. You want to store the datafiles in host2 in the /oradata1 through /oradata7 subdirectories. Although eight datafile exist in the source database, you only need to specify seven locations for the target datafiles because you are excluding the tools tablespace. You want to create two online redo log groups, each with two members of size 200 KB, in the directory /duplogs on the destination host. Assume that host1 and host2 cannot mount each other's file systems by any means such as NFS.

You have disk copies or backup sets stored on disk for all the datafiles and archived redo logs in the source database, and you have manually copied them to host2 by means of an operating system utility. These backups and copies exist in the same location on host2 as they do in host1.

You use an operating system utility to copy the initialization parameter file from host1 to an appropriate location in host2. You have reset all initialization parameters that end in _DEST and specify a path name. You do not set DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT because you are specifying names for datafiles and online logs in the RUN command itself. The auxiliary instance uses a server-side initialization parameter file in the default location so the PFILE parameter is not necessary on the DUPLICATE command.

Example 23–12 illustrates a script that creates the duplicate database. Note that a RUN command is necessary because you can only execute SET NEWNAME within RUN. You do not set a new name for datafile 7 because it is in the tools tablespace, which you are excluding from the duplicate database.

Example 23–12 Duplicating with SET NEWNAME

```
RUN
{
  SET NEWNAME FOR DATAFILE 1 TO '/oradata1/system01.dbf';
  SET NEWNAME FOR DATAFILE 2 TO '/oradata2/undotbs01.dbf';
  SET NEWNAME FOR DATAFILE 3 TO '/oradata3/cwmlite01.dbf';
  SET NEWNAME FOR DATAFILE 4 TO '/oradata4/drsys01';
  SET NEWNAME FOR DATAFILE 5 TO '/oradata5/example01.dbf';
  SET NEWNAME FOR DATAFILE 6 TO '/oradata6/indx01.dbf';
  # Do not set a newname for datafile 7, because it is in the tools tablespace,
  # and you are excluding tools from the duplicate database.
  SET NEWNAME FOR DATAFILE 8 TO '/oradata7/users01.dbf';
  DUPLICATE TARGET DATABASE TO dupdb
    SKIP TABLESPACE tools
    LOGFILE
      GROUP 1 ('/duplogs/redo01a.log',
               '/duplogs/redo01b.log') SIZE 4M REUSE,
      GROUP 2 ('/duplogs/redo02a.log',
               '/duplogs/redo02b.log') SIZE 4M REUSE;
}
```

Using SET NEWNAME to Name Oracle-Managed Files

To store specific datafiles or tempfiles in an Oracle-managed files destination that is independent of the locations of the rest of the database files, use the following steps:

- Set the DB_CREATE_FILE_DEST initialization parameter at the auxiliary instance to the desired location
- Enclose the DUPLICATE command in a RUN block and use SET NEWNAME FOR DATAFILE ... TO NEW and SET NEWNAME FOR TEMPFILE ... TO NEW

The specified datafiles or tempfiles are created with Oracle-managed file names in the location specified by DB_CREATE_FILE_DEST.

As shown in Example 23–13, you can also use SET NEWNAME to direct individual datafiles or tempfiles to a specific ASM disk group.

Example 23–13 Using SET NEWNAME to Create Files in an ASM Disk Group

```
RUN
{
   SET NEWNAME FOR DATAFILE 1 TO "+dgroup1";
   SET NEWNAME FOR DATAFILE 2 TO "+dgroup2";
   .
   .
   .
   DUPLICATE TARGET DATABASE
     TO dupdb
     FROM ACTIVE DATABASE
     SPFILE SET DB_CREATE_FILE_DEST +dgroup3;
}
```

> **See also:** *Oracle Database Backup and Recovery Reference* for details on using SET NEWNAME

Using CONFIGURE AUXNAME to Name Duplicate Files

This section assumes the same circumstances described in "Using SET NEWNAME to Name Duplicate Files" on page 23-22. Example 23–14 is a variation of Example 23–12 that uses CONFIGURE AUXNAME to specify the new datafile names. These new filenames are recorded in the control file and used every time you perform the duplication in the future.

Example 23–14 also uses automatic channels and a client-side initialization parameter file for the database duplication, and uses the LOGFILE clause to specify names and sizes for the online redo logs. In this case the RUN command is not necessary because you are not using SET NEWNAME.

Example 23–14 Using CONFIGURE AUXNAME to Generate Database Filenames

```
CONFIGURE AUXNAME FOR DATAFILE 1 TO '/oradata1/system01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 2 TO '/oradata2/undotbs01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 3 TO '/oradata3/cwmlite01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 4 TO '/oradata4/drsys01';
CONFIGURE AUXNAME FOR DATAFILE 5 TO '/oradata5/example01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 6 TO '/oradata6/indx01.dbf';

DUPLICATE TARGET DATABASE
  TO dupdb
  SKIP TABLESPACE tools
  LOGFILE
```

```
    GROUP 1 ('/duplogs/redo01a.log',
             '/duplogs/redo01b.log') SIZE 4M REUSE,
    GROUP 2 ('/duplogs/redo02a.log',
             '/duplogs/redo02b.log') SIZE 4M REUSE;
```

RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery and then opens the database with the `RESETLOGS` option to create the online redo logs.

After the duplication is complete, you can clear the configured auxiliary names for the datafiles in the duplicate database, so that they are not overwritten by future operations. For example, enter the following commands:

```
CONFIGURE AUXNAME FOR DATAFILE 1 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 2 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 3 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 4 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 5 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 6 CLEAR;
```

Alternatively, you may want to periodically synchronize a duplicate database with the original database that was duplicated. In this case, you can run the `DUPLICATE` command again, essentially re-creating the duplicate database. This technique requires making complete copies of the datafiles of the duplicate database. Run the script in Example 23–15 whenever you want to synchronize the duplicate with the source database. For example, you might run the script daily or weekly.

Example 23–15 Reduplicating the Database

```
DUPLICATE TARGET DATABASE TO dupdb
SKIP TABLESPACE tools
  LOGFILE
    GROUP 1 ('/duplogs/redo01a.log',
             '/duplogs/redo01b.log') SIZE 4M REUSE,
    GROUP 2 ('/duplogs/redo02a.log',
             '/duplogs/redo02b.log') SIZE 4M REUSE;
```

24

Creating Transportable Tablespace Sets

This chapter explains how to use RMAN to create transportable tablespace sets by restoring backups. This discussion assumes that you are familiar the **transportable tablespace** procedure described in *Oracle Database Administrator's Guide*. The procedure in this chapter is an alternative technique for generating transportable tablespace sets.

This chapter contains the following sections:

- Overview of Creating Transportable Tablespace Sets
- Customizing Initialization Parameters for the Auxiliary Instance
- Creating a Transportable Tablespace Set
- Troubleshooting Creation of Transportable Tablespace Sets
- Transportable Tablespace Set Scenarios

Overview of Creating Transportable Tablespace Sets

This section explains the basic concepts and tasks involved in creating transportable tablespace sets from RMAN backups.

Purpose of Creating Transportable Tablespace Sets

A **transportable tablespace set** contains datafiles for a set of tablespaces and an export file containing structural metadata for the set of tablespaces. The export file is generated by Data Pump Export.

One use of transportable tablespace sets is to create a tablespace repository. For example, if you have a database with some tablespaces used for quarterly reporting, you can create transportable sets for these tablespaces for storage in a tablespace repository. Subsequently, versions of the tablespace can be requested from the repository and attached to another database for use in generating reports.

Another use for transportable tablespaces is in an Oracle Streams environment. When preparing to use Oracle Streams to keep a destination database synchronized with a source database, you must perform Oracle Streams instantiation. You must bring the destination database up to a given SCN at which the two databases were known to be synchronized before you can actually use Oracle Streams to move subsequent updates from the source database to the destination database. You can create transportable tablespace sets from backups as part of the Oracle Streams instantiation.

A key benefit of the RMAN `TRANSPORT TABLESPACE` command is that it does not need access to the live datafiles from the tablespaces to be transported. In contrast, the transportable tablespace technique described in *Oracle Database Administrator's Guide*

requires that the tablespaces to be transported are open read-only during the transport. Thus, transporting from backups improves database availability, especially for large tablespaces, because the tablespaces to be transported can remain open for writes during the operation. Also, placing a tablespace in read-only mode can take a long time, depending on current database activity.

The RMAN `TRANSPORT TABLESPACE` command also enables you to specify a target point in time, SCN, or restore point during your recovery window and transport tablespace data as it existed at that time (see "Creating a Transportable Tablespace Set at a Specified Time or SCN" on page 24-8). For example, if your **backup retention policy** guarantees a one week recovery window, and if you want to create transportable tablespaces based on the contents of the database on the last day of the month, then RMAN can perform this task at any time during the first week of the next month.

See also:

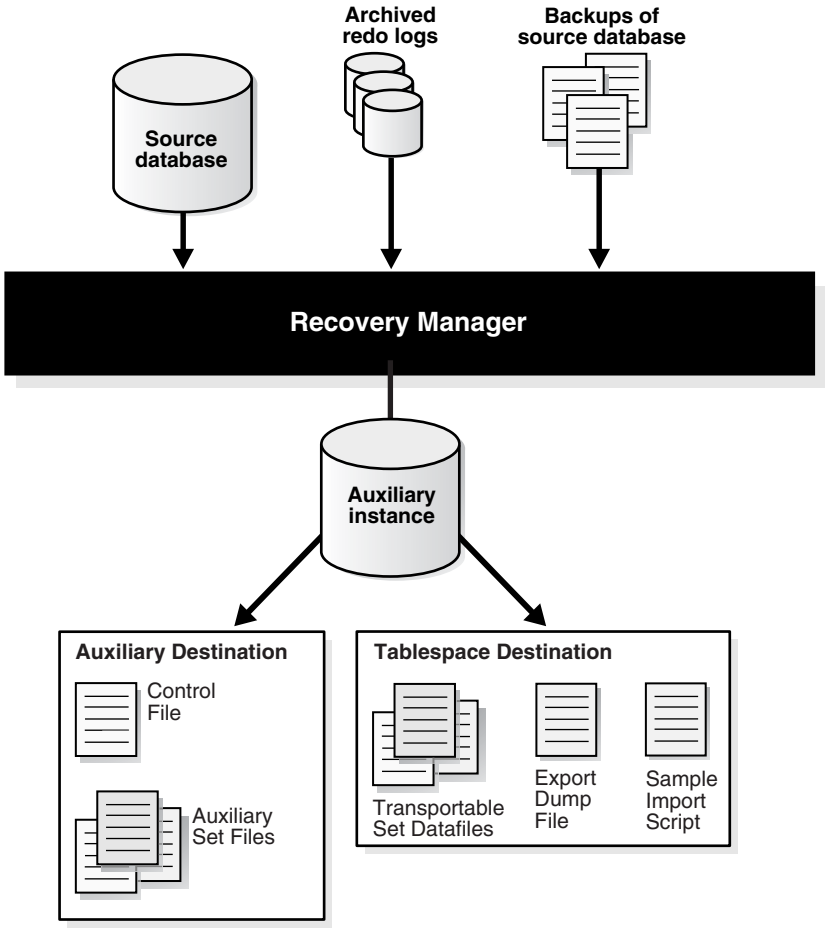
- *Oracle Database Backup and Recovery Reference* for reference information on the `TRANSPORT TABLESPACE` command
- *Oracle Streams Replication Administrator's Guide* for more details on RMAN and tablespace repositories
- *Oracle Streams Replication Administrator's Guide* for more details on RMAN and Oracle Streams instantiations

Basic Concepts of Transportable Tablespace Sets

You create a transportable tablespace set by connecting RMAN to a source database as `TARGET` and then executing the `TRANSPORT TABLESPACE` command. The source database contains the tablespaces to be transported.

You must have a backup of all needed tablespaces and archived redo log files available for use by RMAN that can be recovered to the target point in time for the `TRANSPORT TABLESPACE` operation. Figure 24–1 illustrates the basic process of transportable tablespace creation.

Figure 24–1 RMAN Transportable Tablespace From Backup: Architecture



The process shown in Figure 24–1 occurs in the following phases:

1. RMAN starts an **auxiliary instance**.

   An auxiliary instance is created by RMAN on the same host as the source database to perform the restore and recovery of the tablespaces. RMAN automatically creates an initialization parameter file for the auxiliary instance and starts it `NOMOUNT`.

2. RMAN restores a backup of the source database control file to serve as the auxiliary instance control file and mounts this control file.

3. RMAN restores **auxiliary set** and transportable set datafiles from the backups of the source database.

   The **auxiliary set** includes datafiles and other files required for the tablespace transport but which are not themselves part of the transportable tablespace set. The auxiliary set typically includes the `SYSTEM` and `SYSAUX` tablespaces, tempfiles, and datafiles containing rollback or undo segments. The auxiliary instance has other files associated with it, such as its own control file, parameter file, and online logs, but they are not part of the auxiliary set.

   RMAN stores the auxiliary datafiles in the selected **auxiliary destination**. The auxiliary destination is a disk location where RMAN can store auxiliary set files such as the parameter file, datafiles (other than those in the transportable set),

control files, and online logs of the auxiliary instance during the transport. If the transport succeeds, then RMAN deletes these files.

RMAN stores the transportable set files in the **tablespace destination**. The tablespace destination is a disk location that by default contains the datafile copies and other output files when the tablespace transport command completes.

4. RMAN performs **database point-in-time recovery (DBPITR)** at the auxiliary instance.

   The recovery updates auxiliary and transportable set datafiles to their contents as of the target time specified for the `TRANSPORT DATABASE` command. If no target time is specified, then RMAN recovers with all available redo. RMAN restores archived redo logs from backup as necessary at the auxiliary destination (or other location) and deletes them after they are applied.

5. RMAN opens the auxiliary database with the `RESETLOGS` options.

   The datafiles now reflect the tablespace contents as of the target SCN for the tablespace transport operation.

6. RMAN places the transportable set tablespaces of the auxiliary instance into read-only mode. RMAN also invokes Data Pump Export in transportable tablespace mode to create the **export dump file** for the transportable set.

   By default, the dump file is located in the tablespace destination. To specify the dump file location, see "Specifying Locations for Data Pump Files" on page 24-9.

   RMAN also generates the sample Data Pump import script for use when plugging in the transported tablespaces at a target database. The contents of this script are written to a file named `impscript.sql` in the tablespace destination. The commands for the script are also included in the RMAN command output.

7. If the preceding steps are successful, then RMAN shuts down the auxiliary instance and deletes all files created during `TRANSPORT TABLESPACE` except for the transportable set files, the Data Pump Export file, and the sample import script.

Basic Steps of Creating Transportable Tablespace Sets

Before creating transportable tablespace sets you must meet a number of prerequisites. These prerequisites are described in the `TRANSPORT TABLESPACE` entry in *Oracle Database Backup and Recovery Reference*.

The basic steps are as follows:

1. Start the RMAN client and connect to the source database and, if used, the recovery catalog.
2. If necessary, set additional parameters in the auxiliary instance parameter file.

   This task is described in "Customizing Initialization Parameters for the Auxiliary Instance" on page 24-5.

3. Execute the `TRANSPORT TABLESPACE` command.

   This basic technique is described in "Creating a Transportable Tablespace Set" on page 24-7. Variations on this technique are described in "Transportable Tablespace Set Scenarios" on page 24-8.

4. If the `TRANSPORT TABLESPACE` command fails, troubleshoot the problem and then retry the command until it succeeds.

This technique is described in "Troubleshooting Creation of Transportable Tablespace Sets" on page 24-8.

5. Return to the procedure for transporting tablespaces described in *Oracle Database Administrator's Guide*.

Customizing Initialization Parameters for the Auxiliary Instance

When RMAN creates the auxiliary instance, it creates an initialization parameter file. The default values should work for nearly all `TRANSPORT TABLESPACE` cases, especially if you specify `AUXILIARY DESTINATION` on `TRANSPORT TABLESPACE`.

RMAN can also use an auxiliary instance parameter file that contains values for additional initialization parameters. These values override the values of parameters defined in the default initialization parameter file. You might use an auxiliary instance parameter file for the following reasons:

- To increase `STREAMS_POOL_SIZE` and `SHARED_POOL_SIZE` if needed for Data Pump Export.
- To manage locations for auxiliary instance datafiles (see "Using Initialization Parameters to Name Auxiliary Files" on page 24-11). For example, you do not want all auxiliary instance datafiles stored in the same location on disk, but you do not want to specify the location of every file individually.
- To specify names for online redo logs with `LOG_FILE_NAME_CONVERT` (see "Using Initialization Parameters to Name Auxiliary Files" on page 24-11).

The auxiliary instance parameter file is not intended to be a complete initialization parameter file for the auxiliary instance. Any parameters specified are added to or override the default parameters for the auxiliary instance. It is not necessary to specify parameters in the initialization file that you do not intend to override.

Setting Initialization Parameters for the Auxiliary Instance

RMAN defines the basic initialization parameters in Table 24–1 for the automatic auxiliary instance.

Table 24–1 Default Initialization Parameters

Initialization Parameter	Value
`DB_NAME`	Same as `DB_NAME` of the source database.
`COMPATIBLE`	Same as the compatible setting of the source database.
`DB_UNIQUE_NAME`	Generated unique value based on `DB_NAME`.
`DB_BLOCK_SIZE`	Same as the `DB_BLOCK_SIZE` of the source database.
`DB_FILES`	Same value as `DB_FILES` for the source database
`SHARED_POOL_SIZE`	110 MB. Data Pump Export can require this much space or more. If the default shared pool size is not sufficient for Data Pump Export to run successfully, then specify a larger value. **See Also:** *Oracle Database Reference* for more information on the use of the `SHARED_POOL_SIZE` initialization parameter
`LARGE_POOL_SIZE`	1 MB
`DB_CREATE_FILE_DEST`	Auxiliary destination (only if the `AUXILIARY DESTINATION` argument to `TRANSPORT TABLESPACE` is set). RMAN creates Oracle-managed control files and online logs in this location.

Table 24–1 (Cont.) Default Initialization Parameters

Initialization Parameter	Value
CONTROL_FILES	A generated filename in the auxiliary destination (only if the AUXILIARY DESTINATION argument to TRANSPORT TABLESPACE is set). RMAN creates control files in this location. By default, RMAN creates one control file for the auxiliary instance in an operating system-specific location. In Linux and UNIX, the default location is ?/dbs/cntrl_@.dbf, where ? stands for ORACLE_HOME and @ stands for ORACLE_SID. For an automatic auxiliary instance, RMAN randomly generates the ORACLE_SID. **See Also:** *Oracle Database Reference* for more on the use of the CONTROL_FILES initialization parameter

Overriding one of the basic initialization parameters in Table 24–1 with an inappropriate value in the auxiliary instance parameter file can cause TRANSPORT TABLESPACE to fail. If you encounter a problem, then try returning the initialization parameter to its default value.

> **See Also:** "Using Initialization Parameters to Name Auxiliary Files" on page 24-11 to learn how to use DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT to name files

Setting the Location of the Auxiliary Instance Parameter File

By default, RMAN looks for the auxiliary initialization parameter file at an operating system-dependent location on the host running the RMAN client. Note that this location may not be on the host running the auxiliary instance. For UNIX, this location is ?/rdbms/admin/params_auxint.ora, where question mark (?) stands for ORACLE_HOME on the host running RMAN. If no file is found in the default location, then RMAN does not generate an error.

If you use the default initialization parameters for the auxiliary instance, then check whether an auxiliary instance parameter file exists before running TRANSPORT TABLESPACE.

You can use the RMAN SET AUXILIARY INSTANCE PARAMETER FILE command in a RUN block before TRANSPORT TABLESPACE to specify a different location for the auxiliary instance parameter file. As with the default location of the auxiliary instance parameter file, the path specified when using SET AUXILIARY INSTANCE PARAMETER FILE is a client-side path.

Assume you create a file named /tmp/auxinstparams.ora on the host running the RMAN client. This file contains the following initialization parameter:

```
SHARED_POOL_SIZE=150M;
```

You can then use the initialization parameter file with TRANSPORT TABLESPACE as shown in Example 24–1. The SHARED_POOL_SIZE parameter in /tmp/auxinstparams.ora overrides the default value used for SHARED_POOL_SIZE when RMAN creates the auxiliary instance.

Example 24–1 Specifying an Auxiliary Instance Parameter File

```
RUN
{
  SET AUXILIARY INSTANCE PARAMETER FILE TO '/tmp/auxinstparams.ora';
  TRANSPORT TABLESPACE tbs_2
```

```
    TABLESPACE DESTINATION '/disk1/transportdest'
    AUXILIARY DESTINATION '/disk1/auxdest';
}
```

Creating a Transportable Tablespace Set

This section describes the use of TRANSPORT TABLESPACE in the most basic and automated case. Refer to "Transportable Tablespace Set Scenarios" on page 24-8 for variations on the basic case.

It is assumed that you have met the prerequisites described in the TRANSPORT TABLESPACE entry in *Oracle Database Backup and Recovery Reference*. It also assumed that you have met the requirements described in *Oracle Database Administrator's Guide*:

- Confirmed that tablespace transport is supported between your source and destination platforms
- Identified a self-contained set of tablespaces to include in the transportable set

To create a transportable tablespace set:

1. Start the RMAN client and connect to the source database and, if used, the recovery catalog database.
2. Run the TRANSPORT TABLESPACE command in RMAN.

   In the most basic case, you specify an AUXILIARY DESTINATION clause, which is optional but recommended. RMAN uses default values that work for most cases. If you do not specify an auxiliary location, then ensure that locations are specified for all auxiliary instance files. See the rules described in "Specifying Auxiliary File Locations" on page 24-10 to learn how to name auxiliary files.

   Example 24–2 creates a transportable tablespace set that includes tablespaces tbs_2 and tbs_3.

Example 24–2 Creating a Transportable Tablespace Set

```
TRANSPORT TABLESPACE tbs_2, tbs_3
   TABLESPACE DESTINATION '/disk1/transportdest'
   AUXILIARY DESTINATION '/disk1/auxdest';
```

   After the command completes successfully, note the following results:

   - The transportable set datafiles are left in the location /disk1/transportdest with their original names. Note that the transportable tablespace set datafiles are not automatically converted to the endian format of the destination database by TRANSPORT TABLESPACE. If necessary, use the RMAN CONVERT command to convert the datafiles to the endian format of the destination database after creating the transportable set.
   - The Data Pump Export dump file for the transportable set is named dmpfile.dmp, the export log is named explog.log, and the sample import script is named impscrpt.sql.

     All files are created in /disk1/transportdest. If a file under the name of the export dump file already exists in the tablespace destination, then TRANSPORT TABLESPACE fails when it calls Data Pump Export. If you are repeating a previous TRANSPORT TABLESPACE operation, then make sure to delete the previous output files, including the export dump file.
   - The auxiliary set files are removed from /disk1/auxdest.

3. If necessary, edit the sample import script.

   The sample import script assumes that the files used to import the tablespaces into the destination database are stored in the same locations where they were created by TRANSPORT TABLESPACE. If files have been moved to new disk locations before being plugged in, then you must update the sample script with the new locations of the files before using the script to plug in the transported tablespaces.

4. Return to the process for transporting tablespaces described in *Oracle Database Administrator's Guide*.

Troubleshooting Creation of Transportable Tablespace Sets

When the RMAN TRANSPORT TABLESPACE command fails, the failed auxiliary instance files are left intact in the auxiliary instance destination for troubleshooting.

If you see an error related to shared pool size, then create an auxiliary instance parameter file with larger values for STREAMS_POOL_SIZE and SHARED_POOL_SIZE. This technique is described in "Setting Initialization Parameters for the Auxiliary Instance" on page 24-5.

If your SET NEWNAME, CONFIGURE AUXNAME and DB_FILE_NAME_CONVERT cause multiple files in the auxiliary or transportable tablespace sets to have the same name, then RMAN will report an error during the TRANSPORT TABLESPACE command. To correct the problem, use different values for these parameters to ensure that duplicate file names are not created. Naming techniques are described in "Specifying Auxiliary File Locations" on page 24-10.

Transportable Tablespace Set Scenarios

This section contains the following topics:

- Creating a Transportable Tablespace Set at a Specified Time or SCN
- Specifying Locations for Data Pump Files
- Specifying Auxiliary File Locations

Creating a Transportable Tablespace Set at a Specified Time or SCN

You can specify a target time or SCN with the TRANSPORT TABLESPACE command. During the tablespace transport operation, RMAN restores the tablespace at the auxiliary instance with backups from before the target time and performs point-in-time recovery on the auxiliary database to the specified target time. Backups and archived redo logs needed for this point-in-time recovery must be available.

You can specify the target time with an SCN (in the current incarnation or its ancestors) as shown in Example 24–3.

Example 24–3 Specifying an End SCN

```
TRANSPORT TABLESPACE tbs_2
   TABLESPACE DESTINATION '/disk1/transportdest'
   AUXILIARY DESTINATION '/disk1/auxdest'
   UNTIL SCN 11379;
```

You can also specify a restore point as shown in Example 24–4.

Example 24–4 Specifying an End Restore Point

```
TRANSPORT TABLESPACE tbs_2
   TABLESPACE DESTINATION '/disk1/transportdest'
   AUXILIARY DESTINATION '/disk1/auxdest'
   UNTIL RESTORE POINT 'before_upgrade';
```

You can also specify an end time as shown in Example 24–5.

Example 24–5 Specifying an End Time

```
TRANSPORT TABLESPACE tbs_2
   TABLESPACE DESTINATION '/disk1/transportdest'
   AUXILIARY DESTINATION '/disk1/auxdest'
   UNTIL TIME 'SYSDATE-1';
```

Specifying Locations for Data Pump Files

You can change the names of the Data Pump Export dump file for the transportable set, the sample import script for use at the target database, the log file generated by Data Pump Export, and the directory to which they are written.

By default, these files are stored in the tablespace destination and named as follows:

- The Data Pump Export dump file is named `dmpfile.dmp`.
- The export log file is named `explog.log`.
- The sample import script is named `impscrpt.sql`.

You can place the dump file and the export log in a different directory by using the `DATAPUMP DIRECTORY` clause, passing in the name of a database directory object. Note that the database directory object used by `DATAPUMP DIRECTORY` is not the directory path of an actual file system directory. The value passed corresponds to the `DIRECTORY` command line argument of Data Pump Export. See *Oracle Database Utilities* for more details on the use of directory objects with Data Pump Export.

You can rename these files with the `DUMP FILE`, `EXPORT LOG`, and `IMPORT SCRIPT` clauses of `TRANSPORT TABLESPACE`. The filenames cannot contain full file paths with directory names. If the `DUMP FILE` or `EXPORT LOG` filenames specify file paths, then `TRANSPORT TABLESPACE` fails when it attempts to generate the export dump files. Use the `DATAPUMP DIRECTORY` clause to specify a database directory object that identifies a location for the outputs of Data Pump Export.

The following scenario illustrates the use of `TRANSPORT TABLESPACE` with the `DATAPUMP DIRECTORY`, `DUMP FILE`, `EXPORT LOG` and `IMPORT SCRIPT` filenames specified. Assume that you create a database directory object as follows for use with Data Pump Export:

```
CREATE OR REPLACE DIRECTORY mypumpdir as '/datapumpdest';
```

Example 24–6 shows a `TRANSPORT TABLESPACE` command with optional arguments that specify output file locations.

Example 24–6 Specifying Output File Locations

```
TRANSPORT TABLESPACE tbs_2
   TABLESPACE DESTINATION '/transportdest'
   AUXILIARY DESTINATION '/auxdest'
   DATAPUMP DIRECTORY  mypumpdir
   DUMP FILE 'mydumpfile.dmp'
   IMPORT SCRIPT 'myimportscript.sql'
```

```
  EXPORT LOG 'myexportlog.log';
```

After a successful run, RMAN cleans up the auxiliary destination, creates the Data Pump Export Dump file and the export log in the directory referenced by DATAPUMP DIRECTORY (/datapumpdest/mydumpfile.dmp and /datapumpdest/myexportlog.log), and stores the transportable set datafiles in /transportdest.

Specifying Auxiliary File Locations

Several rules affect the location of auxiliary instance files created during the transport. The simplest technique is to use AUXILIARY DESTINATION and let RMAN manage all file locations automatically. To relocate some or all auxiliary instance files, the following options for specifying file locations appear in order of precedence:

1. SET NEWNAME

   As described in "Using SET NEWNAME for Auxiliary Datafiles" on page 24-10, you can use this command to specify names for datafiles.

2. CONFIGURE AUXNAME

   As described in "Using CONFIGURE AUXNAME for Auxiliary Datafiles" on page 24-11, you can use this command to specify names for datafiles.

3. AUXILIARY DESTINATION

   As described in "Using AUXILIARY DESTINATION to Specify a Location for Auxiliary Files" on page 24-11, you can use this command to specify a location for auxiliary files.

4. LOG_FILE_NAME_CONVERT and DB_FILE_NAME_CONVERT in the initialization parameter file

   As described in "Using Initialization Parameters to Name Auxiliary Files" on page 24-11, you can use this command to specify a location for auxiliary files.

If you use several rules, then the first rule in the list that applies to a file determines the filename.

Using SET NEWNAME for Auxiliary Datafiles

You can use the SET NEWNAME command in a RUN block to specify filenames for use in TRANSPORT TABLESPACE. The SET NEWNAME commands shown in Example 24–7 cause these auxiliary instance datafiles to be restored to the locations named instead of /disk1/auxdest.

Example 24–7 Using SET NEWNAME to Name Auxiliary Datafiles

```
RUN
{
  SET NEWNAME FOR DATAFILE '/oracle/dbs/tbs_12.f'
    TO '/bigdrive/auxdest/tbs_12.f';
  SET NEWNAME FOR DATAFILE '/oracle/dbs/tbs_11.f'
    TO '/bigdrive/auxdest/tbs_11.f';
  TRANSPORT TABLESPACE tbs_2
    TABLESPACE DESTINATION '/disk1/transportdest'
    AUXILIARY DESTINATION '/disk1/auxdest';
}
```

SET NEWNAME is best used with one-time operations. If you expect to create transportable tablespaces from backup regularly for a particular set of tablespaces,

then consider using CONFIGURE AUXNAME instead of SET NEWNAME in order to make persistent settings for the location of the auxiliary instance datafiles.

Using CONFIGURE AUXNAME for Auxiliary Datafiles

You can use the CONFIGURE AUXNAME command to specify persistent locations for transportable tablespace set or auxiliary set datafiles. RMAN restores each datafile for which a CONFIGURE AUXNAME command has been used to the specified location before recovery. RMAN deletes auxiliary set datafiles when the operation is complete, unless the operation failed.

An example illustrates the relationship between CONFIGURE AUXNAME and TRANSPORT ... AUXILIARY DESTINATION. Suppose you want to transport tablespace tbs_11. The tablespace tbs_12, which contains datafile tbs_12.f, is part of the auxiliary set. You execute the following steps:

1. You use the CONFIGURE AUXNAME statement to set a persistent nondefault location for the auxiliary set datafile /oracle/dbs/tbs_12.f.

   For example, you enter the following command:

   ```
   CONFIGURE AUXNAME FOR '/oracle/dbs/tbs_12.f'
                     TO '/disk1/auxdest/tbs_12.f';
   ```

2. You execute the TRANSPORT TABLESPACE command with the AUXILIARY DESTINATION parameter.

   For example, you enter the following command:

   ```
   TRANSPORT TABLESPACE tbs_11
     AUXILIARY DESTINATION '/myauxdest';
   ```

In the preceding scenario, RMAN restores the auxiliary set copy of datafile /oracle/dbs/tbs_12.f to /disk1/auxdest/tbs_12.f instead of the location specified by AUXILIARY DESTINATION. The CONFIGURE AUXNAME setting is higher in the order of precedence than AUXILIARY DESTINATION.

> **Note:** You can view any current CONFIGURE AUXNAME settings by executing the SHOW AUXNAME command, which is described in *Oracle Database Backup and Recovery Reference*.

Using AUXILIARY DESTINATION to Specify a Location for Auxiliary Files

If you use an AUXILIARY DESTINATION argument with TRANSPORT TABLESPACE, then any auxiliary set file that is not moved to another location using SET NEWNAME or CONFIGURE AUXNAME commands is stored in the auxiliary destination during the TRANSPORT TABLESPACE operation.

If you do not use AUXILIARY DESTINATION, then you must use LOG_FILE_NAME_CONVERT to specify the location of the online redo log files for the auxiliary instance. Neither SET NEWNAME nor CONFIGURE AUXNAME can affect the location of the auxiliary instance online redo logs. Thus, if you do not use AUXILIARY DESTINATION or LOG_FILE_NAME_CONVERT, then RMAN has no information about where to create the online redo logs.

Using Initialization Parameters to Name Auxiliary Files

You can use the LOG_FILE_NAME_CONVERT and DB_FILE_NAME_CONVERT initialization parameters in an auxiliary instance parameter file to determine the names for online redo logs and other database files at the auxiliary instance. If no

AUXILIARY DESTINATION clause is specified on the TRANSPORT TABLESPACE command, then these parameters determine the location of any files for which no CONFIGURE AUXNAME or SET NEWNAME command was run.

You cannot use LOG_FILE_NAME_CONVERT or DB_FILE_NAME_CONVERT to generate new Oracle Managed Files (OMF) filenames for files at the auxiliary instance when the original files are OMF files. The database manages the generation of unique filenames in each OMF destination.

You must use an AUXILIARY DESTINATION clause to control the location of the online redo log files. You must use the AUXILIARY DESTINATION clause, SET NEWNAME or CONFIGURE AUXNAME commands, or DB_CREATE_FILE_DEST initialization parameter to specify the location for OMF datafiles.

> **See Also:** *Oracle Database Reference* for more details on the LOG_FILE_NAME_CONVERT and DB_FILE_NAME_CONVERT initialization parameters

25

Transporting Data Across Platforms

You can use RMAN to transport tablespaces across platforms with different endian formats. You can also use RMAN to transport an entire database to a different platform so long as the two platforms have the same endian format.

This chapter contains the following topics:

- Overview of Cross-Platform Data Transportation
- Performing Cross-Platform Tablespace Conversion on the Source Host
- Performing Cross-Platform Datafile Conversion on the Destination Host
- Checking the Database Before Cross-Platform Database Conversion
- Converting Datafiles on the Source Host When Transporting a Database
- Converting Datafiles on the Destination Host When Transporting the Database

Overview of Cross-Platform Data Transportation

This section explains the basic concepts and tasks involved in transporting tablespaces and databases across platforms.

Purpose of Cross-Platform Data Transportation

You can transport tablespaces in a database that runs on one platform into a database that runs on a different platform. Typical uses of cross-platform transportable tablespaces include the following:

- Publishing structured data as transportable tablespaces for distribution to customers, who can convert the tablespaces for integration into their existing databases regardless of platform
- Moving data from a large data warehouse server to data marts on smaller computers such as Linux-based workstations or servers
- Sharing read-only tablespaces across a heterogeneous cluster in which all hosts share the same endian format

A full discussion of transportable tablespaces, their uses, and the different techniques for creating and using them is found in *Oracle Database Administrator's Guide*.

You can also use RMAN to transport an entire database from one platform to another. For example, business requirements demand that you run your databases on less expensive servers that use a different platform. In this case, you can use RMAN to transport the entire database rather than re-create it from scratch and use import utilities or transportable tablespaces to repopulate the data.

You can convert a database on the destination host or source host. Reasons for converting on the destination host include:

- Avoiding performance overhead on the source host due to the conversion process
- Distributing a database from one source system to multiple recipients on several different platforms
- Evaluating a migration path for a new platform

Basic Concepts of Cross-Platform Data Transportation

You must use the RMAN `CONVERT` command in a transportable tablespace operation when the **source platform** is different from the **destination platform** and the endian formats are different. When transporting between platforms for which the endian format is the same, you do not need to use `CONVERT`. You can use operating system utilities to copy the files from the source to the destination.

Tablespace and Datafile Conversion

You can perform tablespace conversion with the RMAN `CONVERT TABLESPACE` command on the source host, but not on the destination host. The `CONVERT TABLESPACE` command does not perform in-place conversion of datafiles. Rather, the command produces output files in the correct format for use on the destination platform. The command does not alter the contents of datafiles in the source database.

You can use `CONVERT DATAFILE` command to convert files on the destination host, but not on the source host. The Data Pump Export utility generates an **export dump file** that, in conjunction with datafiles manually copied to the destination host, can be imported into the destination database. Until the datafiles are transported into the destination database, the datafiles are not associated with a tablespace name in the database. In this case, RMAN cannot translate the tablespace name into a list of datafiles. Therefore, you must use `CONVERT DATAFILE` and identify the datafiles by filename.

Note: Using `CONVERT TABLESPACE` or `CONVERT DATAFILE` is only one step in using cross-platform transportable tablespaces. Read the discussion of transportable tablespaces in *Oracle Database Administrator's Guide in its entirety* before attempting to follow the procedure in this chapter.

Database Conversion

To convert a whole database to a different platform, both platforms must use the same endian format. The RMAN `CONVERT DATABASE` command automates the movement of an entire database from a source platform to a destination platform. The transported database contains the same data as the source database and also has, with a few exceptions, the same settings as the source database.

Files automatically transported to the destination platform include:

- Datafiles that belong to permanent tablespaces

  Unlike transporting tablespaces across platforms, transporting entire databases requires that certain types of blocks, such as blocks in undo segments, be reformatted to ensure compatibility with the destination platform. Even though the endian formats for the source and destination platform are the same, the datafiles for a transportable database must undergo a conversion process. You

cannot simply copy datafiles from one platform to another as you can when transporting tablespaces.

- Initialization parameter file or server parameter file

  If the databases uses a text-based initialization parameter file, then RMAN transports it. If the database uses a server parameter file, then RMAN generates an initialization parameter file based on the server parameter file and transports it and creates a new server parameter file at the destination based on the settings in the initialization parameter file.

  In most cases, some parameters in the initialization parameter file require manual updating for the new database. For example, you may change the `DB_NAME` as well as parameters such as `CONTROL_FILES` that indicate the locations of files on the destination host.

You can convert the format of the datafiles either on the source platform or the destination platform. The `CONVERT DATABASE` command does not itself convert the format of datafiles. Rather, it generates scripts that you can run manually to perform the conversion. The `CONVERT SCRIPT` parameter creates a **convert script** that you can manually execute at the destination host to convert datafile copies in batch mode. The `TRANSPORT SCRIPT` parameter generates a **transport script** that contains SQL statements to create the new database on the destination platform.

Performing Cross-Platform Tablespace Conversion on the Source Host

Refer to the list of `CONVERT` prerequisites described in *Oracle Database Backup and Recovery Reference.* Meet all these prerequisites before doing the steps in this section.

For purposes of illustration, assume that you need to transport tablespaces `finance` and `hr` from source database `prod_source`, which runs on a Sun Solaris host. You plan to transport them to destination database `prod_dest` running on a Linux PC. You plan to store the converted datafiles in the temporary directory `/tmp/transport_linux/` on the source host.

To perform cross-platform tablespace conversion on the source host:

1. Start SQL*Plus and connect to the source database `prod_source` with administrator privileges.
2. Query the name for the destination platform in `V$TRANSPORTABLE_PLATFORM`.

   The database has a list of its own internal names for each platform supporting cross-platform data transport. You may need the exact name of the source or destination platform as a parameter to the `CONVERT` command. Query `V$TRANSPORTABLE_PLATFORM` to get the platform names. The following example queries Linux platform names:

   ```
   SELECT PLATFORM_ID, PLATFORM_NAME, ENDIAN_FORMAT
   FROM   V$TRANSPORTABLE_PLATFORM
   WHERE  UPPER(PLATFORM_NAME) LIKE '%LINUX%';
   ```

   The `PLATFORM_NAME` for Linux on a PC is `Linux IA (32-bit)`.
3. Place the tablespaces to be transported in read-only mode. For example, enter:

   ```
   ALTER TABLESPACE finance READ ONLY;
   ALTER TABLESPACE hr READ ONLY;
   ```
4. Choose a method for naming the output files.

You must use the FORMAT or DB_FILE_NAME_CONVERT arguments to CONVERT to control the names of the output files. The rules are listed in order of precedence:

a. Files that match any patterns provided in CONVERT ... DB_FILE_NAME_CONVERT clause are named based upon this pattern.

b. If you specify a FORMAT clause, then any file not named based on patterns provided in CONVERT ... DB_FILE_NAME_CONVERT clause is named based on the FORMAT pattern.

Note: You cannot use CONVERT ... DB_FILE_NAME_CONVERT to generate output filenames for CONVERT when the source files have Oracle-managed file names and the destination files have Oracle-managed file names.

---

5. Start RMAN and connect to the *source* database (not the destination database) as TARGET. For example, enter:

```
% rman
RMAN> CONNECT TARGET SYS@prod_source
```

6. Run the CONVERT TABLESPACE command to convert the datafiles into the endian format of the destination host.

   In the following example, the FORMAT argument controls the name and location of the converted datafiles:

```
RMAN> CONVERT TABLESPACE finance,hr
2>   TO PLATFORM 'Linux IA (32-bit)'
3>   FORMAT '/tmp/transport_linux/%U';
```

   The result is a set of converted datafiles in the /tmp/transport_linux/ directory, with data in the correct endian format for the Linux IA (32-bit) platform.

   See Also: *Oracle Database Backup and Recovery Reference* for the full semantics of the CONVERT command

7. Follow the rest of the general outline for transporting tablespaces:

   a. Use the Oracle Data Pump Export utility to create the export dump file on the source host.

   b. Move the converted datafiles and the export dump file from the source host to the desired directories on the destination host.

   c. Plug the tablespace into the new database with the Import utility.

   d. If applicable, place the transported tablespaces into read/write mode.

Performing Cross-Platform Datafile Conversion on the Destination Host

Refer to the list of CONVERT prerequisites described in *Oracle Database Backup and Recovery Reference.* Meet these prerequisites before doing the steps in this section.

About Cross-Platform Datafile Conversion on the Destination Host

Datafile conversion necessitates that you choose a technique for naming the output files. You must use the FORMAT or DB_FILE_NAME_CONVERT arguments to CONVERT to control the naming of output files. The rules are listed in order of precedence:

1. Files that match any patterns provided in `CONVERT ... DB_FILE_NAME_CONVERT` clause are named based upon this pattern.
2. If you specify a `FORMAT` clause, then any file not named based on patterns provided in `CONVERT ... DB_FILE_NAME_CONVERT` clause is named based on the `FORMAT` pattern.

> **Note:** You cannot use `CONVERT ... DB_FILE_NAME_CONVERT` to generate output filenames for `CONVERT` when both the source and destination files are Oracle Managed Files.

If the source and destination platforms differ, then you must specify the `FROM PLATFORM` parameter. View platform names by querying `V$TRANSPORTABLE_PLATFORM`. The `FROM PLATFORM` value must match the format of the datafiles to be converted to avoid an error. If you do not specify `FROM PLATFORM`, then this parameter defaults to the value of the destination platform.

Using CONVERT DATAFILE to Convert Datafile Formats

This section explains how to use `CONVERT DATAFILE`. The section assumes that you intend to transport tablespaces `finance` (datafiles `fin/fin01.dbf` and `fin/fin02.dbf`) and `hr` (datafiles `hr/hr01.dbf` and `hr/hr02.dbf`) from a source database named `prod_source`. The database runs on a Sun Solaris host. You plan to transport these tablespaces into a destination database named `prod_dest`, which runs on a Linux PC. You plan to perform conversion on the destination host.

When the datafiles are plugged into the destination database, you plan to store them in `/orahome/dbs` and preserve the current directory structure. That is, datafiles for the `hr` tablespace will be stored in the `/orahome/dbs/hr` subdirectory, and datafiles for the `finance` tablespace will be stored in the `/orahome/dbs/fin` directory.

To perform cross-platform datafile conversion on the destination host:

1. Start SQL*Plus and connect to the source database `prod_source` with administrator privileges.
2. Query the name for the source platform in `V$TRANSPORTABLE_PLATFORM`.

   The database has a list of its own internal names for each platform supporting cross-platform data transport. You may need the exact name of the source or destination platform as a parameter to the `CONVERT` command. For example, you can obtain the platform name of the connected database as follows:

   ```
   SELECT PLATFORM_NAME
   FROM   V$TRANSPORTABLE_PLATFORM
   WHERE  PLATFORM_ID =
          ( SELECT PLATFORM_ID
            FROM   V$DATABASE );
   ```

   For this scenario, assume the `PLATFORM_NAME` for the source host is `Solaris[tm] OE (64-bit)`.
3. Identify the tablespaces to be transported from the source database and place them in read-only mode.

   For example, enter the following SQL statements to place `finance` and `hr` in read-only mode:

   ```
   ALTER TABLESPACE finance READ ONLY;
   ```

```
ALTER TABLESPACE hr READ ONLY;
```

4. On the source host, use Data Pump Export to create the export dump file

   In this example, the dump file is named `expdat.dmp`.

5. Make the export dump file and the datafiles to be transported available to the destination host.

   You can use NFS to make the dump file and current database files (not copies) accessible. Alternatively, you can use an operating system utility to copy these files to the destination host.

   In this example, you store the files in the in the `/tmp/transport_solaris/` directory of the destination host. You preserve the subdirectory structure from the original location of the files, that is, the datafiles are stored as:

   - `/tmp/transport_solaris/fin/fin01.dbf`
   - `/tmp/transport_solaris/fin/fin02.dbf`
   - `/tmp/transport_solaris/hr/hr01.dbf`
   - `/tmp/transport_solaris/hr/hr02.dbf`

6. Start RMAN and connect to the *destination* database (not the *source* database) as `TARGET`. For example, enter the following command:

   ```
   % rman
   RMAN> CONNECT TARGET SYS@prod_dest
   ```

7. Execute the `CONVERT DATAFILE` command to convert the datafiles into the endian format of the destination host.

   In this example, you use `DB_FILE_NAME_CONVERT` to control the name and location of the converted datafiles. You also specify the `FROM PLATFORM` clause.

   ```
   RMAN> CONVERT DATAFILE
   2>   '/tmp/transport_solaris/fin/fin01.dbf',
   3>   '/tmp/transport_solaris/fin/fin02.dbf',
   4>   '/tmp/transport_solaris/hr/hr01.dbf',
   5>   '/tmp/transport_solaris/hr/hr02.dbf'
   6>   DB_FILE_NAME_CONVERT
   7>     '/tmp/transport_solaris/fin','/orahome/dbs/fin',
   8>     '/tmp/transport_solaris/hr','/orahome/dbs/hr'
   9>   FROM PLATFORM 'Solaris[tm] OE (64-bit)
   ```

   The result is a set of converted datafiles in the `/orahome/dbs/` directory that are named as follows:

   - `/orahome/dbs/fin/fin01.dbf`
   - `/orahome/dbs/fin/fin02.dbf`
   - `/orahome/dbs/hr/hr01.dbf`
   - `/orahome/dbs/hr/hr02.dbf`

8. Follow the rest of the general outline for transporting tablespaces:

   a. Plug the tablespace into the new database with the Import utility.

   b. If applicable, place the transported tablespaces into read-only mode.

   See Also: *Oracle Database Backup and Recovery Reference* for the syntax and semantics of the `CONVERT` command

Checking the Database Before Cross-Platform Database Conversion

As explained in "Basic Concepts of Cross-Platform Data Transportation" on page 25-2, you can use the RMAN `CONVERT DATABASE` command to automate the copying of an entire database from one platform to another.

Before converting the database, refer to the list of `CONVERT DATABASE` prerequisites described in *Oracle Database Backup and Recovery Reference*. For example, the source and destination platforms must use the same endian format. Make sure to meet all these prerequisites before attempting the procedure in this section.

The principal prerequisite on cross-platform transportable database is that the source and destination platform must share the same endian format. For example, you can transport a database from Microsoft Windows to Linux for x86 (both little-endian), or from HP-UX to AIX (both big-endian), but not from HP-UX to Linux for x86 (big-endian to little-endian).

> **Note:** If you cannot use `CONVERT DATABASE` because the platforms do not share endian formats, then you can create a new database on a destination platform manually and transport needed tablespaces from the source database with cross-platform transportable tablespaces.

To prepare for database conversion:

1. Start a SQL*Plus session as `SYSDBA` on the source database.
2. Open the database in read-only mode.

   ```
   ALTER DATABASE OPEN READ ONLY;
   ```

3. Make sure that server output is on in SQL*Plus.

   For example, enter the following SQL*Plus command:

   ```
   SET SERVEROUTPUT ON
   ```

4. Execute the `DBMS_TDB.CHECK_DB` function.

   This check ensures that no conditions would prevent the transport of the database, such as incorrect compatibility settings, in-doubt or active transactions, or incompatible endian formats between the source platform and destination platform.

   You can call `CHECK_DB` without arguments to see if a condition at the source database prevents transport. You can also call this function with the arguments shown in Table 25–1.

Table 25–1 CHECK_DB Procedure Parameters

Parameter	Description
`target_platform_name`	The name of the destination platform as it appears in `V$DB_TRANSPORTABLE_PLATFORM`. This parameter is optional, but is required when the `skip_option` parameter is used. If omitted, it is assumed that the destination platform is compatible with the source platform, and only the conditions not related to platform compatibility are tested.

Table 25–1 (Cont.) CHECK_DB Procedure Parameters

Parameter	Description
skip_option	Specifies which, if any, parts of the database to skip when checking whether the database can be transported. Supported values (of type NUMBER) are: ■ SKIP_NONE (or 0), which checks all tablespaces ■ SKIP_OFFLINE (or 2), which skips checking datafiles in offline tablespaces ■ SKIP_READONLY (or 3), which skips checking datafiles in read-only tablespaces

Example 25–1 illustrates executing CHECK_DB on a 32-bit Linux platform for transporting a database to 32-bit Windows, skipping read-only tablespaces.

Example 25–1 Executing DBMS_TDB.CHECK_DB

```
DECLARE
  db_ready BOOLEAN;
BEGIN
  db_ready :=
       DBMS_TDB.CHECK_DB('Microsoft Windows IA (32-bit)',DBMS_TDB.SKIP_READONLY);
END;
/

PL/SQL procedure successfully completed.
```

If no warnings appear, or if DBMS_TDB.CHECK_DB returns TRUE, then you can currently transport the database. Proceed to step 6.

If warnings appears, or if DBMS_TDB.CHECK_DB returns FALSE, then you cannot currently transport the database. Proceed to the next step.

5. Examine the output to learn why the database cannot be transported, fix the problem if possible, and then return to the preceding step. Refer to the DBMS_TDB documentation for the conditions present.
6. Execute DBMS_TDB.CHECK_EXTERNAL to identify any external tables, directories, or BFILEs. RMAN cannot automate the transport of these files, so you must copy the files manually and re-create database directories.

   Example 25–2 shows how to call DBMS_TDB.CHECK_EXTERNAL.

Example 25–2 Executing DBMS_TDB.CHECK_EXTERNAL

```
DECLARE
     external BOOLEAN;
BEGIN
    /* value of external is ignored, but with SERVEROUTPUT set to ON
     * dbms_tdb.check_external displays report of external objects
     * on console */
    external := DBMS_TDB.CHECK_EXTERNAL;
END;
/
```

If no external objects exist, then the procedure completes with no output. If external objects exist, however, then the output is similar to the following:

```
The following external tables exist in the database:
SH.SALES_TRANSACTIONS_EXT
```

```
The following directories exist in the database:
SYS.DATA_PUMP_DIR, SYS.MEDIA_DIR, SYS.DATA_FILE_DIR, SYS.LOG_FILE_DIR
The following BFILEs exist in the database:
PM.PRINT_MEDIA

PL/SQL procedure successfully completed.
```

Converting Datafiles on the Source Host When Transporting a Database

This section assumes that you have already met all of the CONVERT DATABASE prerequisites and followed the steps in "Checking the Database Before Cross-Platform Database Conversion" on page 25-7. The goal of this procedure is to convert the format of datafiles on the source host as part of a cross-platform database transport.

Assume that you want to convert a database running on Solaris to a database that runs on Windows.

To convert the database on the source platform:

1. Open the source database in read-only mode.

   ```
   ALTER DATABASE OPEN READ ONLY;
   ```

2. Start RMAN and connect to the source database as TARGET. For example, enter the following commands:

   ```
   % rman
   RMAN> CONNECT TARGET SYS@source_db
   ```

3. Run the CONVERT DATABASE command.

   Example 25–3 shows a CONVERT DATABASE command (sample output included). The TRANSPORT SCRIPT parameter specifies the location of the generated SQL script that you can use to create the new database. The TO PLATFORM parameter indicates the platform of the destination database. The DB_FILE_NAME_CONVERT parameter specifies the naming scheme for the generated datafiles.

Example 25–3 Converting a Database on the Source Host

```
RMAN> CONVERT DATABASE
2>   NEW DATABASE 'newdb'
3>   TRANSPORT SCRIPT '/tmp/convertdb/transportscript.sql'
4>   TO PLATFORM 'Microsoft Windows IA (32-bit)'
5>   DB_FILE_NAME_CONVERT '/disk1/oracle/dbs' '/tmp/convertdb';

Starting convert at 25-NOV-06
using channel ORA_DISK_1

External table SH.SALES_TRANSACTIONS_EXT found in the database

Directory SYS.DATA_PUMP_DIR found in the database
Directory SYS.MEDIA_DIR found in the database
Directory SYS.DATA_FILE_DIR found in the database
Directory SYS.LOG_FILE_DIR found in the database

BFILE PM.PRINT_MEDIA found in the database

User SYS with SYSDBA and SYSOPER privilege found in password file
User OPER with SYSDBA privilege found in password file
channel ORA_DISK_1: starting datafile conversion
input datafile fno=00001 name=/disk1/oracle/dbs/tbs_01.f
```

```
converted datafile=/tmp/convertdb/tbs_01.f
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting datafile conversion
input datafile fno=00002 name=/disk1/oracle/dbs/tbs_ax1.f
converted datafile=/tmp/convertdb/tbs_ax1.f
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:03
.
.
.
channel ORA_DISK_1: starting datafile conversion
input datafile fno=00016 name=/disk1/oracle/dbs/tbs_52.f
converted datafile=/tmp/convertdb/tbs_52.f
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:01
Run SQL script /tmp/convertdb/transportscript.sql on the destination platform
  to create database
Edit init.ora file init_00gb3vfv_1_0.ora. This PFILE will be used to
  create the database on the destination platform
To recompile all PL/SQL modules, run utlirp.sql and utlrp.sql on
  the destination platform
To change the internal database identifier, use DBNEWID Utility
Finished backup at 25-NOV-06
```

4. After `CONVERT DATABASE` completes, you can open the source database read/write again.
5. Move the datafiles generated by `CONVERT DATABASE` to the desired locations on the destination host.

   In Example 25–3, the command creates the files in the `/tmp/convertdb/` directory on the source host. Move these files to the directory on the destination host that will contain the destination database files.
6. If the path to the datafiles is different on the destination host, then edit the transport script to refer to the new datafile locations.
7. If necessary, edit the initialization parameter file to change any settings for the destination database.

   You should edit several entries at the top of the initialization parameter file when the database is moved to the destination platform. For example, the initialization parameter file may look as follows:

   ```
   # Please change the values of the following parameters:
    control_files            = "/tmp/convertdb/cf_D-NEWDBT_id-1778429277_00gb9u2s"
    db_recovery_file_dest    = "/tmp/convertdb/orcva"
    db_recovery_file_dest_size= 10737418240
    instance_name            = "NEWDBT"
    service_names            = "NEWDBT.regress.rdbms.dev.us.oracle.com"
    plsql_native_library_dir = "/tmp/convertdb/plsqlnld1"
    db_name                  = "NEWDBT"
   ```

8. If necessary, edit the transport script to use the new names for the converted datafiles.

   In Example 25–3, the transport script is named `/tmp/convertdb/transportscript.sql`. You run this script on the *destination* host to actually create the database. Thus, you must edit this script with the correct names for the datafiles.
9. On the destination host, start SQL*Plus and connect to the destination database instance as `SYSDBA` using operating system authentication.

For example, connect as follows:

```
SQL> CONNECT / AS SYSDBA
```

If you choose not to use operating system authentication, then you must first configure Oracle Net files, create a **password file**, and start the listener. You can then connect to the instance with a net service name.

10. Execute the transport script in SQL*Plus to create the new database on the destination host.

```
SQL> @transportscript
```

When the transport script finishes, the creation of the new database is complete.

Converting Datafiles on the Destination Host When Transporting the Database

This section assumes that you have already met all of the CONVERT DATABASE prerequisites and followed the steps in "Checking the Database Before Cross-Platform Database Conversion" on page 25-7. The goal of this procedure is to convert the format of datafiles on the destination host as part of a cross-platform database transport.

Performing the datafile conversion in the following phases:

1. Performing Preliminary Datafile Conversion Steps on the Source Host
2. Converting Datafiles on the Destination Host

Performing Preliminary Datafile Conversion Steps on the Source Host

In this procedure, you execute the CONVERT DATABASE command on the source host. This command generates an initialization parameter file and scripts that you can edit for use on the destination host. You also copy the unconverted datafiles from the source host to the destination host.

To perform preliminary datafile conversion steps on the source host:

1. Ensure that the database is open in read-only mode.
2. Start RMAN and connect to the *source* database as TARGET.

   For example, enter the following commands:

   ```
   % rman
   RMAN> CONNECT TARGET SYS@source_db
   ```

3. Run the CONVERT DATABASE ON DESTINATION command.

   Example 25–4 shows a sample CONVERT DATABASE command (sample output included). The ON DESTINATION PLATFORM parameter specifies that any CONVERT commands required for datafiles should be performed on the destination platform rather than the source platform. The FORMAT parameter specifies the naming scheme for the generated files.

Example 25–4 Executing CONVERT DATABASE ON DESTINATION PLATFORM

```
RMAN> CONVERT DATABASE
2>   ON DESTINATION PLATFORM
3>   CONVERT SCRIPT '/tmp/convertdb/convertscript-target'
4>   TRANSPORT SCRIPT '/tmp/convertdb/transportscript-target'
5>   NEW DATABASE 'newdbt'
```

```
6>   FORMAT '/tmp/convertdb/%U';

Starting convert at 28-JAN-05
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=39 devtype=DISK

External table SH.SALES_TRANSACTIONS_EXT found in the database

Directory SYS.DATA_PUMP_DIR found in the database
Directory SYS.MEDIA_DIR found in the database
Directory SYS.DATA_FILE_DIR found in the database
Directory SYS.LOG_FILE_DIR found in the database

BFILE PM.PRINT_MEDIA found in the database

User SYS with SYSDBA and SYSOPER privilege found in password file
User OPER with SYSDBA privilege found in password file
channel ORA_DISK_1: starting to check datafiles
input datafile fno=00001 name=/disk1/oracle/dbs/tbs_01.f
channel ORA_DISK_1: datafile checking complete, elapsed time: 00:00:00
channel ORA_DISK_1: starting to check datafiles
input datafile fno=00002 name=/disk1/oracle/dbs/tbs_ax1.f
channel ORA_DISK_1: datafile checking complete, elapsed time: 00:00:00
channel ORA_DISK_1: starting to check datafiles
input datafile fno=00017 name=/disk1/oracle/dbs/tbs_03.f
channel ORA_DISK_1: datafile checking complete, elapsed time: 00:00:00
.
.
.
channel ORA_DISK_1: starting to check datafiles
input datafile fno=00015 name=/disk1/oracle/dbs/tbs_51.f
channel ORA_DISK_1: datafile checking complete, elapsed time: 00:00:00
channel ORA_DISK_1: starting to check datafiles
input datafile fno=00016 name=/disk1/oracle/dbs/tbs_52.f
channel ORA_DISK_1: datafile checking complete, elapsed time: 00:00:00
Run SQL script /tmp/convertdb/transportscript-target on the destination platform
to create database
Edit init.ora file /tmp/convertdb/init_00gb9u2s_1_0.ora. This PFILE will be used
to create the database on the destination platform
Run RMAN script /tmp/convertdb/convertscript-target on destination platform to
convert datafiles
To recompile all PL/SQL modules, run utlirp.sql and utlrp.sql on the destination
platform
To change the internal database identifier, use DBNEWID Utility
Finished backup at 28-JAN-05
```

The command in Example 25–4 creates a transport script, an initialization parameter file for the new database, and a convert script containing RMAN `CONVERT DATAFILE` commands for each datafile being converted.

Note: `CONVERT DATABASE ON DESTINATION PLATFORM` does *not* produce converted datafile copies. The command only creates scripts.

---

4. Use an operating system utility to copy the following files to a temporary location on the destination host:
   - The datafiles to be converted

- The convert script
- The transport script
- The initialization file for the destination database

5. Make the source database read/write.

Converting Datafiles on the Destination Host

This section explains how to use the script created in the previous section to convert the datafiles on the destination host.

The convert script created in the previous phase uses the original datafile names of the source database files. The FORMAT parameter specifies the name that was generated with the FORMAT or DB_FILE_NAME_CONVERT parameter of CONVERT DATABASE.

If the datafiles of the source database are accessible from the destination host with the same path names, then so long as the source database is read-only you can run the convert script on the destination host without any changes. For example, if the source and destination hosts both use NFS to mount a disk containing the source datafiles, and if the mount point for both hosts is /fs1/dbs/, then no editing is needed.

To convert the datafiles on the destination host:

1. If necessary, edit the convert script.

   In the script, one CONVERT DATAFILE command exists for each datafile to be converted. The convert script should indicate the current temporary filenames of the unconverted datafiles and the output filenames of the converted datafiles. A typical convert script looks as follows:

```
RUN
{
 CONVERT
 FROM PLATFORM 'Linux IA (32-bit)'
 PARALLELISM 10
  DATAFILE '/disk1/oracle/dbs/tbs_01.f'
  FORMAT
 '/tmp/convertdb/data_D-TV_I-1778429277_TS-SYSTEM_FNO-1_7qgb9u2s';

 DATAFILE '/disk1/oracle/dbs/tbs_ax1.f'
 FORMAT
 '/tmp/convertdb/data_D-TV_I-1778429277_TS-SYSAUX_FNO-2_7rgb9u2s';

 DATAFILE '/disk1/oracle/dbs/tbs_03.f'
 FORMAT
 '/tmp/convertdb/data_D-TV_I-1778429277_TS-SYSTEM_FNO-17_7sgb9u2s';

 DATAFILE '/disk1/oracle/dbs/tbs_51.f'
 FORMAT
 '/tmp/convertdb/data_D-TV_I-1778429277_TS-TBS_5_FNO-15_8egb9u2u';

 DATAFILE '/disk1/oracle/dbs/tbs_52.f'
  FORMAT
 '/tmp/convertdb/data_D-TV_I-1778429277_TS-TBS_5_FNO-16_8fgb9u2u';
}
```

   Edit each DATAFILE command in the convert script to specify the temporary location of each datafile as input. Also, edit the FORMAT parameter of each command to specify the desired final location of the datafiles of the transported database.

2. If necessary, edit the initialization parameter file on the destination host to change settings for the destination database.

   You should edit several entries at the top of the initialization parameter file before moving the database to the destination platform. For example, the initialization parameter file may look as follows:

   ```
   # Please change the values of the following parameters:
   control_files            = "/tmp/convertdb/cf_D-NEWDBT_id-1778429277_00gb9u2s"
   db_recovery_file_dest    = "/tmp/convertdb/orcva"
   db_recovery_file_dest_size= 10737418240
   instance_name            = "NEWDBT"
   service_names            = "NEWDBT.regress.rdbms.dev.us.oracle.com"
   plsql_native_library_dir = "/tmp/convertdb/plsqlnld1"
   db_name                  = "NEWDBT"
   ```

3. On the destination host, use SQL*Plus to start the database instance in `NOMOUNT` mode.

   Specify the initialization parameter file that you copied in the preceding step. For example, enter the following command:

   ```
   SQL> STARTUP NOMOUNT PFILE='/tmp/init_convertdb_00i2gj63_1_0.ora'
   ```

4. Start RMAN and connect to the destination database (not the source database) as `TARGET`. For example, enter the following command:

   ```
   % rman
   RMAN> CONNECT TARGET SYS@prod_dest
   ```

5. Run the convert script at the RMAN prompt. For example, enter the following command:

   ```
   RMAN> @/tmp/convertdb/convertscript-target
   ```

6. Shut down the database instance.

   This step is necessary because the transport script that needs to execute already includes a `STARTUP NOMOUNT` command.

7. If necessary, edit the transport script to use the new names for the converted datafiles.

   In Example 25–3, the transport script is `/tmp/convertdb/transportscript.sql`. You run this script on the destination host to create the database. Thus, you must edit this script with the correct names for the datafiles.

8. Execute the transport script in SQL*Plus.

   For example, create the new database on the destination host as follows:

   ```
   SQL> @/tmp/convertdb/transportscript
   ```

When the transport script completes, the destination database is created.

26

Performing ASM Data Migration

This chapter describes how to migrate data into and out of **Automatic Storage Management (ASM)** storage with RMAN. This chapter includes the following topics:

- Overview of ASM Data Migration
- Preparing to Migrate the Database to ASM
- Migrating the Database to ASM
- Migrating a Database from ASM to Alternative Storage
- Moving Datafiles Between ASM Disk Groups

Overview of ASM Data Migration

This section explains the basic concepts and tasks involved in migrating data to and from ASM.

Purpose of ASM Data Migration

Alternatives to ASM storage include file systems, raw disks, and SAN configurations. ASM includes a number of benefits over these storage alternatives, including performance optimization, redundancy protection, and load balancing. You do not need to obtain a third-party Logical Volume Manager because ASM manages disks for you. Oracle Real Application Clusters (Oracle RAC) databases benefit from ASM because it provides ready-made shared storage.

If a database currently uses a storage system other than ASM, then you can migrate all or part of the database into ASM, thereby simplifying database administration. You can also migrate a flash recovery area to ASM.

Native operating system commands such as Linux `cp` or Windows `COPY` cannot write or read files in ASM storage. Because RMAN can read and write ASM files, you can use RMAN to copy datafiles into and out of ASM storage or between ASM disk groups. This technique is useful if you need to store backups on user-managed disks.

Basic Concepts of ASM Data Migration

You can migrate data to ASM with RMAN even if you are not using RMAN as your primary backup tool. The migration requires one RMAN database backup.

If you have sufficient disk space to hold the entire database both in ASM and alternative storage systems, then you can move a database directly into ASM. If you do not have sufficient storage, then you can back the database up to tape, create an ASM disk group that uses old disk space, and restore the database from tape to ASM.

After you set the location of the new recovery area, existing backups remain in the old recovery area and count against the total disk quota of the recovery area. The backups are deleted from the old recovery area when space is needed. These backups are usable by RMAN. You do not need to move legacy backups to the new ASM recovery area unless you need disk space. To free space consumed by files in the old recovery area, you can back them up to tape or migrate them to the ASM recovery area.

Note: A **foreign archived redo log** is a log received by a logical standby database for a LogMiner session. Foreign archived redo logs cannot be migrated.

Migrating a database from ASM to an alternative storage system is similar to migration from an alternative storage system to ASM. The primary change is to modify each step to refer to file locations in the alternative storage system.

See Also: *Oracle Database Storage Administrator's Guide* to learn how to migrate the database to ASM with Enterprise Manager

Basics Steps of Data Migration to ASM

The basic steps of migrating the whole database and **flash recovery area** from alternative storage to ASM are as follows:

1. Back up the database and server parameter file, and disable **Oracle Flashback Database**.

   This step is described in "Preparing to Migrate the Database to ASM" on page 26-2.

2. Restore files to ASM, recover the database, and optionally migrate the flash recovery area to ASM.

   This step is described in "Migrating the Database to ASM" on page 26-4.

To migrate files from alternative storage to ASM, see "Migrating a Database from ASM to Alternative Storage" on page 26-8.

Preparing to Migrate the Database to ASM

This section explains how to prepare the database for migration. This section makes the following assumptions:

- You want to migrate the database to two ASM disk groups: `+DATA` for the database and `+FRA` for the flash recovery area.
- The database to be migrated to ASM storage is named `mydb`.

Note: If you do *not* want to migrate the flash recovery area, then skip step 10.

To prepare the database for ASM migration:

1. If the `COMPATIBLE` setting for the database is less than 11.0.0, then make any read-only transportable tablespaces read/write.

   Read-only transportable tablespaces cannot be migrated because RMAN cannot back them up.

2. If the database is a **physical standby database**, and if managed recovery is started, then stop managed recovery.

   For example, connect SQL*Plus to the database with SYSDBA privileges, and execute the following statement to stop managed recovery:

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
   ```

   Keep this terminal window open.

3. Copy the server parameter file or initialization parameter file to a temporary location.

   The following example uses an operating system utility to copy the server parameter file:

   ```
   % cp spfileMYDB.ora orig_spfileMYDB.ora
   ```

4. In a new terminal window, start RMAN session and connect as TARGET to the database to be migrated. Optionally, connect to a recovery catalog.

5. Back up the datafiles to the ASM disk group.

   The following example uses a RUN command to make a **level 0 incremental backup** and allocates four channels to increase the backup speed. Increase or decrease this number accordingly. The format clause specifies +DATA, which is the name of the ASM disk group to be used for storing the database.

   ```
   RUN
   {
     ALLOCATE CHANNEL dev1 DEVICE TYPE DISK;
     ALLOCATE CHANNEL dev2 DEVICE TYPE DISK;
     ALLOCATE CHANNEL dev3 DEVICE TYPE DISK;
     ALLOCATE CHANNEL dev4 DEVICE TYPE DISK;
     BACKUP AS COPY
       INCREMENTAL LEVEL 0
       DATABASE
       FORMAT '+DATA'
       TAG 'ORA_ASM_MIGRATION';
   }
   ```

6. If **block change tracking** is enabled for the database, then optionally make a level 1 incremental backup that you can use later to recover the database copy.

   The following example makes an incremental level 1 copy of the level 0 backup created in the previous step:

   ```
   RUN
   {
     ALLOCATE CHANNEL dev1 DEVICE TYPE DISK;
     ALLOCATE CHANNEL dev2 DEVICE TYPE DISK;
     ALLOCATE CHANNEL dev3 DEVICE TYPE DISK;
     ALLOCATE CHANNEL dev4 DEVICE TYPE DISK;
     BACKUP INCREMENTAL LEVEL 1
       FOR RECOVER OF COPY WITH TAG 'ORA_ASM_MIGRATION'
       DATABASE;
   }
   ```

7. If the database is in ARCHIVELOG mode, and if the database is open, then archive the online logs.

   The following example uses the SQL command to archive the current redo logs:

   ```
   RMAN> SQL "ALTER SYSTEM ARCHIVE LOG CURRENT";
   ```

8. If the database instance is currently using a server parameter file, then back it up.

   The following example backs up the server parameter file:

   ```
   RMAN> BACKUP AS BACKUPSET SPFILE;
   ```

9. If block change tracking is enabled, then disable it.

   The following command disables block change tracking:

   ```
   RMAN> SQL "ALTER DATABASE DISABLE BLOCK CHANGE TRACKING";
   ```

10. If Flashback Database is enabled, then disable it and drop any guaranteed restore points.

    Note: If you are not migrating the recovery area, then skip this step.

    Disabling Flashback Database is necessary because you cannot migrate **flashback logs** to ASM. The following command disables Flashback Database:

    ```
    RMAN> SQL "ALTER DATABASE FLASHBACK OFF";
    ```

    The following command drops the guaranteed restore point named `Q106`:

    ```
    RMAN> SQL "DROP RESTORE POINT Q106";
    ```

11. Shut down the database consistently.

    The following command shuts down the database:

    ```
    RMAN> SHUTDOWN IMMEDIATE;
    ```

Migrating the Database to ASM

The following procedure is intended to minimize database downtime. Note that the steps differ slightly depending on whether you are migrating a primary or standby database. The procedure makes the same assumptions described in "Preparing to Migrate the Database to ASM" on page 26-2. If you are not migrating the recovery area to ASM, then you need to modify some of the steps, which are noted.

Note: The following procedure switches between SQL*Plus and RMAN, so keep a terminal window open for each utility.

To migrate the database to ASM:

1. Follow the steps in "Preparing to Migrate the Database to ASM" on page 26-2.
2. Restore or create a server parameter file in ASM storage.

   The steps depend on whether the database is using a server parameter file:

   - If the database is using a server parameter file, then restore it to the ASM disk group with the following commands, where *sid* is the SID of the instance:

     ```
     RMAN> STARTUP MOUNT;
     RMAN> RESTORE SPFILE TO '+DATA/spfilesid.ora';
     RMAN> SHUTDOWN IMMEDIATE;
     ```

- If the database is not using a server parameter file, then create one in ASM. Execute the CREATE SPFILE command in SQL*Plus as follows, where *sid* is the SID of the database (the command spans two lines):

  ```
  SQL> CREATE SPFILE='+DATA/spfilesid.ora' FROM PFILE='?/dbs/initsid.ora';
  ```

  Afterward, delete spfile*sid*.ora and init*sid*.ora from the ?/dbs directory and create a new init*sid*.ora with the following line of content:

  ```
  SPFILE='+DATA/spfilesid.ora'
  ```

3. Set Oracle Managed Files initialization parameters to ASM locations.

   Note: If you are not migrating the flash recovery area, then do not change the DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE initialization parameter settings. However, you must set DB_CREATE_ONLINE_LOG_DEST_*n* parameter to an ASM location for migration of the online redo logs.

   Set the DB_CREATE_FILE_DEST and optional DB_CREATE_ONLINE_LOG_DEST_*n* initialization parameters to ASM disk groups. If the database uses a recovery area, then change the recovery area location to the ASM disk group. Also, change the recovery area size.

   Execute commands in SQL*Plus as shown in the following example. The example assumes that the size of the flash recovery area is 100 GB and specifies the disk group +FRA for the flash recovery area.

   ```
   SQL> STARTUP FORCE NOMOUNT;
   SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST='+DATA' SID='*';
   SQL> ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE=100G SID='*';
   SQL> ALTER SYSTEM SET DB_RECOVERY_FILE_DEST='+FRA' SID='*';
   ```

4. Set the CONTROL_FILES initialization parameter to ASM locations.

   If you *are* migrating the flash recovery area, then enter the following commands in SQL*Plus to restart the database instance and set the control file locations to disk groups +DATA and +FRA:

   ```
   SQL> STARTUP FORCE NOMOUNT;
   SQL> ALTER SYSTEM SET CONTROL_FILES='+DATA','+FRA' SCOPE=SPFILE SID='*';
   ```

   If you are *not* migrating the flash recovery area, then enter the following commands in SQL*Plus to restart the database instance and set the control file locations to disk group +DATA:

   ```
   SQL> STARTUP FORCE NOMOUNT;
   SQL> ALTER SYSTEM SET CONTROL_FILES='+DATA','+DATA' SCOPE=SPFILE SID='*';
   ```

5. Migrate the control file to ASM and mount the control file.

   Switch to the RMAN terminal to restore the control file. In the following example, *original_cf_name* is a control file name in the initialization parameter file before migration:

   ```
   RMAN> STARTUP FORCE NOMOUNT;
   RMAN> RESTORE CONTROLFILE FROM 'original_cf_name';
   RMAN> ALTER DATABASE MOUNT;
   ```

6. Migrate the datafiles to ASM.

Use RMAN to switch to the database copy that you created in step 5 in "Preparing to Migrate the Database to ASM" on page 26-2. The switch renames all the datafiles to files on ASM disk groups. Afterward, recover the database. If incremental backups were taken, then RMAN applies them during recovery. For example, enter the following commands at the RMAN prompt:

```
SWITCH DATABASE TO COPY;
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev2 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev3 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev4 DEVICE TYPE DISK;
  RECOVER DATABASE;
}
```

7. If the database uses block change tracking or Flashback Database, then enable these features.

   Note: If you are not migrating the recovery area, then you do not need to enable Flashback Database because you did not disable it.

   For example, enter the following statements in SQL*Plus:

   ```
   SQL> ALTER DATABASE ENABLE BLOCK CHANGE TRACKING USING FILE '+DATA';
   SQL> ALTER DATABASE FLASHBACK ON;
   ```

8. Place the database in its normal operation mode.

   The normal operational mode depends on whether the database is a primary or standby database:

   - If the database is a primary database, then open it as follows:

     ```
     SQL> ALTER DATABASE OPEN;
     ```

   - If the database is a standby database, then resume managed recovery mode as follows:

     ```
     SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
     ```

9. Drop the tempfiles and re-create them in ASM.

   Use SQL*Plus to re-create the tempfiles. In the following example, the name of the tempfile in the original storage is *tempfile_name*. The name of the temporary tablespace is *temp_tbs_name*.

   ```
   SQL> ALTER DATABASE TEMPFILE 'tempfile_name' DROP;
   SQL> ALTER TABLESPACE temp_tbs_name ADD TEMPFILE;
   ```

10. Migrate the online redo log files.

    If this is a primary database, then add new log group members in ASM and drop the old members. You can use the following PL/SQL script to migrate the online redo log groups into an ASM disk group. The PL/SQL script assumes that the Oracle Managed Files initialization parameters specified in step 3 are set.

Example 26–1 Migrating the Online Redo Logs

```
SET SERVEROUTPUT ON;
DECLARE
```

```
   CURSOR rlc IS
      SELECT GROUP# GRP, THREAD# THR, BYTES, 'NO' SRL
      FROM   V$LOG
      UNION
      SELECT GROUP# GRP, THREAD# THR, BYTES, 'YES' SRL
      FROM   V$STANDBY_LOG
      ORDER BY 1;
   stmt     VARCHAR2(2048);
BEGIN
   FOR rlcRec IN rlc LOOP
      IF (rlcRec.srl = 'YES') THEN
         stmt := 'ALTER DATABASE ADD STANDBY LOGFILE THREAD ' ||
                 rlcRec.thr || ' SIZE ' || rlcRec.bytes;
         EXECUTE IMMEDIATE stmt;
         stmt := 'ALTER DATABASE DROP STANDBY LOGFILE GROUP ' || rlcRec.grp;
         EXECUTE IMMEDIATE stmt;
      ELSE
         stmt := 'ALTER DATABASE ADD LOGFILE THREAD ' ||
                 rlcRec.thr || ' SIZE ' ||  rlcRec.bytes;
         EXECUTE IMMEDIATE stmt;
         BEGIN
            stmt := 'ALTER DATABASE DROP LOGFILE GROUP ' || rlcRec.grp;
            DBMS_OUTPUT.PUT_LINE(stmt);
            EXECUTE IMMEDIATE stmt;
         EXCEPTION
            WHEN OTHERS THEN
               EXECUTE IMMEDIATE 'ALTER SYSTEM SWITCH LOGFILE';
               EXECUTE IMMEDIATE 'ALTER SYSTEM CHECKPOINT GLOBAL';
               EXECUTE IMMEDIATE stmt;
         END;
      END IF;
   END LOOP;
END;
/
```

11. Optionally, migrate backups and copies in the old flash recovery area to ASM as follows:

a. If foreign archived logs exists in the recovery area, then you cannot migrate them to ASM. Execute the following command at the RMAN prompt:

```
RMAN> DELETE REMOTE ARCHIVELOG ALL;
```

b. Back up archived redo log files, backup sets, and datafile copies to ASM. For example, execute the following command at the RMAN prompt:

```
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev2 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev3 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev4 DEVICE TYPE DISK;

  BACKUP AS COPY ARCHIVELOG ALL DELETE INPUT;
  BACKUP BACKUPSET ALL DELETE INPUT;
  BACKUP AS COPY DATAFILECOPY ALL DELETE INPUT;
}
```

Migrating a Database from ASM to Alternative Storage

Migrating a database from ASM to an alternative storage system is essentially the reverse of the migration to ASM. Modify the steps in "Preparing to Migrate the Database to ASM" on page 26-2 and "Migrating the Database to ASM" on page 26-4 as follows:

- If the procedure specifies Oracle Managed Files locations, then alter the procedure to use locations in alternative storage.
- If the `FORMAT` clause of the `BACKUP` command specifies an ASM location, then change the backup format to an alternative storage location.
- If a filename used in a SQL statement is an ASM location, then change it to a filename in the alternative storage location.

Moving Datafiles Between ASM Disk Groups

You may want to move an active datafile in an `ARCHIVELOG` mode database from one ASM disk group to another. In this case, you use `BACKUP AS COPY` to copy the datafile to the new disk group and `SET NEWNAME` and `SWITCH` commands to rename the datafile in the control file.

For this scenario, assume that you use ASM disk groups `ASMDSK2` and `ASMDSK1`. You want to move datafile `+ASMDSK2/rdbms/datafile/tbs_5.256.565313879` to disk group `ASMDSK1`.

To move a datafile from one ASM disk group to another ASM disk group:

1. Start RMAN and connect to the target database.
2. Generate a report that shows the names of the datafiles.

   For example, execute the following `REPORT` command after connecting RMAN to the target database. Note both the datafile number and the datafile name of the file to be moved.

   ```
   REPORT SCHEMA;
   ```

3. Back up the datafile to the new ASM disk group.

   For example, issue the following `BACKUP AS COPY` command to back up the datafile on `ASMDSK2` to `ASMDSK1`:

   ```
   BACKUP AS COPY
     DATAFILE "+ASMDSK2/rdbms/datafile/tbs_5.256.565313879"
     FORMAT   "+ASMDSK1";
   ```

   You could also specify the datafile by datafile number as in the following example:

   ```
   BACKUP AS COPY
     DATAFILE 23
     FORMAT   "+ASMDSK1";
   ```

4. Find the name of the datafile that you intend to move to a new disk group and take it offline.

   For example, execute the following `SQL` command in the RMAN client. Note that you use two single quotes around the name of the datafile:

   ```
   SQL "ALTER DATABASE DATAFILE
     ' ' +ASMDSK2/rdbms/datafile/tbs_5.256.565313879 ' ' OFFLINE";
   ```

5. Point the control file to the newly created copy of the datafile.

   For example, run the `SWITCH ... TO COPY` command in the RMAN client as follows. The `TO COPY` option of `SWITCH` switches the datafile to the most recent copy of the datafile. You can specify the datafile by name or number.

   ```
   SWITCH DATAFILE "+ASMDSK2/rdbms/datafile/tbs_5.256.565313879" TO COPY;
   ```

   The output of this command will display the new name of the datafile.

6. Recover the renamed datafile.

   For example, run the `RECOVER` command in the RMAN client as follows. Note that you can specify the datafile by name or number.

   ```
   RECOVER DATAFILE "+ASMDSK1/rdbms/datafile/tbs_5.256.603733209";
   ```

7. Bring the datafile online.

   For example, execute the `SQL` command in the RMAN client as follows:

   ```
   SQL "ALTER DATABASE DATAFILE
     ' ' +ASMDSK1/rdbms/datafile/tbs_5.256.603733209 ' ' ONLINE";
   ```

8. Delete the datafile copy from the original ASM disk group.

   In this scenario, `+ASMDSK2/rdbms/datafile/tbs_5.256.565313879` is the original datafile in `ASMDSK2`. Because you issued `SET NEWNAME` and `SWITCH` commands for this datafile, the original file is now recorded in the RMAN repository as a datafile copy. You can execute a `DELETE` command in the RMAN client as follows to remove this file:

   ```
   DELETE DATAFILECOPY "+ASMDSK2/rdbms/datafile/tbs_5.256.603733209";
   ```

Part VIII

Performing User-Managed Backup and Recovery

The following chapters describe how to perform backup and recovery when using a user-managed backup and recovery strategy, that is, one that does not depend upon RMAN. This part of the book contains these chapters:

- Chapter 27, "Making User-Managed Database Backups"
- Chapter 28, "Performing User-Managed Database Flashback and Recovery"
- Chapter 29, "Performing User-Managed Recovery: Advanced Scenarios"

27

Making User-Managed Database Backups

This chapter describes methods of backing up an Oracle database in a user-managed backup and recovery strategy, that is, a strategy that does not depend on using Recovery Manager (RMAN).

This chapter contains the following sections:

- Querying V$ Views to Obtain Backup Information
- Making User-Managed Backups of the Whole Database
- Making User-Managed Backups of Tablespaces and Datafiles
- Making User-Managed Backups of the Control File
- Making User-Managed Backups of Archived Redo Logs
- Making User-Managed Backups in SUSPEND Mode
- Making User-Managed Backups to Raw Devices
- Making Backups with the Volume Shadow Copy Service (VSS)
- Verifying User-Managed Datafile Backups

Querying V$ Views to Obtain Backup Information

Before making a backup, you must identify all the files in your database and decide what to back up. You can use `V$` views to obtain this information.

Listing Database Files Before a Backup

Use `V$DATAFILE` and `V$CONTROLFILE` to identify the datafiles and control files for your database. This same procedure works whether you named these files manually or allowed Oracle Managed Files to name them.

> **Caution:** Never back up online redo log files.

To list datafiles and control files:

1. Start SQL*Plus and query `V$DATAFILE` to obtain a list of datafiles. For example, enter:

   ```
   SELECT NAME FROM V$DATAFILE;
   ```

   You can also join the `V$TABLESPACE` and `V$DATAFILE` views to obtain a listing of datafiles along with their associated tablespaces:

```
SELECT   t.NAME "Tablespace", f.NAME "Datafile"
FROM     V$TABLESPACE t, V$DATAFILE f
WHERE    t.TS# = f.TS#
ORDER BY t.NAME;
```

2. Obtain the filenames of the current control files by querying the V$CONTROLFILE view. For example, issue the following query:

   ```
   SELECT NAME FROM V$CONTROLFILE;
   ```

   Note that you only need to back up one copy of a multiplexed control file.

3. If you plan to take a control file backup with the ALTER DATABASE BACKUP CONTROLFILE TO '*filename*' statement, then save a list of all datafiles and online redo log files with the control file backup. Because the current database structure may not match the database structure at the time a given control file backup was created, saving a list of files recorded in the backup control file can aid the recovery procedure.

Determining Datafile Status for Online Tablespace Backups

To check whether a datafile is part of a current online tablespace backup, query the V$BACKUP view.

This view is useful only for user-managed online tablespace backups, because neither RMAN backups nor offline tablespace backups require the datafiles of a tablespace to be in **backup mode**. Some user-managed backup procedures require you to place the tablespace in backup mode to protect against the possibility of a fractured block. However, updates to the database create more than the usual amount of redo in backup mode.

The V$BACKUP view is most useful when the database is open. It is also useful immediately after an instance failure because it shows the backup status of the files at the time of the failure. Use this information to determine whether you have left any tablespaces in backup mode.

V$BACKUP is not useful if the control file currently in use is a restored backup or a new control file created after the media failure occurred. A restored or re-created control file does not contain the information the database needs to populate V$BACKUP accurately. Also, if you have restored a backup of a file, this file's STATUS in V$BACKUP reflects the backup status of the older version of the file, not the most current version. Thus, this view can contain misleading data about restored files.

For example, the following query displays which datafiles are currently included in a tablespace that has been placed in backup mode:

```
SELECT t.name AS "TB_NAME", d.file# as "DF#", d.name AS "DF_NAME", b.status
FROM   V$DATAFILE d, V$TABLESPACE t, V$BACKUP b
WHERE  d.TS#=t.TS#
AND    b.FILE#=d.FILE#
AND    b.STATUS='ACTIVE';
```

The following sample output shows that the tools and users tablespaces currently have ACTIVE status:

```
TB_NAME                DF#        DF_NAME                          STATUS
---------------------  ---------- -------------------------------  ------
TOOLS                  7          /oracle/oradata/trgt/tools01.dbf  ACTIVE
USERS                  8          /oracle/oradata/trgt/users01.dbf  ACTIVE
```

In the STATUS column, NOT ACTIVE indicates that the file is not currently in backup mode (that is, you have not executed the ALTER TABLESPACE . . . BEGIN BACKUP or ALTER DATABASE BEGIN BACKUP statement), whereas ACTIVE indicates that the file is currently in backup mode.

Making User-Managed Backups of the Whole Database

You can make a whole database backup of all files in a database after the database has been shut down with the NORMAL, IMMEDIATE, or TRANSACTIONAL options. A whole database backup taken while the database is open or after an instance failure or SHUTDOWN ABORT is inconsistent. In such cases, the files are inconsistent with respect to the **database checkpoint** SCN.

You can make a whole database backup if a database is operating in either ARCHIVELOG or NOARCHIVELOG mode. If you run the database in NOARCHIVELOG mode, however, then the backup must be consistent; that is, you must shut down the database cleanly before the backup.

The set of backup files that results from a consistent whole database backup is consistent because all files are checkpointed to the same SCN. You can restore the consistent database backup without further recovery. After restoring the backup files, you can perform additional recovery steps to recover the database to a more current time if the database is operated in ARCHIVELOG mode. Also, you can take inconsistent whole database backups if your database is in ARCHIVELOG mode.

Control files play a crucial role in database restore and recovery. For databases running in ARCHIVELOG mode, Oracle recommends that you back up control files with the ALTER DATABASE BACKUP CONTROLFILE TO '*filename*' statement.

> **See Also:** "Making User-Managed Backups of the Control File" on page 27-10 for more information about backing up control files

Making Consistent Whole Database Backups

This section describes how to back up the database with an operating system utility.

To make a consistent whole database backup:

1. If the database is open, then use SQL*Plus to shut down the database with the NORMAL, IMMEDIATE, or TRANSACTIONAL options.
2. Use an operating system utility to make backups of all datafiles as well as all control files specified by the CONTROL_FILES parameter of the initialization parameter file. Also, back up the initialization parameter file and other Oracle product initialization files. To find these files, do a search for *.ora starting in your Oracle home directory and recursively search all of its subdirectories.

   For example, you can back up the datafiles, control files and archived logs to /disk2/backup as follows:

   ```
   % cp $ORACLE_HOME/oradata/trgt/*.dbf /disk2/backup
   % cp $ORACLE_HOME/oradata/trgt/arch/* /disk2/backup/arch
   ```

3. Restart the database with the STARTUP command in SQL*Plus.

> **See Also:** *Oracle Database Administrator's Guide* for more information on starting up and shutting down a database

Making User-Managed Backups of Tablespaces and Datafiles

The technique for making user-managed backups of tablespaces and datafiles depends on whether the files are offline or online.

Making User-Managed Backups of Offline Tablespaces and Datafiles

Note the following guidelines when backing up offline tablespaces:

- You cannot offline the `SYSTEM` tablespace or a tablespace with active undo segments. The following technique cannot be used for such tablespaces.
- Assume that a table is in tablespace `Primary` and its index is in tablespace `Index`. Taking tablespace `Index` offline while leaving tablespace `Primary` online can cause errors when DML is issued against the indexed tables located in `Primary`. The problem only manifests when the access method chosen by the optimizer needs to access the indexes in the `Index` tablespace.

To back up offline tablespaces:

1. Before beginning a backup of a tablespace, identify the tablespace's datafiles by querying the `DBA_DATA_FILES` view. For example, assume that you want to back up the `users` tablespace. Enter the following statement in SQL*Plus:

```
SELECT TABLESPACE_NAME, FILE_NAME
  FROM SYS.DBA_DATA_FILES
  WHERE TABLESPACE_NAME = 'USERS';

TABLESPACE_NAME                FILE_NAME
------------------------------ --------------------------------
USERS                          /oracle/oradata/trgt/users01.dbf
```

   In this example, `/oracle/oradata/trgt/users01.dbf` is a fully specified filename corresponding to the datafile in the `users` tablespace.

2. Take the tablespace offline using normal priority if possible because it guarantees that you can subsequently bring the tablespace online without having to recover it. For example:

```
SQL> ALTER TABLESPACE users OFFLINE NORMAL;
```

3. Back up the offline datafiles. For example:

```
% cp /oracle/oradata/trgt/users01.dbf /d2/users01_'date "+%m_%d_%y"'.dbf
```

4. Bring the tablespace online. For example:

```
ALTER TABLESPACE users ONLINE;
```

   Note: If you took the tablespace offline using temporary or immediate priority, then you cannot bring the tablespace online unless you perform tablespace recovery.

5. Archive the unarchived redo logs so that the redo required to recover the tablespace backup is archived. For example, enter:

```
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

Making User-Managed Backups of Online Tablespaces and Datafiles

You can back up all or only specific datafiles of an online tablespace while the database is open. The procedure differs depending on whether the online tablespace is read/write or read-only.

> **Note:** You should not back up temporary tablespaces.

Making User-Managed Backups of Online Read/Write Tablespaces

You must put a read/write tablespace in backup mode to make user-managed datafile backups when the tablespace is online and the database is open. The `ALTER TABLESPACE ... BEGIN BACKUP` statement places a tablespace in backup mode. In backup mode, the database copies whole changed data blocks into the redo stream. After you take the tablespace out of backup mode with the `ALTER TABLESPACE ... END BACKUP` or `ALTER DATABASE END BACKUP` statement, the database advances the **datafile checkpoint** SCN to the current **database checkpoint** SCN.

When restoring a datafile backed up in this way, the database asks for the appropriate set of redo log files to apply if recovery be needed. The redo logs contain all changes required to recover the datafiles and make them consistent.

To back up online read/write tablespaces in an open database:

1. Before beginning a backup of a tablespace, identify all of the datafiles in the tablespace with the `DBA_DATA_FILES` data dictionary view. For example, assume that you want to back up the `users` tablespace. Enter the following:

```
SELECT TABLESPACE_NAME, FILE_NAME
FROM   SYS.DBA_DATA_FILES
WHERE  TABLESPACE_NAME = 'USERS';

TABLESPACE_NAME                FILE_NAME
------------------------------ --------------------
USERS                          /oracle/oradata/trgt/users01.dbf
USERS                          /oracle/oradata/trgt/users02.dbf
```

2. Mark the beginning of the online tablespace backup. For example, the following statement marks the start of an online backup for the tablespace `users`:

```
SQL> ALTER TABLESPACE users BEGIN BACKUP;
```

> **Caution:** If you do not use `BEGIN BACKUP` to mark the beginning of an online tablespace backup and wait for this statement to complete before starting your copies of online tablespaces, then the datafile copies produced are not usable for subsequent recovery operations. Attempting to recover such a backup is risky and can return errors that result in inconsistent data. For example, the attempted recovery operation can issue a **fuzzy file** warning, and can lead to an inconsistent database that you cannot open.

3. Back up the online datafiles of the online tablespace with operating system commands. For example, Linux and UNIX users might enter:

```
% cp /oracle/oradata/trgt/users01.dbf /d2/users01_'date "+%m_%d_%y"'.dbf
% cp /oracle/oradata/trgt/users02.dbf /d2/users02_'date "+%m_%d_%y"'.dbf
```

4. After backing up the datafiles of the online tablespace, run the SQL statement ALTER TABLESPACE with the END BACKUP option. For example, the following statement ends the online backup of the tablespace users:

```
SQL> ALTER TABLESPACE users END BACKUP;
```

5. Archive the unarchived redo logs so that the redo required to recover the tablespace backup is archived. For example, enter:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

Caution: If you fail to take the tablespace out of backup mode, then Oracle Database continues to write copies of data blocks in this tablespace to the online redo logs, causing performance problems. Also, you receive an ORA-01149 error if you try to shut down the database with the tablespaces still in backup mode.

Making Multiple User-Managed Backups of Online Read/Write Tablespaces

When backing up several online tablespaces, you can back them up either serially or in parallel. Use either of the following procedures depending on your needs.

Backing Up Online Tablespaces in Parallel You can simultaneously create datafile copies of multiple tablespaces requiring backups in backup mode. Note, however, that by putting all tablespaces in online mode at once, you can generate large redo logs if there is heavy update activity on the affected tablespaces, because the redo must contain a copy of each changed data block in each changed datafile. Be sure to consider the size of the likely redo before using the procedure outlined here.

To back up online tablespaces in parallel:

1. Prepare the online tablespaces for backup by issuing all necessary ALTER TABLESPACE statements at once. For example, put tablespaces users, tools, and indx in backup mode as follows:

```
SQL> ALTER TABLESPACE users BEGIN BACKUP;
SQL> ALTER TABLESPACE tools BEGIN BACKUP;
SQL> ALTER TABLESPACE indx BEGIN BACKUP;
```

   If you are backing up all tablespaces, you might want to use this command:

```
SQL> ALTER DATABASE BEGIN BACKUP;
```

2. Back up all files of the online tablespaces. For example, a Linux or UNIX user might back up datafiles with the *.dbf suffix as follows:

```
% cp $ORACLE_HOME/oradata/trgt/*.dbf /disk2/backup/
```

3. Take the tablespaces out of backup mode as in the following example:

```
SQL> ALTER TABLESPACE users END BACKUP;
SQL> ALTER TABLESPACE tools END BACKUP;
SQL> ALTER TABLESPACE indx END BACKUP;
```

   Again, it you are handling all datafiles at once you can use the ALTER DATABASE command instead of ALTER TABLESPACE:

```
SQL> ALTER DATABASE END BACKUP;
```

4. Archive the online redo logs so that the redo required to recover the tablespace backups will be available for later media recovery. For example, enter:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

Backing Up Online Tablespaces Serially You can place all tablespaces requiring online backups in backup mode one at a time. Oracle recommends the serial backup option because it minimizes the time between ALTER TABLESPACE . . . BEGIN/END BACKUP statements. During online backups, more redo information is generated for the tablespace because whole data blocks are copied into the redo log.

To back up online tablespaces serially:

1. Prepare a tablespace for online backup. For example, to put tablespace users in backup mode enter the following:

```
SQL> ALTER TABLESPACE users BEGIN BACKUP;
```

   In this case you probably do not want to use ALTER DATABASE BEGIN BACKUP to put all tablespaces in backup mode simultaneously, because of the unnecessary volume of redo log information generated for tablespaces in online mode.

2. Back up the datafiles in the tablespace. For example, enter:

```
% cp /oracle/oradata/trgt/users01.dbf /d2/users01_'date "+%m_%d_%y"'.dbf
```

3. Take the tablespace out of backup mode. For example, enter:

```
SQL> ALTER TABLESPACE users END BACKUP;
```

4. Repeat this procedure for each remaining tablespace.
5. Archive the unarchived redo logs so that the redo required to recover the tablespace backups is archived. For example, enter:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

Ending a Backup After an Instance Failure or SHUTDOWN ABORT

The following situations can cause a tablespace backup to fail and be incomplete:

- The backup completed, but you did not run the ALTER TABLESPACE . . . END BACKUP statement.
- An instance failure or SHUTDOWN ABORT interrupted the backup.

Whenever crash recovery is required, if a datafile is in backup mode when an attempt is made to open it, then the database will not open the database until either a recovery command is issued, or the datafile is taken out of backup mode.

For example, the database may display a message such as the following at startup:

```
ORA-01113: file 12 needs media recovery
ORA-01110: data file 12: '/oracle/dbs/tbs_41.f'
```

If the database indicates that the datafiles for multiple tablespaces require media recovery because you forgot to end the online backups for these tablespaces, then so long as the database is mounted, running the ALTER DATABASE END BACKUP statement takes all the datafiles out of backup mode simultaneously.

In high availability situations, and in situations when no DBA is monitoring the database, the requirement for user intervention is intolerable. Hence, you can write a crash recovery script that does the following:

1. Mounts the database
2. Runs the ALTER DATABASE END BACKUP statement
3. Runs ALTER DATABASE OPEN, allowing the system to come up automatically

An automated crash recovery script containing ALTER DATABASE END BACKUP is especially useful in the following situations:

- All nodes in an Oracle Real Application Clusters (Oracle RAC) configuration fail.
- One node fails in a **cold failover cluster** (that is, a cluster that is *not* a RAC configuration in which the secondary node must mount and recover the database when the first node fails).

Alternatively, you can take the following manual measures after the system fails with tablespaces in backup mode:

- Recover the database and avoid issuing END BACKUP statements altogether.
- Mount the database, then run ALTER TABLESPACE . . . END BACKUP for each tablespace still in backup mode.

Ending Backup Mode with the ALTER DATABASE END BACKUP Statement You can run the ALTER DATABASE END BACKUP statement when you have multiple tablespaces still in backup mode. The primary purpose of this command is to allow a crash recovery script to restart a failed system without DBA intervention. You can also perform the following procedure manually.

To take tablespaces out of backup mode simultaneously:

1. Mount but do not open the database. For example, enter:

```
SQL> STARTUP MOUNT
```

2. If performing this procedure manually (that is, not as part of a crash recovery script), query the V$BACKUP view to list the datafiles of the tablespaces that were being backed up before the database was restarted:

```
SQL>  SELECT * FROM V$BACKUP WHERE STATUS = 'ACTIVE';
FILE#      STATUS             CHANGE#    TIME
---------- ------------------ ---------- ---------
        12 ACTIVE                  20863 25-NOV-02
        13 ACTIVE                  20863 25-NOV-02
        20 ACTIVE                  20863 25-NOV-02
 3 rows selected.
```

3. Issue the ALTER DATABASE END BACKUP statement to take all datafiles currently in backup mode out of backup mode. For example, enter:

```
SQL> ALTER DATABASE END BACKUP;
```

   You can use this statement only when the database is mounted but not open. If the database is open, then use ALTER TABLESPACE . . . END BACKUP or ALTER DATABASE DATAFILE . . . END BACKUP for each affected tablespace or datafile.

> **Caution:** Do not use ALTER DATABASE END BACKUP if you have restored any of the affected files from a backup.

Ending Backup Mode with the SQL*Plus RECOVER Command The ALTER DATABASE END BACKUP statement is not the only way to respond to a failed online backup: you can

also run the SQL*Plus RECOVER command. This method is useful when you are not sure whether someone has restored a backup, because if someone has indeed restored a backup, then the RECOVER command brings the backup up to date. Only run the ALTER DATABASE END BACKUP or ALTER TABLESPACE . . . END BACKUP statement if you are sure that the files are current.

> **Note:** The RECOVER command method is slow because the database must scan redo generated from the beginning of the online backup.

To take tablespaces out of backup mode with the RECOVER command:

1. Mount the database. For example, enter:

```
SQL> STARTUP MOUNT
```

2. Recover the database as normal. For example, enter:

```
SQL> RECOVER DATABASE
```

3. Use the V$BACKUP view to confirm that there are no active datafiles:

```
SQL>  SELECT * FROM V$BACKUP WHERE STATUS = 'ACTIVE';
FILE#      STATUS             CHANGE#    TIME
---------- ------------------ ---------- ---------
0 rows selected.
```

> **See Also:** Chapter 28, "Performing User-Managed Database Flashback and Recovery" for information on recovering a database

Making User-Managed Backups of Read-Only Tablespaces

When backing up an online read-only tablespace, you can simply back up the online datafiles. You do not have to place the tablespace in backup mode because the database is not permitting changes to the datafiles.

If the set of read-only tablespaces is self-contained, then in addition to backing up the tablespaces with operating system commands, you can also export the tablespace metadata with the transportable tablespace functionality. In the event of a media error or a user error (such as accidentally dropping a table in the read-only tablespace), you can transport the tablespace back into the database.

> **See Also:** *Oracle Database Administrator's Guide* to learn how to transport tablespaces

To back up online read-only tablespaces in an open database:

1. Query the DBA_TABLESPACES view to determine which tablespaces are read-only. For example, run this query:

```
SELECT TABLESPACE_NAME, STATUS
FROM DBA_TABLESPACES
WHERE STATUS = 'READ ONLY';
```

2. Before beginning a backup of a read-only tablespace, identify all of the tablespace's datafiles by querying the DBA_DATA_FILES data dictionary view. For example, assume that you want to back up the history tablespace:

```
SELECT TABLESPACE_NAME, FILE_NAME
FROM SYS.DBA_DATA_FILES
WHERE TABLESPACE_NAME = 'HISTORY';
```

```
TABLESPACE_NAME                FILE_NAME
------------------------------ --------------------
HISTORY                        /oracle/oradata/trgt/history01.dbf
HISTORY                        /oracle/oradata/trgt/history02.dbf
```

3. Back up the online datafiles of the read-only tablespace with operating system commands. You do not have to take the tablespace offline or put the tablespace in backup mode because users are automatically prevented from making changes to the read-only tablespace. For example:

```
% cp $ORACLE_HOME/oradata/trgt/history*.dbf  /disk2/backup/
```

> **Note:** When restoring a backup of a read-only tablespace, take the tablespace offline, restore the datafiles, then bring the tablespace online. A backup of a read-only tablespace is still usable if the read-only tablespace is made read/write after the backup, but the restored backup will require recovery.

4. Optionally, export the metadata in the read-only tablespace. By using the transportable tablespace feature, you can quickly restore the datafiles and import the metadata in case of media failure or user error. For example, export the metadata for tablespace `history` as follows:

```
% expdp DIRECTORY=dpump_dir1 DUMPFILE=hs.dmp TRANSPORT_TABLESPACES=history
  LOGFILE=tts.log
```

> **See Also:** *Oracle Database Reference* for more information about the `DBA_DATA_FILES` and `DBA_TABLESPACES` views

Making User-Managed Backups of the Control File

Back up the control file of a database after making a structural modification to a database operating in `ARCHIVELOG` mode. To back up a database's control file, you must have the `ALTER DATABASE` system privilege.

Backing Up the Control File to a Binary File

The primary method for backing up the control file is to use a SQL statement to generate a binary file. A binary backup is preferable to a trace file backup because it contains additional information such as the archived log history, offline range for read-only and offline tablespaces, and backup sets and copies (if you use RMAN). If `COMPATIBLE` is 10.2 or higher, binary control file backups include tempfile entries.

To back up the control file after a structural change:

1. Make the desired change to the database. For example, you may create a new tablespace:

```
CREATE TABLESPACE tbs_1 DATAFILE 'file_1.f' SIZE 10M;
```

2. Back up the database's control file, specifying a filename for the output binary file. The following example backs up a control file to `/disk1/backup/cf.bak`:

```
ALTER DATABASE BACKUP CONTROLFILE TO '/disk1/backup/cf.bak' REUSE;
```

   Specify `REUSE` to make the new control file overwrite one that currently exists.

Backing Up the Control File to a Trace File

You can back up the control file to a text file that contains a CREATE CONTROLFILE statement. You can edit the trace file to create a script that creates a new control file based on the control file that was current when you created the trace file.

If you specify neither the RESETLOGS nor NORESETLOGS option in the SQL statement, then the resulting trace file contains versions of the control file for both RESETLOGS and NORESETLOGS options. Tempfile entries are included in the output using ALTER TABLESPACE ... ADD TEMPFILE statements.

To avoid recovering offline normal or read-only tablespaces, edit them out of the CREATE CONTROLFILE statement. When you open the database with the re-created control file, the database marks these omitted files as MISSING. You can run an ALTER DATABASE RENAME FILE statement to rename them to their original filenames.

The trace file containing the CREATE CONTROLFILE statement is stored in a subdirectory determined by the DIAGNOSTIC_DEST initialization parameter. You can look in the database alert log for the name and location of the trace file to which the CREATE CONTROLFILE statement was written. See *Oracle Database Administrator's Guide* to learn how to locate the alert log.

To back up the control file to a trace file:

1. Mount or open the database.
2. Execute the following SQL statement:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

> **See Also:** "Recovery of Read-Only Files with a Re-Created Control File" on page 29-8 for special issues relating to read-only, offline normal, and temporary files included in CREATE CONTROLFILE statements

Making User-Managed Backups of Archived Redo Logs

To save disk space in your primary archiving location, you may want to back up archived logs to tape or to an alternative disk location. If you archive to multiple locations, then only back up one copy of each log sequence number.

To back up archived redo logs:

1. To determine which archived redo log files that the database has generated, query V$ARCHIVED_LOG. For example, run the following query:

```
SELECT THREAD#,SEQUENCE#,NAME
FROM V$ARCHIVED_LOG;
```

2. Back up one copy of each log sequence number by using an operating system utility. This example backs up all logs in the primary archiving location to a disk devoted to log backups:

```
% cp $ORACLE_HOME/oracle/trgt/arch/* /disk2/backup/arch
```

> **See Also:** *Oracle Database Reference* for more information about the data dictionary views

Making User-Managed Backups in SUSPEND Mode

This section contains the following topics:

- About the Suspend/Resume Feature
- Making Backups in a Suspended Database

About the Suspend/Resume Feature

Some third-party tools allow you to mirror a set of disks or logical devices, that is, maintain an exact duplicate of the primary data in another location, and then **split the mirror**. Splitting the mirror involves separating the copies so that you can use them independently.

With the SUSPEND/RESUME functionality, you can suspend I/O to the database, then split the mirror and make a backup of the split mirror. By using this feature, which complements the backup mode functionality, you can suspend database I/Os so that no new I/O can be performed. You can then access the suspended database to make backups without I/O interference.

You do not need to use SUSPEND/RESUME to make split mirror backups in most cases, although it is necessary if your system requires the database cache to be free of **dirty buffers** before a volume can be split. Some RAID devices benefit from suspending writes while the split operation is occurring; your RAID vendor can advise you on whether your system would benefit from this feature.

The ALTER SYSTEM SUSPEND statement suspends the database by halting I/Os to datafile headers, datafiles, and control files. When the database is suspended, all pre-existing I/O operations can complete; however, any new database I/O access attempts are queued.

The ALTER SYSTEM SUSPEND and ALTER SYSTEM RESUME statements operate on the database and not just the instance. If the ALTER SYSTEM SUSPEND statement is entered on one system in a RAC configuration, then the internal locking mechanisms propagate the halt request across instances, thereby suspending I/O operations for all active instances in a given cluster.

Making Backups in a Suspended Database

After a successful database suspension, you can back up the database to disk or break the mirrors. Because suspending a database does not guarantee immediate termination of I/O, Oracle recommends that you precede the ALTER SYSTEM SUSPEND statement with a BEGIN BACKUP statement so that the tablespaces are placed in backup mode.

You must use conventional user-managed backup methods to back up split mirrors. RMAN cannot make database backups or copies because these operations require reading the datafile headers. After the database backup is finished or the mirrors are re-silvered, then you can resume normal database operations using the ALTER SYSTEM RESUME statement.

Backing up a suspended database without splitting mirrors can cause an extended database outage because the database is inaccessible during this time. If backups are taken by splitting mirrors, however, then the outage is nominal. The outage time depends on the size of cache to flush, the number of datafiles, and the time required to break the mirror.

Note the following restrictions for the SUSPEND/RESUME feature:

- In a RAC configuration, you should not start a new instance while the original nodes are suspended.
- No checkpoint is initiated by the ALTER SYSTEM SUSPEND or ALTER SYSTEM RESUME statements.

- You cannot issue SHUTDOWN with IMMEDIATE, NORMAL, or TRANSACTIONAL options while the database is suspended.
- Issuing SHUTDOWN ABORT on a database that was already suspended reactivates the database. This prevents media recovery or crash recovery from hanging.

To make a split mirror backup in SUSPEND mode:

1. Place the database tablespaces in backup mode. For example, to place tablespace users in backup mode enter:

   ```
   ALTER TABLESPACE users BEGIN BACKUP;
   ```

   If you are backing up all of the tablespaces for your database, you can instead use:

   ```
   ALTER DATABASE BEGIN BACKUP;
   ```

2. If your mirror system has problems with splitting a mirror while disk writes are occurring, then suspend the database. For example, issue the following:

   ```
   ALTER SYSTEM SUSPEND;
   ```

3. Check to make sure that the database is suspended by querying V$INSTANCE. For example:

   ```
   SELECT DATABASE_STATUS FROM V$INSTANCE;

   DATABASE_STATUS
   -----------------
   SUSPENDED
   ```

4. Split the mirrors at the operating system or hardware level.
5. End the database suspension. For example, issue the following statement:

   ```
   ALTER SYSTEM RESUME;
   ```

6. Check to make sure that the database is active by querying V$INSTANCE. For example, enter:

   ```
   SELECT DATABASE_STATUS FROM V$INSTANCE;

   DATABASE_STATUS
   -----------------
   ACTIVE
   ```

7. Take the specified tablespaces out of backup mode. For example, enter the following to take tablespace users out of backup mode:

   ```
   ALTER TABLESPACE users END BACKUP;
   ```

8. Copy the control file and archive the online redo logs as usual for a backup.

Caution: Do not use the ALTER SYSTEM SUSPEND statement as a substitute for placing a tablespace in backup mode.

See Also:

- "Making Split Mirror Backups with RMAN" on page 9-8
- *Oracle Database Administrator's Guide* for more information about the `SUSPEND/RESUME` feature
- *Oracle Database SQL Language Reference* for the `ALTER SYSTEM SUSPEND` syntax

Making User-Managed Backups to Raw Devices

A **raw device** is a disk or partition that does not have a file system. In other words, a raw device can contain only a single file. Backing up files on raw devices poses operating system specific issues. The following sections discuss some of these issues on UNIX, Linux, and Windows.

Backing Up to Raw Devices on Linux and UNIX

When backing up to or from raw devices, the `dd` command on Linux and UNIX is the most common backup utility. See your operating system-specific documentation for complete details about this utility.

Using `dd` effectively requires specifying the correct options, based on your database. Table 27–1 lists details about your database that affect the options you use for `dd`.

Table 27–1 Aspects of the Database Important for dd Usage

Data	Explanation
Block size	You can specify the size of the buffer that `dd` uses to copy data. For example, you can specify that `dd` should copy data in units of 8 KB or 64 KB. Note that the block size for `dd` need not correspond to either the Oracle block size or the operating system block size: it is merely the size of the buffer used by `dd` when making the copy.
Raw offset	On some systems, the beginning of the file on the raw device is reserved for use by the operating system. This storage space is called the **raw offset**. Oracle should not back up or restore these bytes.
Size of Oracle block `0`	At the beginning of every Oracle file, the operating system-specific code places an Oracle block called **block 0**. The generic Oracle code does not recognize this block, but the block is included in the size of the file on the operating system. Typically, this block is the same size as the other Oracle blocks in the file.

The information in Table 27–1 enables you to set the `dd` options specified in Table 27–2.

Table 27–2 Options for dd Command

This option ...	Specifies ...
`if`	The name of the input file, that is, the file that you are reading.
`of`	The name of the output file, that is, the file to which you are writing.
`bs`	The buffer size used by `dd` to copy data.
`skip`	The number of `dd` buffers to skip on the input raw device if a raw offset exists. For example, if you are backing up a file on a raw device with a 64 KB raw offset, and the `dd` buffer size is 8 KB, then you can specify `skip=8` so that the copy starts at offset 64 KB.

Table 27–2 (Cont.) Options for dd Command

This option ...	Specifies ...
`seek`	The number of `dd` buffers to skip on the output raw device if a raw offset exists. For example, if you are backing up a file onto a raw device with a 64 KB raw offset, and the `dd` buffer size is 8 KB, then you can specify `skip=8` so that the copy starts at offset 64 KB.
`count`	The number of blocks on the input raw device for `dd` to copy. It is best to specify the exact number of blocks to copy when copying from raw device to file system, otherwise any extra space at the end of the raw volume that is not used by the Oracle datafile is copied to the file system. Remember to include block `0` in the total size of the input file. For example, if the `dd` block size is 8 KB, and you are backing up a 30720 KB datafile, then you can set `count=3841`. This value for `count` actually backs up 30728 KB: the extra 8 KB are for Oracle block `0`.

Because a raw device can be the input or output device for a backup, you have four possible scenarios for the backup. The possible options for `dd` depend on which scenario you choose, as illustrated in Table 27–3.

Table 27–3 Scenarios Involving dd Backups

Backing Up from ...	Backing Up to ...	Options Specified for dd Command
Raw device	Raw device	`if`, `of`, `bs`, `skip`, `seek`, `count`
Raw device	File system	`if`, `of`, `bs`, `skip`, `count`
File system	Raw device	`if`, `of`, `bs`, `seek`
File system	File system	`if`, `of`, `bs`

Backing Up with the dd utility on Linux and UNIX: Examples

For these examples of `dd` utility usage, assume the following:

- You are backing up a 30720 KB datafile.
- The beginning of the datafile has a block 0 of 8 KB.
- The raw offset is 64 KB.
- You set the `dd` block size to 8 KB when a raw device is involved in the copy.

In the following example, you back up from one raw device to another raw device:

```
% dd if=/dev/rsd1b of=/dev/rsd2b bs=8k skip=8 seek=8 count=3841
```

In the following example, you back up from a raw device to a file system:

```
% dd if=/dev/rsd1b of=/backup/df1.dbf bs=8k skip=8 count=3841
```

In the following example, you back up from a file system to a raw device:

```
% dd if=/backup/df1.dbf of=/dev/rsd2b bs=8k seek=8
```

In the following example, you back up from a file system to a file system, and so can set the block size to a high value to boost I/O performance:

```
% dd if=/oracle/dbs/df1.dbf of=/backup/df1.dbf bs=1024k
```

Backing Up to Raw Devices on Windows

Like Linux and UNIX, Windows supports raw disk partitions in which the database can store datafiles, online logs, and control files. Each raw partition is assigned either a drive letter or physical drive number and does not contain a file system. As in Linux and UNIX, each raw partition on Windows is mapped to a single file.

Windows differs from Linux and UNIX in the naming convention for Oracle files. On Windows, raw datafile names are formatted as follows:

```
\\.\drive_letter:
\\.\PHYSICALDRIVEdrive_number
```

For example, the following are possible raw filenames:

```
\\.\G:
\\.\PHYSICALDRIVE3
```

Note that you can also create aliases to raw filenames. The standard Oracle database installation provides a `SETLINKS` utility that can create aliases such as `\\.\Datafile12` that point to filenames such as `\\.\PHYSICALDRIVE3`.

The procedure for making user-managed backups of raw datafiles is basically the same as for copying files on an Windows file system, except that you should use the Oracle `OCOPY` utility rather than the Windows-supplied `copy.exe` or `ntbackup.exe` utilities. `OCOPY` supports 64-bit file I/O, physical raw drives, and raw files. Note that `OCOPY` cannot back up directly to tape.

To display online documentation for `OCOPY`, enter `OCOPY` by itself at the Windows prompt. Sample output follows:

```
Usage of OCOPY:
     ocopy from_file [to_file [a | size_1 [size_n]]]
     ocopy -b from_file to_drive
     ocopy -r from_drive to_dir
```

Note the important `OCOPY` options described in the following table.

Table 27–4 OCOPY Options

Option	Action
`b`	Splits the input file into multiple output files. This option is useful for backing up to devices that are smaller than the input file.
`r`	Combines multiple input files and writes to a single output file. This option is useful for restoring backups created with the `-b` option.

Backing Up with OCOPY: Example

In this example, assume the following:

- Datafile `12` is mounted on the `\\.\G:` raw partition.
- The `C:` drive mounts a file system.
- The database is open.

To back up the datafile on the raw partition `\\.\G:` to a local file system, you can run the following command at the prompt after placing datafile `12` in backup mode:

```
OCOPY "\\.G:" C:\backup\datafile12.bak
```

Specifying the -b and -r Options for OCOPY: Example

In this example, assume the following:

- `\\.\G:` is a raw partition containing datafile 7
- The `E:` drive is a removable disk drive.
- The database is open.

To back up the datafile onto drive `E:`, you can execute the following command at the Windows prompt after placing datafile 7 in backup mode:

```
# first argument is filename, second argument is drive
OCOPY -b "\\.\G:" E:\
```

When drive `E:` fills up, you can use another disk. In this way, you can divide the backup of datafile 7 into multiple files.

Similarly, to restore the backup, take the tablespace containing datafile 7 offline and run this command:

```
# first argument is drive, second argument is directory
OCOPY -r E:\ "\\.\G:"
```

Making Backups with the Volume Shadow Copy Service (VSS)

Volume Shadow Copy Service (VSS) is a set of Windows APIs that enable applications to create consistent snapshots called **shadow copies**. The **Oracle VSS writer** runs as a service on Windows systems and is integrated with VSS-enabled applications. You can use these applications to create snapshots of database files managed by the Oracle instance. For example, you can make shadow copies of an Oracle database while it is open read/write.

> **See Also:** *Oracle Database Platform Guide for Microsoft Windows* to learn how to back up and recover the database with VSS-enabled applications

Verifying User-Managed Datafile Backups

You should periodically verify your backups to ensure that they are usable for recovery.

Testing the Restore of Datafile Backups

The best way to test the usability of datafile backups is to restore them to a separate host and attempt to open the database, performing media recovery if necessary. This option requires that you have a separate host available for the restore procedure.

> **See Also:** "Performing Complete Database Recovery" on page 28-7 to learn how to recover files with SQL*Plus

Running the DBVERIFY Utility

The DBVERIFY program is an external command-line utility that performs a physical data structure integrity check on an offline datafile. Use DBVERIFY primarily when you need to ensure that a user-managed backup of a datafile is valid before it is restored or as a diagnostic aid when you have encountered data corruption problems.

The name and location of DBVERIFY is dependent on your operating system. For example, to perform an integrity check on datafile `users01.dbf` on Linux or UNIX, run the `dbv` command as follows:

```
% dbv file=users01.dbf
```

Sample `dbv` output follows:

```
DBVERIFY - Verification starting : FILE = users01.dbf


DBVERIFY - Verification complete

Total Pages Examined         : 250
Total Pages Processed (Data) : 1
Total Pages Failing   (Data) : 0
Total Pages Processed (Index): 0
Total Pages Failing   (Index): 0
Total Pages Processed (Other): 2
Total Pages Processed (Seg)  : 0
Total Pages Failing   (Seg)  : 0
Total Pages Empty            : 247
Total Pages Marked Corrupt   : 0
Total Pages Influx           : 0
```

See Also: *Oracle Database Utilities* to learn about DBVERIFY

28

Performing User-Managed Database Flashback and Recovery

This chapter describes how to restore and recover a database and use the flashback features of Oracle when using a user-managed backup and recovery strategy, that is, a a strategy that does not depend on RMAN.

This chapter includes the following topics:

- Performing Flashback Database with SQL*Plus
- Overview of User-Managed Media Recovery
- Performing Complete Database Recovery
- Performing Incomplete Database Recovery
- Recovering a Database in NOARCHIVELOG Mode
- Troubleshooting Media Recovery

Performing Flashback Database with SQL*Plus

Oracle Flashback Database returns your entire database to a previous state without requiring you to restore files from backup. The SQL*Plus `FLASHBACK DATABASE` command performs the same function as the RMAN `FLASHBACK DATABASE` command: it returns the database to a prior state.

Flashback Database requires you to create a flash recovery area for your database and enable the collection of flashback logs. See Chapter 16, "Performing Flashback and Database Point-in-Time Recovery" for more details about how the Flashback Database feature works, requirements for using Flashback Database, and how to enable collection of flashback logs required for Flashback Database. The requirements and preparations are the same whether you use RMAN or SQL*Plus.

To perform a flashback of the database with SQL*Plus:

1. Query the target database to determine the range of possible flashback SCNs. The following SQL*Plus queries show you the latest and earliest SCN in the flashback window:

```
SELECT CURRENT_SCN FROM V$DATABASE;

SELECT OLDEST_FLASHBACK_SCN, OLDEST_FLASHBACK_TIME
FROM   V$FLASHBACK_DATABASE_LOG;
```

2. Use other flashback features if necessary to identify the SCN or time of the unwanted changes to your database.

3. Start SQL*Plus with administrator privileges and run the FLASHBACK DATABASE statement to return the database to a prior TIMESTAMP or SCN. For example:

```
FLASHBACK DATABASE TO SCN 46963;
FLASHBACK DATABASE TO TIMESTAMP '2002-11-05 14:00:00';
FLASHBACK DATABASE
  TO TIMESTAMP to_timestamp('2002-11-11 16:00:00', 'YYYY-MM-DD HH24:MI:SS');
```

4. When the operation completes, open the database read-only and perform queries to make sure you have recovered the data you need.

   If your chosen target time was not far enough in the past, then use another FLASHBACK DATABASE statement. Otherwise, you can use RECOVER DATABASE to return the database back to the present time and then try another FLASHBACK DATABASE statement.

5. When satisfied with the results, open the database with the RESETLOGS option.

   If appropriate, you can also use Data Pump Export to save lost data, use RECOVER DATABASE to return the database to the present, and reimport the lost object.

> **See Also:** *Oracle Database Advanced Application Developer's Guide* to learn how to use related flashback features such as Flashback Query and Flashback Transaction Query

Overview of User-Managed Media Recovery

This section provides an overview of recovery with SQL*Plus. This section contains the following topics:

- About User-Managed Restore and Recovery
- Automatic Recovery with the RECOVER Command
- Recovery When Archived Logs Are in the Default Location
- Recovery When Archived Logs Are in a Nondefault Location
- Recovery Cancellation
- Parallel Media Recovery

About User-Managed Restore and Recovery

Typically, you restore a file when a media failure or user error has damaged or deleted one or more datafiles. In a user-managed restore operation, you use an operating system utility to restore a backup of the file.

If a media failure affects datafiles, then the recovery procedure depends on:

- The archiving mode of the database: ARCHIVELOG or NOARCHIVELOG
- The type of media failure
- The files affected by the media failure (datafiles, control files, archived redo logs, and the server parameter file are all candidates for restore operations)

If either a permanent or temporary media failure affects any datafiles of a database operating in NOARCHIVELOG mode, then the database automatically shuts down. If the media failure is temporary, then correct the underlying problem and restart the database. Usually, crash recovery will recover all committed transactions from the online redo log. If the media failure is permanent, then recover the database as described in "Recovering a Database in NOARCHIVELOG Mode" on page 28-16.

Table 28–1 explains the implications for media recovery when you lose files in database that runs in ARCHIVELOG mode.

Table 28–1 User-Managed Restore Operations

If you lose . . .	Then . . .
Datafiles in the SYSTEM tablespace or datafiles with active undo segments	The database automatically shuts down. If the hardware problem is temporary, then fix it and restart the database. Usually, crash recovery recovers lost transactions. If the hardware problem is permanent, then restore the datafiles from backups and recover the database as described in "Performing Closed Database Recovery" on page 28-8.
Datafiles not in the SYSTEM tablespace or datafiles that do not contain active rollback or undo segments	Affected datafiles are taken offline, but the database stays open. If the unaffected portions of the database must remain available, then do not shut down the database. Take tablespaces containing problem datafiles offline using the temporary option, then recover them as described in "Performing Open Database Recovery" on page 28-11.
All copies of the current control file	You must restore a backup control file and then open the database with the RESETLOGS option. If you do not have a backup, then you can attempt to re-create the control file. If possible, use the script included in the ALTER DATABASE BACKUP CONTROLFILE TO TRACE output. Additional work may be required to match the control file structure with the current database structure.
One copy of a multiplexed control file	Copy one of the intact multiplexed control files into the location of the damaged or missing control file and open the database. If you cannot copy the control file to its original location, then edit the initialization parameter file to reflect a new location or remove the damaged control file. Then, open the database.
One or more archived logs required for media recovery	You must restore backups of these archived logs for recovery to proceed. You can restore either to the default or nondefault location. If you do not have backups, then you must performing incomplete recovery up to an SCN before the first missing redo log and open RESETLOGS.
The server parameter file (SPFILE)	If you have a backup of the server parameter file, then restore it. Alternatively, if you have a backup of the client-side initialization parameter file, then you can restore a backup of this file, start the instance, and then re-create the server parameter file.

Note: Restore and recovery of Oracle-managed files is no different from restore and recovery of user-named files.

To perform media recovery, Oracle recommends that you use the RECOVER statement in SQL*Plus. You can also use the SQL statement ALTER DATABASE RECOVER, but the RECOVER statement is simpler in most cases. To start any type of media recovery, you must adhere to the following restrictions:

- You must have administrator privileges.
- All recovery sessions must be compatible.
- One session cannot start complete media recovery while another performs incomplete media recovery.
- You cannot start media recovery if you are connected to the database through a shared server process.

Automatic Recovery with the RECOVER Command

When using SQL*Plus to perform media recovery, the easiest strategy is to perform automatic recovery with the SQL*Plus RECOVER command. Automatic recovery

initiates recovery without manually prompting SQL*Plus to apply each individual archived redo log.

When using SQL*Plus, you have the following options for automating the application of the default filenames of archived redo logs needed during recovery:

- Issuing `SET AUTORECOVERY ON` before issuing the `RECOVER` command. If you recovery with `SET AUTORECOVERY OFF`, which is the default, then you must enter filenames manually or accept the suggested filename by pressing `Enter`.
- Specifying the `AUTOMATIC` keyword as an option of the `RECOVER` command.

In either case, no interaction is required when you issue the `RECOVER` command if the necessary files are in the correct locations with the correct names. When the database successfully applies a redo log file, the following message is returned:

```
Log applied.
```

You are then prompted for the next redo log in the sequence. If the most recently applied log is the last required log, then recovery is terminated.

The filenames used when you use automatic recovery are derived from the concatenated values of `LOG_ARCHIVE_FORMAT` with `LOG_ARCHIVE_DEST_`*n*, where *n* is the highest value among all enabled, local destinations. For example, assume the following initialization parameter settings are in effect in the database instance:

```
LOG_ARCHIVE_DEST_1 = "LOCATION=/arc_dest/loc1/"
LOG_ARCHIVE_DEST_2 = "LOCATION=/arc_dest/loc2/"
LOG_ARCHIVE_DEST_STATE_1 = DEFER
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
LOG_ARCHIVE_FORMAT = arch_%t_%s_%r.arc
```

In this example, SQL*Plus automatically suggests the filename `/arc_dest/loc2/arch_%t_%s_%r.arc` (where `%t` is the thread, `%s` is the sequence and `%r` is the resetlogs ID).

Automatic Recovery with SET AUTORECOVERY

After restoring datafile backups, you can run the `SET AUTORECOVERY ON` command to enable on automatic recovery. For example, you could enter the following commands in SQL*Plus to perform automatic recovery and open the database:

```
STARTUP MOUNT
SET AUTORECOVERY ON
RECOVER DATABASE
ALTER DATABASE OPEN;
```

> **Note:** After issuing the SQL*Plus `RECOVER` command, you can view all files that have been considered for recovery in the `V$RECOVERY_FILE_STATUS` view. You can access status information for each file in the `V$RECOVERY_STATUS` view. These views are not accessible after you terminate the recovery session.

Automatic Recovery with the AUTOMATIC Option of the RECOVER Command

Besides using `SET AUTORECOVERY` to turn on automatic recovery, you can also simply specify the `AUTOMATIC` keyword in the `RECOVER` command. For example, you could enter the following commands in SQL*Plus to perform automatic recovery and open the database:

```
STARTUP MOUNT
```

```
RECOVER AUTOMATIC DATABASE
ALTER DATABASE OPEN;
```

If you use an Oracle Real Application Clusters configuration, and if you are performing incomplete recovery or using a backup control file, then the database can only compute the name of the first archived redo log file from the *first* redo thread. You may have to manually apply the first log file from the other redo threads. After the first log file in a given thread has been supplied, the database can suggest the names of the subsequent logs in this thread.

Recovery When Archived Logs Are in the Default Location

Recovering when the archived logs are in their default location is the simplest case. As a log is needed, the database suggests the filename. If you are running nonautomatic media recovery with SQL*Plus, then the output is displayed in the format shown by this example:

```
ORA-00279: change 53577 generated at 11/26/02 19:20:58 needed for thread 1
ORA-00289: suggestion : /oracle/oradata/trgt/arch/arcr_1_802.arc
ORA-00280: change 53577 for thread 1 is in sequence #802
Specify log: [<RET> for suggested | AUTO | FROM logsource | CANCEL ]
```

Similar messages are returned when you use an `ALTER DATABASE ... RECOVER` statement. However, no prompt is displayed.

The database constructs suggested archived log filenames by concatenating the current values of the initialization parameters `LOG_ARCHIVE_DEST_`*n* (where *n* is the highest value among all enabled, local destinations) and `LOG_ARCHIVE_FORMAT` and using log history data from the control file. The following are possible settings:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION = /oracle/oradata/trgt/arch/'
LOG_ARCHIVE_FORMAT = arcr_%t_%s.arc

SELECT NAME FROM V$ARCHIVED_LOG;

NAME
----------------------------------------
/oracle/oradata/trgt/arch/arcr_1_467.arc
/oracle/oradata/trgt/arch/arcr_1_468.arc
/oracle/oradata/trgt/arch/arcr_1_469.arc
```

Thus, if all the required archived log files are mounted at the `LOG_ARCHIVE_DEST_1` destination, and if the value for `LOG_ARCHIVE_FORMAT` is never altered, then the database can suggest and apply log files to complete media recovery automatically.

Recovery When Archived Logs Are in a Nondefault Location

Performing media recovery when archived logs are not in their default location adds an extra step. You have the following mutually exclusive options:

- Edit the `LOG_ARCHIVE_DEST_`*n* parameter that specifies the location of the archived redo logs, then recover as usual.
- Use the `SET` statement in SQL*Plus to specify the nondefault log location before recovery, or the `FROM` parameter of the `RECOVER` command

Resetting the Archived Log Destination

You can edit the initialization parameter file or issue `ALTER SYSTEM` statements to change the default location of the archived redo logs.

To change the default archived log location before recovery:

1. Use an operating system utility to restore the archived logs to a nondefault location. For example, enter:

```
% cp /backup/arch/* /tmp/
```

2. Change the value for the archive log parameter to the nondefault location. You can issue `ALTER SYSTEM` statements while the instance is started, or edit the initialization parameter file and then start the database instance. For example, while the instance is shut down edit the parameter file as follows:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION=/tmp/'
LOG_ARCHIVE_FORMAT = arcr_%t_%s.arc
```

3. Using SQL*Plus, start a new instance by specifying the edited initialization parameter file, and then mount the database. For example, enter:

```
STARTUP MOUNT
```

4. Begin media recovery as usual. For example, enter:

```
RECOVER DATABASE
```

Overriding the Archived Log Destination

In some cases, you may want to override the current setting for the archiving destination parameter as a source for archived log files.

To recover archived logs in a nondefault location with SET LOGSOURCE:

1. Using an operating system utility, copy the archived redo logs to an alternative location. For example, enter:

```
% cp $ORACLE_HOME/oradata/trgt/arch/* /tmp
```

2. Specify the alternative location within SQL*Plus for the recovery operation. Use the `LOGSOURCE` parameter of the `SET` statement or the `RECOVER . . . FROM` clause of the `ALTER DATABASE` statement. For example, start SQL*Plus and run:

```
SET LOGSOURCE "/tmp"
```

3. Recover the offline tablespace. For example, to recover the offline tablespace `users` do the following:

```
RECOVER AUTOMATIC TABLESPACE users
```

4. Alternatively, you can avoid running `SET LOGSOURCE` and simply run:

```
RECOVER AUTOMATIC TABLESPACE users FROM "/tmp"
```

> **Note:** Overriding the redo log source does not affect the archive redo log destination for online redo log groups being archived.

Recovery Cancellation

If you start media recovery and must then interrupt it, then either enter `CANCEL` when prompted for a redo log file, or use your operating system's interrupt signal if you must terminate when recovering an individual datafile, or when automated recovery is in progress. After recovery is canceled, you can resume it later with the `RECOVER` command. Recovery resumes where it left off when it was canceled.

Parallel Media Recovery

By default, Oracle uses **parallel media recovery** to improve performance of the roll forward phase of media recovery. In parallel media recovery, the database uses a "division of labor" approach to allocate different processes to different data blocks while rolling forward, thereby making the procedure more efficient. The number of processes used is derived from the `CPU_COUNT` initialization parameter, which by default is equal to the number of CPUs on the system. For example, if parallel recovery is performed on a system where `CPU_COUNT` is 4, and only one datafile is recovered, then four spawned processes read blocks from the archive logs and apply redo.

Typically, media recovery is limited by data block reads and writes. Parallel recovery attempts to use all of the available I/O bandwidth of the system to improve performance. Unless there is a system I/O bottleneck or poor asynchronous I/O support, parallel recovery is likely to improve performance of recovery.

To override the default behavior of performing parallel recovery, use the `RECOVER` with the `NOPARALLEL` option, or `RECOVER PARALLEL 0`. Note that the `RECOVERY_PARALLELISM` initialization parameter controls instance or crash recovery *only*. Media recovery is not affected by the value used for `RECOVERY_PARALLELISM`.

> **See Also:** *SQL*Plus User's Guide and Reference* for more information about the SQL*Plus `RECOVER . . . PARALLEL` and `NOPARALLEL` statements

Performing Complete Database Recovery

Typically, you perform complete recovery of the database when a media failure has made one or more datafiles inaccessible. The `V$RECOVER_FILE` view indicates which files need recovery. When you perform complete database recovery, you use all available redo to recover the database to the current SCN.

Depending on the circumstances, you can either recover the whole database at once or recover individual tablespaces or datafiles. Because you do not have to open the database with the `RESETLOGS` option after complete recovery, you have the option of recovering some datafiles at one time and the remaining datafiles later.

The procedures in this section assume the following:

- The current control file is available. If you need to restore or re-create the control file, then see "Recovering After Loss of All Current Control Files" on page 29-2 and "Re-Creating a Control File" on page 29-6.
- You have backups of all needed datafiles. If you are missing datafile backups, then see "Re-Creating Datafiles When Backups Are Unavailable" on page 29-8.
- All necessary archived redo logs are available. If you are missing redo required to completely recover the database, then you must perform database point-in-time recovery, see "Performing Incomplete Database Recovery" on page 28-13.

This section describes the steps necessary to complete media recovery operations, and includes the following topics:

- Performing Closed Database Recovery
- Performing Open Database Recovery

Performing Closed Database Recovery

This section describes steps to perform complete recovery while the database is not open. You can recover either all damaged datafiles in one operation or perform individual recovery of each damaged datafile in separate operations.

To restore and recover damaged or missing datafiles:

1. If the database is open, query V$RECOVER_FILE to determine which datafiles need to be recovered and why they need to be recovered.

   If you are planning to perform complete recovery rather than point-in-time recovery, then you can recover only those datafiles which require recovery, rather than the whole database. Note that for point-in-time recovery, you must restore and recover all datafiles, unless you perform TSPITR as described in Chapter 20, "Performing RMAN Tablespace Point-in-Time Recovery (TSPITR)". You can also use Flashback Database, but this procedure affects all datafiles and returns the entire database to a past time.

   You can query V$RECOVER_FILE to list datafiles requiring recovery by datafile number with their status and error information.

   ```
   SELECT FILE#, ERROR, ONLINE_STATUS, CHANGE#, TIME
   FROM   V$RECOVER_FILE;
   ```

   Note: You cannot use V$RECOVER_FILE with a control file restored from backup or a control file that was re-created after the time of the media failure affecting the datafiles. A restored or re-created control file does not contain the information needed to update V$RECOVER_FILE accurately.

   You can also perform useful joins using the datafile number and the V$DATAFILE and V$TABLESPACE views, to get the datafile and tablespace names. Use the following SQL*Plus commands to format the output of the query:

   ```
   COL DF# FORMAT 999
   COL DF_NAME FORMAT A35
   COL TBSP_NAME FORMAT A7
   COL STATUS FORMAT A7
   COL ERROR FORMAT A10
   COL CHANGE# FORMAT 99999999
   SELECT r.FILE# AS df#, d.NAME AS df_name, t.NAME AS tbsp_name,
          d.STATUS, r.ERROR, r.CHANGE#, r.TIME
   FROM   V$RECOVER_FILE r, V$DATAFILE d, V$TABLESPACE t
   WHERE  t.TS# = d.TS#
   AND    d.FILE# = r.FILE#;
   ```

   The ERROR column identifies the problem for each file requiring recovery.

2. Query V$ARCHIVED_LOG and V$RECOVERY_LOG to determine which archived redo log files are needed.

   V$ARCHIVED_LOG lists filenames for all archived redo logs, whereas V$RECOVERY_LOG lists only the archived redo logs that the database needs to perform media recovery. The latter view also includes the probable names of the files using LOG_ARCHIVE_FORMAT.

> **Note:** V$RECOVERY_LOG is only populated when media recovery is required for a datafile. Thus, this view is not useful in the case of a planned recovery, such as recovery from a user error.
>
> If a datafile requires recovery, but no backup of the datafile exists, then you need all redo generated starting from the time when the datafile was added to the database.

3. If all archived logs are available in the default location, then skip to the next step.

   If some archived logs need to be restored, and if sufficient space is available, then restore the required archived redo log files to the location specified by LOG_ARCHIVE_DEST_1. The database locates the correct log automatically when required during media recovery. For example, you might enter a command such as the following on Linux or UNIX:

   ```
   % cp /disk2/arch/* $ORACLE_HOME/oradata/trgt/arch
   ```

   If sufficient space is not available, then restore some or all of the required archived redo log files to an alternative location.

4. If the database is open, then shut it down. For example:

   ```
   SHUTDOWN IMMEDIATE
   ```

5. Inspect the media to determine the source of the problem.

   If the hardware problem that caused the media failure was temporary, and if the data was undamaged (for example, a disk or controller power failure), then no media recovery is required: start the database and resume normal operations.

   If you cannot repair the problem, then proceed to the next step.

6. If the files are permanently damaged, then identify the most recent backups for the damaged files. Restore *only* the datafiles damaged by the media failure: do not restore undamaged datafiles or any online redo log files.

   For example, if *ORACLE_HOME*/oradata/trgt/users01.dbf is the only damaged file, then you may find that /backup/users01_10_24_02.dbf is the most recent backup of this file. If you do not have a backup of a specific datafile, then you may be able to create an empty replacement file that can be recovered.

7. Use an operating system utility to restore the datafiles to their default location or to a new location. For example, a Linux or UNIX user restoring users01.dbf to its default location might enter:

   ```
   % cp /backup/users01_10_24_06.dbf $ORACLE_HOME/oradata/trgt/users01.dbf
   ```

   Use the following guidelines when determining where to restore datafile backups:

   - If hardware problem is repaired and you can restore the datafiles to their default locations, then restore the datafiles to their default locations and begin media recovery.
   - If the hardware problem persists and you cannot restore datafiles to their original locations, then restore the datafiles to an alternative storage device. Indicate the new location of these files in the control file with ALTER DATABASE RENAME FILE. See *Oracle Database Administrator's Guide*, as necessary.

- If you are restoring a datafile to a raw disk or partition, then the technique is basically the same as when restoring to a file on a file system. Be aware of the naming conventions for files on raw devices (which differ depending on the operating system), however, and use an operating system utility that supports raw devices.

8. Connect to the database with administrator privileges, then start a new instance and mount, but do not open, the database. For example, enter:

```
STARTUP MOUNT
```

9. If you restored one or more damaged datafiles to alternative locations, then update the control file of the database to reflect the new datafile names. For example, to change the filename of the datafile in tablespace `users` you might enter:

```
ALTER DATABASE RENAME FILE '?/oradata/trgt/users01.dbf' TO
                           '/disk2/users01.dbf';
```

10. Obtain the datafile names and statuses of all datafiles by checking the list of datafiles that normally accompanies the current control file or querying the `V$DATAFILE` view. For example, enter:

```
SELECT NAME,STATUS FROM V$DATAFILE;
```

11. Ensure that all datafiles requiring recovery are online. The only exceptions are datafiles in an offline tablespace that was taken offline normally or datafiles in a read-only tablespace. For example, to guarantee that a datafile named `/oracle/dbs/tbs_10.f` is online, enter the following:

```
ALTER DATABASE DATAFILE '/oracle/dbs/tbs_10.f' ONLINE;
```

    If a specified datafile is already online, then the database ignores the statement. If you prefer, create a script to bring all datafiles online at once as in the following:

```
SPOOL onlineall.sql
SELECT 'ALTER DATABASE DATAFILE '''||name||''' ONLINE;' FROM V$DATAFILE;
SPOOL OFF

SQL> @onlineall
```

12. If you restored archived redo logs to an alternative location, then you can specify the location before media recovery with the `LOGSOURCE` parameter of the `SET` command in SQL*Plus. For example, if the logs are staged in `/tmp`, you can enter the following command:

```
SET LOGSOURCE /tmp
```

    Alternatively, use the `FROM` parameter on the `RECOVER` command in the following step. For example, if the logs are staged in `/tmp`, you can enter the following command:

```
RECOVER AUTOMATIC FROM '/tmp' DATABASE
```

    Note: Overriding the redo log source does not affect the archive redo log destination for online redo log groups being archived.

13. Issue a statement to recover the database, tablespace, or datafile. For example, enter one of the following `RECOVER` commands:

```
RECOVER AUTOMATIC DATABASE   # whole database
```

```
RECOVER AUTOMATIC TABLESPACE users   # specific tablespace
RECOVER AUTOMATIC DATAFILE '?/oradata/trgt/users01.dbf'; # specific datafile
```

If you choose not to automate the application of archived redo logs, then you must accept or reject each prompted log. If you automate recovery, then the database applies the logs automatically. Recovery continues until all required archived and online redo logs have been applied to the restored datafiles. The database notifies you when media recovery is complete:

```
Media recovery complete.
```

Note that if no archived redo logs are required for complete media recovery, then the database applies all necessary online redo log files and terminates recovery.

14. After recovery terminates, open the database for use:

```
ALTER DATABASE OPEN;
```

> **See Also:** "Overview of User-Managed Media Recovery" on page 28-2 for more information about applying redo log files and *Oracle Database Reference* for descriptions of `V$` views

15. After archived logs are applied, and after making sure that a copy of each archived log group still exists in offline storage, delete the restored copy of the archived redo log file to free disk space. For example:

```
% rm /tmp/*.arc
```

> **See Also:** *Oracle Database Reference* for more information about the data dictionary views, and "Overview of User-Managed Media Recovery" on page 28-2 for an overview of log application during media recovery

Performing Open Database Recovery

It is possible for a media failure to occur while the database remains open, leaving the undamaged datafiles online and available for use. Damaged datafiles—but not the tablespaces that contain them—are automatically taken offline if the database writer is unable to write to them. Queries that cannot read damaged files return errors, but the datafiles are not taken offline because of the failed queries. For example, you may run a SQL query and see output such as:

```
ERROR at line 1:
ORA-01116: error in opening database file 3
ORA-01110: data file 11: '/oracle/oradata/trgt/cwmlite02.dbf'
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
```

You cannot use the procedure in this section to perform complete media recovery on the `SYSTEM` tablespace while the database is open. If the media failure damages datafiles of the `SYSTEM` tablespace, then the database automatically shuts down.

To restore datafiles in an open database:

1. Follow step 1 through step 3 in "Performing Closed Database Recovery" on page 28-8.
2. If the database is open, then take all tablespaces containing damaged datafiles offline. For example, if tablespace `users` and `tools` contain damaged datafiles, then execute the following SQL statements:

```
ALTER TABLESPACE users OFFLINE TEMPORARY;
ALTER TABLESPACE tools OFFLINE TEMPORARY;
```

Note that if you specify TEMPORARY, then Oracle Database performs a **checkpoint** for all online datafiles in the tablespace. Files that are offline when you issue this statement may require media recovery before you bring the tablespace back online. If you specify IMMEDIATE, then you must perform media recovery on the tablespace before bringing it back online.

3. Inspect the media to determine the source of the problem.

   As explained in "Running the DBVERIFY Utility" on page 27-17, you can use the DBVERIFY utility to run an integrity check on offline datafiles.

   If the hardware problem that caused the media failure was temporary, and if the data was undamaged, then no media recovery is required. You can bring the offline tablespaces online and resume normal operations. If you cannot repair the problem, or if DBVERIFY reports corrupt blocks, then proceed to the next step.

4. If files are permanently damaged, then use operating system commands to restore the most recent backup files of *only* the datafiles damaged by the media failure. For example, to restore users01.dbf you might use the cp command on Linux or UNIX as follows:

   ```
   % cp /disk2/backup/users01.dbf $ORACLE_HOME/oradata/trgt/users01.dbf
   ```

   If the hardware problem is fixed and the datafiles can be restored to their original locations, then do so. Otherwise, restore the datafiles to an alternative storage device.Do not restore undamaged datafiles, online redo logs, or control files.

   > **Note:** In some circumstances, if you do not have a backup of a specific datafile, you can use ALTER DATABASE CREATE DATAFILE to create an empty replacement file that is recoverable.

5. If you restored one or more damaged datafiles to alternative locations, then update the control file of the database to reflect the new datafile names. For example, to change the filename of the datafile in tablespace users you might enter:

   ```
   ALTER DATABASE RENAME FILE '?/oradata/trgt/users01.dbf' TO
                              '/disk2/users01.dbf';
   ```

6. If you restored archived redo logs to an alternative location, then you can specify the location before media recovery with the LOGSOURCE parameter of the SET command in SQL*Plus. For example, if the logs are staged in /tmp, you can enter the following command:

   ```
   SET LOGSOURCE /tmp
   ```

   Alternatively, use the FROM parameter on the RECOVER command in the following step. For example, if the logs are staged in /tmp, you can enter the following command:

   ```
   RECOVER AUTOMATIC FROM '/tmp' TABLESPACE users, tools;
   ```

   > **Note:** Overriding the redo log source does not affect the archive redo log destination for online redo log groups being archived.

7. Connect to the database with administrator privileges, and start offline tablespace recovery of all damaged datafiles in one or more offline tablespaces using one step. For example, recover `users` and `tools` as follows:

```
RECOVER AUTOMATIC TABLESPACE users, tools;
```

   The database begins the roll forward phase of media recovery by applying the necessary archived and online redo logs to reconstruct the restored datafiles. Unless the applying of files is automated with `RECOVER AUTOMATIC` or `SET AUTORECOVERY ON`, the database prompts for each required redo log file.

   Recovery continues until all required archived logs have been applied to the datafiles. The online redo logs are then automatically applied to the restored datafiles to complete media recovery. If no archived redo logs are required for complete media recovery, then the database does not prompt for any. Instead, all necessary online redo logs are applied, and media recovery is complete.

8. When the damaged tablespaces are recovered up to the moment that media failure occurred, bring the offline tablespaces online. For example, to bring tablespaces `users` and `tools` online, issue the following statements:

```
ALTER TABLESPACE users ONLINE;
ALTER TABLESPACE tools ONLINE;
```

> **See Also:** *Oracle Database Administrator's Guide* to learn about creating datafiles and *Oracle Database SQL Language Reference* to learn about `ALTER DATABASE RENAME FILE`

Performing Incomplete Database Recovery

Typically, you perform database point-in-time recovery (DBPITR) in the following situations:

- You want to recover the database to an SCN before a user or administrative error.
- The database contains corrupt blocks.
- Complete database recovery failed because all necessary archived redo logs were not available.
- You are creating a test or reporting database from production database backups.

If the database is operating in `ARCHIVELOG` mode, and if the only copy of an archived redo log file is damaged, then the damaged file does not affect the present operation of the database. Table 28–2 describes situations that can arise depending on when the redo log was written and when you backed up the datafile.

Table 28–2 Loss of Archived Redo Logs

If you backed up . . .	Then . . .
All datafiles after the filled online redo log group (which is now archived) was written	The archived version of the filled online redo log group is not required for complete media recovery.
A specific datafile before the filled online redo log group was written	If the corresponding datafile is damaged by a permanent media failure, then use the most recent backup of the damaged datafile and perform tablespace point-in-time recovery of the damaged datafile, up to the damaged archived redo log file.

> **Caution:** If you know that an archived redo log group has been damaged, then immediately back up all datafiles so that you will have a whole database backup that does not require the damaged archived redo log.

The technique for DBPITR is very similar to the technique described in "Performing Closed Database Recovery" on page 28-8, except that you terminate DBPITR by specifying a particular time or SCN or entering `CANCEL`. Cancel-based recovery prompts you with the suggested filenames of archived redo logs. Recovery stops when you specify `CANCEL` instead of a filename or when all redo has been applied to the datafiles. Cancel-based recovery is the best technique if you want to control which archived log terminates recovery.

The procedures in this section assume the following:

- The current control file is available. If you need to restore or re-create the control file, then see "Recovering After Loss of All Current Control Files" on page 29-2.
- You have backups of all needed datafiles. If you are missing datafile backups, then see "Re-Creating Datafiles When Backups Are Unavailable" on page 29-8.

This section contains the following topics:

- Performing Cancel-Based Incomplete Recovery
- Performing Time-Based or Change-Based Incomplete Recovery

Performing Cancel-Based Incomplete Recovery

In cancel-based recovery, recovery proceeds by prompting you with the suggested filenames of archived redo log files. Recovery stops when you specify `CANCEL` instead of a filename or when all redo has been applied to the datafiles.

To perform cancel-based recovery:

1. Follow steps step 1 through step 8 in "Performing Closed Database Recovery" on page 28-8.
2. Begin cancel-based recovery by issuing the following command in SQL*Plus:

   ```
   RECOVER DATABASE UNTIL CANCEL
   ```

   > **Note:** If you fail to specify the `UNTIL` clause on the `RECOVER` command, then the database assumes a complete recovery and will not open until all redo is applied.

   The database applies the necessary redo log files to reconstruct the restored datafiles. The database supplies the name it expects to find from `LOG_ARCHIVE_DEST_1` and requests you to stop or proceed with applying the log file. Note that if the control file is a backup, then you must supply the names of the online redo logs if you want to apply the changes in these logs.
3. Continue applying redo log files until the last log has been applied to the restored datafiles, then cancel recovery by executing the following command:

   ```
   CANCEL
   ```

   The database indicates whether recovery is successful. If you cancel before all the datafiles have been recovered to a consistent SCN and then try to open the

database, then you will get an ORA-1113 error if more recovery is necessary. You can query V$RECOVER_FILE to determine whether more recovery is needed, or if a backup of a datafile was not restored prior to starting incomplete recovery.

4. Open the database with the RESETLOGS option. You must always reset the logs after incomplete recovery or recovery with a backup control file. For example:

```
ALTER DATABASE OPEN RESETLOGS;
```

If you attempt to OPEN RESETLOGS when you should not, or if you neglect to reset the log when you should, then the database returns an error and does not open the database. Correct the problem and try again.

> **See Also:** "About User-Managed Media Recovery Problems" on page 28-18 for descriptions of situations that can cause ALTER DATABASE OPEN RESETLOGS to fail

5. After opening the database with the RESETLOGS option, check the alert log.

> **Note:** The easiest way to locate trace files and the alert log is to run the following SQL query: SELECT NAME, VALUE FROM V$DIAG_INFO.

When you open with the RESETLOGS option, the database returns different messages depending on whether recovery was complete or incomplete. If the recovery was complete, then the following message appears in alert log:

```
RESETLOGS after complete recovery through change scn
```

If the recovery was incomplete, then this message is reported in the alert log, where *scn* refers to the end point of incomplete recovery:

```
RESETLOGS after incomplete recovery UNTIL CHANGE scn
```

Also check the alert log to determine whether the database detected inconsistencies between the data dictionary and the control file. Table 28–3 describes two possible scenarios.

Table 28–3 Inconsistencies Between Data Dictionary and Control File

Datafile Listed in Control File	Datafile Listed in Data Dictionary	Result
Yes	No	References to the unlisted datafile are removed from the control file. A message in the alert log indicates what was found.
No	Yes	The database creates a placeholder entry in the control file under MISSING*nnnnn* (where *nnnnn* is the file number in decimal). MISSING*nnnnn* is flagged in the control file as offline and requiring media recovery. You can make the datafile corresponding to MISSING*nnnnn* accessible by using ALTER DATABASE RENAME FILE for MISSING*nnnnn* so that it points to the datafile. If you do not have a backup of this datafile, then drop the tablespace.

Performing Time-Based or Change-Based Incomplete Recovery

This section describes how to specify an SCN or time for the end point of recovery. If your database is affected by seasonal time changes (for example, daylight savings time), then you may experience a problem if a time appears twice in the redo log and you want to recover to the second, or later time. To handle time changes, perform cancel-based or change-based recovery.

To perform change-based or time-based recovery:

1. Follows steps step 1 through step 8 in "Performing Closed Database Recovery" on page 28-8.
2. Issue the `RECOVER DATABASE UNTIL` statement to begin recovery. If recovering to an SCN, then specify as a decimal number without quotation marks. For example, to recover through SCN 10034 issue:

   ```
   RECOVER DATABASE UNTIL CHANGE 10034;
   ```

   If recovering to a time, then the time is always specified using the following format, delimited by single quotation marks: `'YYYY-MM-DD:HH24:MI:SS'`. The following statement recovers the database up to a specified time:

   ```
   RECOVER DATABASE UNTIL TIME '2000-12-31:12:47:30'
   ```

3. Apply the necessary redo log files to recover the restored datafiles. The database automatically terminates the recovery when it reaches the correct time, and returns a message indicating whether recovery is successful.

   Note: Unless recovery is automated, the database supplies the name from `LOG_ARCHIVE_DEST_1` and asks you to stop or proceed with after each log. If the control file is a backup, then after the archived logs are applied you must supply the names of the online logs.

4. Follow steps 4 and 5 in "Performing Cancel-Based Incomplete Recovery" on page 28-14.

Recovering a Database in NOARCHIVELOG Mode

If a media failure damages datafiles in a `NOARCHIVELOG` database, then the only option for recovery is usually to restore a consistent whole database backup. If you are using logical backups created by Oracle Data Pump Export to supplement regular physical backups, then you can also attempt to restore the database by importing an exported backup of the database into a re-created database or a database restored from an old backup.

To restore and recover the most recent whole database backup:

1. If the database is open, then shut down the database. For example, enter:

   ```
   SHUTDOWN IMMEDIATE
   ```

2. If possible, correct the media problem so that the backup database files can be restored to their original locations.
3. Restore the most recent whole database backup with operating system commands. Restore all of the datafiles and control files of the whole database backup, not just the damaged files. If the hardware problem has not been corrected and some or all

of the database files must be restored to alternative locations, then restore the whole database backup to a new location. The following example restores a whole database backup to its default location:

```
% cp /backup/*.dbf $ORACLE_HOME/oradata/trgt/
```

4. If necessary, edit the restored initialization parameter file to indicate the new location of the control files. For example:

```
CONTROL_FILES = "/new_disk/oradata/trgt/control01.dbf"
```

5. Start an instance using the restored and edited parameter file and mount, but do not open, the database. For example:

```
STARTUP MOUNT
```

6. If the restored datafile filenames will be different (as will be the case when you restore to a different file system or directory, on the same node or a different node), then update the control file to reflect the new datafile locations. For example, to rename datafile `1` you might enter:

```
ALTER DATABASE RENAME FILE '?/oradata/trgt/system01.dbf' TO
                           '/new_disk/oradata/system01.dbf';
```

7. If the online redo logs were located on a damaged disk, and the hardware problem is not corrected, then specify a new location for each affected online log. For example, enter:

```
ALTER DATABASE RENAME FILE '?/oradata/trgt/redo01.log' TO
                           '/new_disk/oradata/redo_01.log';
ALTER DATABASE RENAME FILE '?/oradata/trgt/redo02.log' TO
                           '/new_disk/oradata/redo_02.log';
```

8. Because online redo logs are never backed up, you cannot restore them with the datafiles and control files. In order to allow the database to reset the online redo logs, you must first mimic incomplete recovery:

```
RECOVER DATABASE UNTIL CANCEL
CANCEL
```

9. Open the database in `RESETLOGS` mode. This command clears the online redo logs and resets the log sequence to 1:

```
ALTER DATABASE OPEN RESETLOGS;
```

Note that restoring a `NOARCHIVELOG` database backup and then resetting the log discards all changes to the database made from the time the backup was taken to the time of the failure.

> **See Also:** *Oracle Database Administrator's Guide* for more information about renaming and relocating datafiles, and *Oracle Database SQL Language Reference* to learn about `ALTER DATABASE RENAME FILE`

Troubleshooting Media Recovery

This section describes how to troubleshoot user-managed media recovery, that is, media recovery performed without using Recovery Manager (RMAN). This section includes the following topics:

- About User-Managed Media Recovery Problems

- Investigating the Media Recovery Problem: Phase 1
- Trying to Fix the Recovery Problem Without Corrupting Blocks: Phase 2
- Deciding Whether to Allow Recovery to Corrupt Blocks: Phase 3
- Allowing Recovery to Corrupt Blocks: Phase 4
- Performing Trial Recovery

About User-Managed Media Recovery Problems

Table 28–4, " Media Recovery Problems" describes potential problems that can occur during media recovery.

Table 28–4 Media Recovery Problems

Problem	Description
Missing or misnamed archived log	Recovery stops because the database cannot find the archived log recorded in the control file.
When you attempt to open the database, error `ORA-1113` indicates that a datafile needs media recovery	This error commonly occurs because: ■ You are performing incomplete recovery but failed to restore all needed datafile backups. ■ Incomplete recovery stopped before datafiles reached a consistent SCN. ■ You are recovering datafiles from an online backup, but not enough redo was applied to make the datafiles consistent. ■ You are performing recovery with a backup control file, and did not specify the location of a needed online redo log. ■ A datafile is undergoing media recovery when you attempt to open the database. ■ Datafiles needing recovery were not brought online before executing `RECOVER DATABASE`, and so were not recovered.
Redo record problems	Two possible cases are as follows: ■ Recovery stops because of failed consistency checks, a problem called **stuck recovery**. Stuck recovery can occur when an underlying operating system or storage system loses a write issued by the database during normal operation. ■ The database signals an internal error when applying the redo. This problem can be caused by an Oracle Database bug. If **checksum** verification is not being used, then the errors can also be caused by corruptions to the redo or data blocks.
Corrupted archived logs	Logs may be corrupted while they are stored on or copied between storage systems. If `DB_BLOCK_CHECKSUM` is enabled, then the database usually signals a checksum error. If checksum checking is disabled, then log corruption may appear as a problem with redo.
Archived logs with incompatible parallel redo format	If you enable the parallel redo feature, then the database generates redo logs in a new format. Prior releases of Oracle are unable to apply parallel redo logs. However, releases prior to Oracle9*i* Release 2 (9.2) can detect the parallel redo format and indicate the inconsistency with the following error message: `External error 00303, 00000, "cannot process Parallel Redo".`
Corrupted data blocks	A datafile backup may have contained a corrupted data block, or the data block may become corrupted either during recovery or when it was copied to the backup. If `DB_BLOCK_CHECKSUM` is enabled, then the database computes a checksum for each block during normal operations and stores it in the block before writing it to disk. When the database reads the block from disk later, it recomputes the checksum and compares it to the stored value. If they do not match, then the database signals a checksum errors. If checksum checking is disabled, then the problem may also appear as a redo corruption.
Random problems	Memory corruptions and other transient problems can occur during recovery.

The symptoms of media recovery problems are usually external or internal errors signaled during recovery. For example, an external error indicates that a redo block or a data block has failed checksum verification checks. Internal errors can be caused by either bugs in the database or errors arising from the underlying operating system and hardware.

If media recovery encounters a problem while recovering a database backup, then whether or not it is a stuck recovery problem or a problem during redo application, the database always stops and leaves the datafiles undergoing recovery in a consistent state, that is, at a consistent SCN preceding the failure. You can then do one of the following:

- Open the database read-only to investigate the problem.
- Open the database with the `RESETLOGS` option, as long as the requirements for opening `RESETLOGS` have been met. Note that the `RESETLOGS` restrictions apply to opening the **physical standby database** as well, because a standby database is updated by a form of media recovery.

In general, opening the database read-only or opening with the `RESETLOGS` option require all online datafiles to be recovered to the same SCN. If this requirement is not met, then the database may signal `ORA-1113` or other errors when you attempt to open. Some common causes of `ORA-1113` are described in Table 28–4, " Media Recovery Problems".

The basic methodology for responding to media recovery problems occurs in the following phases:

1. Try to identify the cause of the problem. Run a trial recovery if needed.
2. If the problem is related to missing redo logs or you suspect there is a redo log, memory, or data block corruption, then try to resolve it using the methods described in Table 28–5.
3. If you cannot resolve the problem using the methods described in Table 28–5, then do one of the following:
   - Open the database with the `RESETLOGS` option if you are recovering a whole database backup. If you have performed serial media recovery, then the database contains all the changes up to but not including the changes at the SCN where the corruption occurred. No changes from this SCN onward are in the recovered part of the database. If you have restored online backups, then opening `RESETLOGS` succeeds only if you have recovered through all the `ALTER . . . END BACKUP` operations in the redo stream.
   - Proceed with recovery by allowing media recovery to corrupt data blocks. After media recovery completes, try performing **block media recovery** using RMAN.
   - Call Oracle Support Services as a last resort.

   See Also: "Performing Disaster Recovery" on page 19-8 to learn about block media recovery

Investigating the Media Recovery Problem: Phase 1

If media recovery encounters a problem, then obtain as much information as possible after recovery halts. You do not want to waste time fixing the wrong problem, which may in fact make matters worse.

The goal of this initial investigation is to determine whether the problem is caused by incorrect setup, corrupted redo logs, corrupted data blocks, memory corruption, or

other problems. If you see a **checksum** error on a data block, then the data block is corrupted. If you see a checksum error on a redo log block, then the redo log is corrupted.

Sometimes the cause of a recovery problem can be difficult to determine. Nevertheless, the methods in this chapter allow you to quickly recover a database even when you do not completely understand the cause of the problem.

To investigate media recovery problems:

1. Examine the `alert.log` to see whether the error messages give general information about the nature of the problem. For example, does the `alert_SID.log` indicate any checksum failures? Does the `alert_SID.log` indicate that media recovery may have to corrupt data blocks in order to continue?
2. Check the trace file generated by the Oracle process during recovery. It may contain additional error information.

Trying to Fix the Recovery Problem Without Corrupting Blocks: Phase 2

Depending on the type of media recovery problem you suspect, you have different solutions at your disposal. You can try one or a combination of the techniques described in Table 28–5. Note that these techniques are fairly safe: in almost all cases, they should not cause damage to the database.

Table 28–5 Media Recovery Solutions

If you suspect . . .	Then . . .
Missing or misnamed archived redo logs	Determine whether you entered the correct filename. If you did, then check to see whether the log is missing from the operating system. If it is missing, and if you have a backup, then restore the backup and apply the log. If you do not have a backup, then if possible perform incomplete recovery up to the point of the missing log.
`ORA-1113` for `ALTER DATABASE OPEN`	Review the causes of this error in Table 28–4, " Media Recovery Problems". Make sure that all read/write datafiles requiring recovery are online. If you use a backup control file for recovery, then the control file and datafiles must be at a consistent SCN for the database to be opened. If you do not have the necessary redo, then you must re-create the control file.
Corrupt archived logs	The log is corrupted if the checksum verification on the log redo block fails. If `DB_BLOCK_CHECKSUM` is not enabled either during the recovery session or when the database generated the redo, then recovery problems may be caused by corrupted logs. If the log is corrupt and an alternate copy of the corrupt log is available, then try to apply it and see whether this tactic fixes the problem. The `DB_BLOCK_CHECKSUM` initialization parameter determines whether checksums are computed for redo log and data blocks.

Table 28–5 (Cont.) Media Recovery Solutions

If you suspect . . .	Then . . .
Archived logs with incompatible parallel redo format	If you are running an Oracle release prior to Oracle9*i* Release 2, and if you are attempting to apply redo logs created with the parallel redo format, then you must do the following steps: **1.** Upgrade the database to a later release. **2.** Perform media recovery. **3.** Shut down the database consistently and back up the database. **4.** Downgrade the database to the original release.
Memory corruption or transient problems	You may be able to fix the problem by shutting down the database and restarting recovery. The database should be left in a consistent state if the second attempt also fails.
Corrupt data blocks	Restore and recover the datafile again with user-managed methods, or restore and recover individual data blocks with the RMAN `RECOVER ... BLOCK` command. This technique may fix the problem. A data block is corrupted if the checksum verification on the block fails. If `DB_BLOCK_CHECKING` is disabled, then a corrupted data block problem may appear as a redo problem. If you must proceed with media recovery, then you may want to corrupt the block now and continue recovery, and then use RMAN to perform block media recovery later.

If you cannot fix the problem with the methods described in Table 28–5, then there may be no easy way to fix the problem without losing data. You have these options:

- Open the database with the `RESETLOGS` option (for whole database recovery).

  This solution discards all changes after the point where the redo problem occurred, but guarantees a logically consistent database.

- Allow media recovery to corrupt one or more data blocks and then proceed.

  This option only succeeds if the alert log indicates that recovery can continue if it is allowed to corrupt a data block, which should be the case for most recovery problems. This option is best if you need to bring up the database quickly and recover all changes. If you are considering this option, then proceed to "Deciding Whether to Allow Recovery to Corrupt Blocks: Phase 3" on page 28-21.

> **See Also:** "Performing Disaster Recovery" on page 19-8 to learn how to perform block media recovery with the `RECOVER ... BLOCK` command

Deciding Whether to Allow Recovery to Corrupt Blocks: Phase 3

When media recovery encounters a problem, the alert log may indicate that recovery can continue if it is allowed to corrupt the data block causing the problem. The alert log contains information about the block: its block type, block address, the tablespace it belongs to, and so forth. For blocks containing user data, the alert log may also report the data object number.

In this case, the database can proceed with recovery if it is allowed to mark the problem block as corrupt. Nevertheless, this response is not always advisable. For example, if the block is an important block in the `SYSTEM` tablespace, marking the block as corrupt can eventually prevent you from opening the recovered database. Another consideration is whether the recovery problem is isolated. If this problem is followed immediately by many other problems in the redo stream, then you may want to open the database with the `RESETLOGS` option.

For a block containing user data, you can usually query the database to find out which object or table owns this block. If the database is not open, then you should be able to open the database read-only, even if you are recovering a whole database backup. The following example cancels recovery and opens read-only:

```
CANCEL
ALTER DATABASE OPEN READ ONLY;
```

Assume that the data object number reported in the `alert_SID.log` is `8031`. You can determine the owner, object name, and object type by issuing this query:

```
SELECT OWNER, OBJECT_NAME, SUBOBJECT_NAME, OBJECT_TYPE
FROM DBA_OBJECTS
WHERE DATA_OBJECT_ID = 8031;
```

To determine whether a recovery problem is isolated, you can run a diagnostic **trial recovery**, which scans the redo stream for problems but does not actually make any changes to the recovered database. If a trial recovery discovers any recovery problems, then it reports them in the `alert_SID.log`. You can use the `RECOVER ... TEST` statement to invoke trial recovery, as described in "Executing the RECOVER ... TEST Statement" on page 28-24.

After you have done these investigations, you can follow the guidelines in Table 28–6 to decide whether to allow recovery to corrupt blocks.

Table 28–6 Guidelines for Allowing Recovery to Permit Corruption

If the problem is . . .	and the block is . . .	Then . . .
not isolated	n/a	You should probably open the database with the `RESETLOGS` option. This response is important for stuck recovery problems, because stuck recovery can be caused by the operating system or a storage system losing writes. If an operating system or storage system suddenly fails, then it can cause stuck recovery problems on several blocks.
isolated	in the `SYSTEM` tablespace	Do not corrupt the block, because it may eventually prevent you from opening the database. However, sometimes data in the `SYSTEM` tablespace is unimportant. If you must corrupt a `SYSTEM` block and recover all changes, then contact Oracle Support Services.
isolated	index data	Consider corrupting index blocks because the index can be rebuilt later after the database has been recovered.
isolated	user data	Decide based on the importance of the data. If you continue with datafile recovery and corrupt a block, then you lose data in the block. However, you can use RMAN to perform block media recovery later after datafile recovery completes. If you open `RESETLOGS`, then the database is consistent but loses any changes made after the point where recovery was stopped.
isolated	rollback or undo data	If all of the transactions are committed, then consider corrupting the rollback or undo block. The database is not harmed if the transactions that generated the undo are never rolled back. However, if those transactions are rolled back, then corrupting the undo block can cause problems. If you are unsure, then contact Oracle Support Services.

See Also: "Performing Trial Recovery" on page 28-23 to learn how to perform trial recovery, and "Allowing Recovery to Corrupt Blocks: Phase 4" on page 28-23 if you decide to corrupt blocks

Allowing Recovery to Corrupt Blocks: Phase 4

If you decide to allow recovery to proceed in spite of block corruptions, then run the `RECOVER` command with the `ALLOW` *n* `CORRUPTION` clause, where *n* is the number of allowable corrupt blocks.

To allow recovery to corrupt blocks:

1. Ensure that all normal recovery preconditions are met. For example, if the database is open, then take tablespaces offline before attempting recovery.
2. Run the `RECOVER` command. The following statements shows a valid example:

```
RECOVER DATABASE ALLOW 5 CORRUPTION
```

Performing Trial Recovery

When problems such as stuck recovery occur, you have a difficult choice. If the block is relatively unimportant, and if the problem is isolated, then it is better to corrupt the block. But if the problem is not isolated, then it may be better to open the database with the `RESETLOGS` option.

Because of this situation, the Oracle Database supports **trial recovery**. A trial recovery applies redo in a way similar to normal media recovery, but it never writes its changes to disk and it always rolls back its changes. Trial recovery occurs only in memory.

> **See Also:** "Allowing Recovery to Corrupt Blocks: Phase 4" on page 28-23

How Trial Recovery Works

By default, if a trial recovery encounters a stuck recovery or similar problem, then it always marks the data block as corrupt in memory when this action can allow recovery to proceed. The database writes errors generated during trial recovery to alert files. These errors are clearly marked as test run errors.

Like normal media recovery, trial recovery can prompt you for archived log filenames and ask you to apply them. Trial recovery ends when:

- The database runs out of the maximum number of buffers in memory that trial recovery is permitted to use
- An unrecoverable error is signaled, that is, an error that cannot be resolved by corrupting a data block
- You cancel or interrupt the recovery session
- The next redo record in the redo stream changes the control file
- All requested redo has been applied

When trial recovery ends, the database removes all effects of the test run from the system—except the possible error messages in the alert files. If the instance fails during trial recovery, then the database removes all effects of trial recovery from the system because trial recovery never writes changes to disk.

Trial recovery lets you foresee what problems might occur if you were to continue with normal recovery. For problems caused by ongoing memory corruption, trial recovery and normal recovery can encounter different errors.

Executing the RECOVER ... TEST Statement

You can use the TEST option for any RECOVER command. For example, you can start SQL*Plus and then issue any of the following commands:

```
RECOVER DATABASE TEST
RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL TEST
RECOVER TABLESPACE users TEST
RECOVER DATABASE UNTIL CANCEL TEST
```

By default, trial recovery always attempts to corrupt blocks in memory if this action allows trial recovery to proceed. In other words, trial recovery by default can corrupt an unlimited number of data blocks. You can specify the ALLOW *n* CORRUPTION clause on the RECOVER . . . TEST statement to limit the number of data blocks trial recovery can corrupt in memory.

A trial recovery command is usable in any scenario in which a normal recovery command is usable. Nevertheless, you should only need to run trial recovery when recovery runs into problems.

29

Performing User-Managed Recovery: Advanced Scenarios

This chapter describes several common media failure scenarios. It shows how to recover from each failure when using a user-managed backup and recovery strategy, that is, a strategy that does not depend on Recovery Manager. This chapter includes the following topics:

- Responding to the Loss of a Subset of the Current Control Files
- Recovering After Loss of All Current Control Files
- Re-Creating a Control File
- Re-Creating Datafiles When Backups Are Unavailable
- Recovering NOLOGGING Tables and Indexes
- Recovering Transportable Tablespaces
- Recovering After the Loss of Online Redo Log Files
- Recovering from a Dropped Table Without Using Flashback Features
- Dropping a Database with SQL*Plus

Responding to the Loss of a Subset of the Current Control Files

Use the following procedures to recover a database if a permanent media failure has damaged one or more control files of a database and at least one current control file has *not* been damaged by the media failure.

Copying a Multiplexed Control File to a Default Location

If the disk and file system containing the lost control file are intact, then you can simply copy one of the intact control files to the location of the missing control file. In this case, you do not have to edit the `CONTROL_FILES` initialization parameter.

To replace a damaged control file by copying a multiplexed control file:

1. If the instance is still running, then shut it down:

   ```
   SQL> SHUTDOWN ABORT
   ```

2. Correct the hardware problem that caused the media failure. If you cannot repair the hardware problem quickly, then proceed with database recovery by restoring damaged control files to an alternative storage device, as described in "Copying a Multiplexed Control File to a Nondefault Location" on page 29-2.

3. Use an intact multiplexed copy of the database's current control file to copy over the damaged control files. For example, to replace `bad_cf.f` with `good_cf.f`, you might enter:

   ```
   % cp /oracle/good_cf.f /oracle/dbs/bad_cf.f
   ```

4. Start a new instance and mount and open the database. For example, enter:

   ```
   SQL> STARTUP
   ```

Copying a Multiplexed Control File to a Nondefault Location

Assuming that the disk and file system containing the lost control file are not intact, then you cannot copy one of the good control files to the location of the missing control file. In this case, you must alter the `CONTROL_FILES` initialization parameter to indicate a new location for the missing control file.

To restore a control file to a nondefault location:

1. If the instance is still running, then shut it down:

   ```
   SQL> SHUTDOWN ABORT
   ```

2. If you cannot correct the hardware problem that caused the media failure, then copy the intact control file to alternative locations.

   For example, to copy a good version of `control01.dbf` to a new disk location you might issue:

   ```
   % cp /disk1/oradata/trgt/control01.dbf /new_disk/control01.dbf
   ```

3. Edit the parameter file of the database so that the `CONTROL_FILES` parameter reflects the current locations of all control files and excludes all control files that were not restored.

   Assume the initialization parameter file contains the following setting:

   ```
   CONTROL_FILES='/disk1/oradata/trgt/control01.dbf','/bad_disk/control02.dbf'
   ```

   You can edit the `CONTROL_FILES` initialization parameter as follows:

   ```
   CONTROL_FILES='/disk1/oradata/trgt/control01.dbf','/new_disk/control02.dbf'
   ```

4. Start a new instance and mount and open the database. For example:

   ```
   SQL> STARTUP
   ```

Recovering After Loss of All Current Control Files

Use the following procedures to restore a backup control file if a permanent media failure has damaged all control files of a database and you have a backup of the control file. When a control file is inaccessible, you can start the instance, but not mount the database. If you attempt to mount the database when the control file is unavailable, then you receive the following error message:

```
ORA-00205: error in identifying control file, check alert log for more info
```

Note: The easiest way to locate trace files and the alert log is to run the following SQL query: `SELECT NAME, VALUE FROM V$DIAG_INFO`.

You cannot mount and open the database until the control file is accessible again. If you restore a backup control file, then you must open RESETLOGS.

As indicated in Table 29–1, the procedure for restoring the control file depends on whether the online redo logs are available.

Table 29–1 Scenarios When Control Files Are Lost

Status of Online Logs	Status of Datafiles	Restore Procedure
Available	Current	If the online logs contain redo necessary for recovery, then restore a backup control file and apply the logs during recovery. You must specify the filename of the online logs containing the changes in order to open the database. After recovery, open RESETLOGS. **Note:** If you re-create a control file, then it is not necessary to OPEN RESETLOGS after recovery when the online redo logs are accessible.
Unavailable	Current	If the online logs contain redo necessary for recovery, then re-create the control file. Because the online redo logs are inaccessible, open RESETLOGS.
Available	Backup	Restore a backup control file, perform complete recovery, and then open RESETLOGS.
Unavailable	Backup	Restore a backup control file, perform incomplete recovery, and then open RESETLOGS.

Recovering with a Backup Control File in the Default Location

If possible, restore the control file to its original location. In this way, you avoid having to specify new control file locations in the initialization parameter file.

To restore a backup control file to its default location:

1. If the instance is still running, shut it down:

```
SQL> SHUTDOWN ABORT
```

2. Correct the hardware problem that caused the media failure.

3. Restore the backup control file to all locations specified in the CONTROL_FILES parameter. For example, if /disk1/oradata/trgt/control01.dbf and /disk2/oradata/trgt/control02.dbf are the control file locations listed in the server parameter file, then use an operating system utility to restore the backup control file to these locations:

```
% cp /backup/control01.dbf /disk1/oradata/trgt/control01.dbf
% cp /backup/control02.dbf /disk2/oradata/trgt/control02.dbf
```

4. Start a new instance and mount the database. For example, enter:

```
SQL> STARTUP MOUNT
```

5. Begin recovery by executing the RECOVER command with the USING BACKUP CONTROLFILE clause. Specify UNTIL CANCEL if you are performing incomplete recovery. For example, enter:

```
SQL> RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL
```

6. Apply the prompted archived logs. If you then receive another message saying that the required archived log is missing, then it probably means that a necessary

redo record is located in the online redo logs. This situation can occur when unarchived changes were located in the online logs when the instance crashed.

For example, assume that you see the following:

```
ORA-00279: change 55636 generated at 11/08/2002 16:59:47 needed for thread 1
ORA-00289: suggestion : /oracle/work/arc_dest/arcr_1_111.arc
ORA-00280: change 55636 for thread 1 is in sequence #111
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

You can specify the name of an online redo log and press Enter (you may have to try this a few times until you find the correct log):

```
ORACLE_HOME/oradata/redo01.dbf
Log applied.
Media recovery complete.
```

If the online logs are inaccessible, then you can cancel recovery without applying them. If all datafiles are current, and if redo in the online logs is required for recovery, then you cannot open the database without applying the online logs. If the online logs are inaccessible, then you must re-create the control file, using the procedure described in "Re-Creating a Control File" on page 29-6.

7. Open the database with the RESETLOGS option after finishing recovery:

```
SQL> ALTER DATABASE OPEN RESETLOGS;
```

Recovering with a Backup Control File in a Nondefault Location

If you cannot restore the control file to its original place because the media damage is too severe, then you must specify new control file locations in the server parameter file. A valid control file must be available in all locations specified by the CONTROL_FILES initialization parameter. If not, then the database prevents you from the mounting the database.

To restore a control file to a nondefault location:

- Follow the steps in "Recovering with a Backup Control File in the Default Location" on page 29-3, except after step 2 add the following step:

  Edit all locations specified in the CONTROL_FILES initialization parameter to reflect the new control file locations. Assume that the control file locations listed in the server parameter file are as follows, and both disks are inaccessible:

  ```
  CONTROL_FILES='/disk1/oradata/trgt/control01.dbf',
                '/disk2/oradata/trgt/control02.dbf'
  ```

  You can edit the initialization parameter file and specify accessible locations, as shown in the following example:

  ```
  CONTROL_FILES='/disk3/cf/control01.dbf','/disk4/cf/control02.dbf'
  ```

Recovering Through an Added Datafile with a Backup Control File

If database recovery with a backup control file rolls forward through a CREATE TABLESPACE or an ALTER TABLESPACE ADD DATAFILE operation, then the database stops recovery when applying the redo record for the added files and lets you confirm the filenames.

For example, suppose the following sequence of events occurs:

1. You back up the database

2. You create a new tablespace containing the following datafiles: `/disk1/oradata/trgt/test01.dbf` and `/disk1/oradata/trgt/test02.dbf`.
3. You restore a backup control file and perform media recovery through the `CREATE TABLESPACE` operation.

You may see the following error when applying the `CREATE TABLESPACE` redo data:

```
ORA-00283: recovery session canceled due to errors
ORA-01244: unnamed datafile(s) added to control file by media recovery
ORA-01110: data file 11: '/disk1/oradata/trgt/test02.dbf'
ORA-01110: data file 10: '/disk1/oradata/trgt/test01.dbf'
```

To recover through an `ADD DATAFILE` operation:

1. View the files added by querying `V$DATAFILE`. For example:

```
SELECT FILE#,NAME
FROM V$DATAFILE;

FILE#           NAME
--------------- ----------------------
1               /disk1/oradata/trgt/system01.dbf
.
.
.
10               /disk1/oradata/trgt/UNNAMED00001
11               /disk1/oradata/trgt/UNNAMED00002
```

2. If multiple unnamed files exist, then determine which unnamed file corresponds to which datafile by using one of these methods:
   - Open the `alert_`*`SID`*`.log`, which contains messages about the original file location for each unnamed file.
   - Derive the original file location of each unnamed file from the error message and `V$DATAFILE`: each unnamed file corresponds to the file in the error message with the same file number.
3. Issue the `ALTER DATABASE RENAME FILE` statement to rename the datafiles. For example, enter:

```
ALTER DATABASE RENAME FILE '/db/UNNAMED00001' TO
                           '/disk1/oradata/trgt/test01.dbf';
ALTER DATABASE RENAME FILE '/db/UNNAMED00002' TO
                           '/disk1/oradata/trgt/test02.dbf';
```

4. Continue recovery by issuing the previous recovery statement. For example:

```
RECOVER AUTOMATIC DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL
```

Recovering Read-Only Tablespaces with a Backup Control File

If you have a read-only tablespace on read-only or slow media, then you may encounter errors or poor performance when recovering with the `USING BACKUP CONTROLFILE` option. This situation occurs when the backup control file indicates that a tablespace was read/write when the control file was backed up. In this case, media recovery may attempt to write to the files. For read-only media, the database issues an error saying that it cannot write to the files. For slow media, such as a hierarchical storage system backed up by tapes, performance may suffer.

To avoid these problems, use current control files rather than backups to recover the database. If you need to use a backup control file, then you can also avoid this problem if the read-only tablespace has not suffered a media failure. You have the following alternatives for recovering read-only and slow media when using a backup control file:

- Take datafiles from read-only tablespaces offline before doing recovery with a backup control file, and then bring the files online at the end of media recovery.
- Use the correct version of the control file for the recovery. If the tablespace will be read-only when recovery completes, then the control file backup must be from a time when the tablespace was read-only. Similarly, if the tablespace will be read/write at the end of recovery, then the control file must be from a time when the tablespace was read/write.

Re-Creating a Control File

If all control files have been lost in a permanent media failure, but all online redo log members remain intact, then you can recover the database after creating a new control file. Note that you are *not* required to open the database with the `RESETLOGS` option after the recovery.

Depending on the existence and currency of a control file backup, you have the options listed in the following table for generating the text of the `CREATE CONTROLFILE` statement. Note that changes to the database are recorded in the `alert_`*SID*`.log`, so check this log when deciding which option to choose.

Table 29–2 Options for Creating the Control File

If you . . .	Then . . .
Executed `ALTER DATABASE BACKUP CONTROLFILE TO TRACE NORESETLOGS` after you made the last structural change to the database, and if you have saved the SQL command trace output	Use the `CREATE CONTROLFILE` statement from the trace output as-is.
Performed your most recent execution of `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` before you made a structural change to the database	Edit the output of `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` to reflect the change. For example, if you recently added a datafile to the database, then add this datafile to the `DATAFILE` clause of the `CREATE CONTROLFILE` statement.
Backed up the control file with the `ALTER DATABASE BACKUP CONTROLFILE TO` *`filename`* statement (not the `TO TRACE` option)	Use the control file copy to obtain SQL output. Create a temporary database instance, mount the backup control file, and then run `ALTER DATABASE BACKUP CONTROLFILE TO TRACE NORESETLOGS`. If the control file copy predated a recent structural change, then edit the trace to reflect the change.
Do not have a control file backup in either `TO TRACE` format or `TO` *`filename`* format	Execute the `CREATE CONTROLFILE` statement manually (refer to *Oracle Database SQL Language Reference*).

Note: If your character set is not the default US7ASCII, then you must specify the character set as an argument to the `CREATE CONTROLFILE` statement. The database character set is written to the alert log at startup. The character set information is also recorded in the `BACKUP CONTROLFILE TO TRACE` output.

To create a new control file and recover the database:

1. Start the database in NOMOUNT mode. For example, enter:

```
STARTUP NOMOUNT
```

2. Create the control file with the CREATE CONTROLFILE statement, specifying the NORESETLOGS option (refer to Table 29–2 for options). The following example assumes that the character set is the default US7ASCII:

```
CREATE CONTROLFILE REUSE DATABASE SALES NORESETLOGS ARCHIVELOG
     MAXLOGFILES 32
     MAXLOGMEMBERS 2
     MAXDATAFILES 32
     MAXINSTANCES 16
     MAXLOGHISTORY 1600
LOGFILE
     GROUP 1 (
       '/diska/prod/sales/db/log1t1.dbf',
       '/diskb/prod/sales/db/log1t2.dbf'
     )  SIZE 100K
     GROUP 2 (
       '/diska/prod/sales/db/log2t1.dbf',
       '/diskb/prod/sales/db/log2t2.dbf'
     )  SIZE 100K,
DATAFILE
     '/diska/prod/sales/db/database1.dbf',
     '/diskb/prod/sales/db/filea.dbf';
```

   After creating the control file, the instance mounts the database.

3. Recover the database as normal (*without* specifying the USING BACKUP CONTROLFILE clause):

```
RECOVER DATABASE
```

4. Open the database after recovery completes (RESETLOGS option not required):

```
ALTER DATABASE OPEN;
```

5. Immediately back up the control file. The following SQL statement backs up a database's control file to /backup/control01.dbf:

```
ALTER DATABASE BACKUP CONTROLFILE TO '/backup/control01.dbf' REUSE;
```

> **See Also:** "Backing Up the Control File to a Trace File" on page 27-11, and "Re-Creating Datafiles When Backups Are Unavailable" on page 29-8

Recovering Through a RESETLOGS with a Created Control File

You can recover backups through an OPEN RESETLOGS operation so long as:

- You have a current, backup, or created control file that knows about the prior incarnations
- You have all available archived redo logs

If you need to re-create the control file, then the trace file generated by ALTER DATABASE BACKUP CONTROLFILE TO TRACE will contain the necessary commands to re-construct the complete incarnation history. The V$DATABASE_INCARNATION view displays the RESETLOGS history known to the control file, while the V$LOG_HISTORY view displays the archived log history.

It is possible for the incarnation history to be incomplete in the in re-created control file. For example, archived logs necessary for recovery may be missing. In this case, it is possible to create incarnation records explicitly with the ALTER DATABASE REGISTER LOGFILE statement.

In the following example, you register four logs that are necessary for recovery but are not recorded in the re-created control file, and then recover the database:

```
ALTER DATABASE REGISTER LOGFILE '/disk1/oradata/trgt/arch/arcr_1_1_42343523.arc';
ALTER DATABASE REGISTER LOGFILE '/disk1/oradata/trgt/arch/arcr_1_1_34546466.arc';
ALTER DATABASE REGISTER LOGFILE '/disk1/oradata/trgt/arch/arcr_1_1_23435466.arc';
ALTER DATABASE REGISTER LOGFILE '/disk1/oradata/trgt/arch/arcr_1_1_12343533.arc';
RECOVER AUTOMATIC DATABASE;
```

Recovery of Read-Only Files with a Re-Created Control File

If a current or backup control file is unavailable for recovery, then you can execute a CREATE CONTROLFILE statement. Read-only files should not be listed in the CREATE CONTROLFILE statement so that recovery can skip these files. No recovery is required for read-only datafiles unless you restored backups of these files from a time when the datafiles were read/write.

After you create a new control file and attempt to mount and open the database, the database performs a data dictionary check against the files listed in the control file. For each file that is not listed in the CREATE CONTROLFILE statement but is present in the data dictionary, an entry is created for them in the control file. These files are named as MISSING*nnnnn*, where *nnnnn* is a five digit number starting with 0.

After the database is open, rename the read-only files to their correct filenames by executing the ALTER DATABASE RENAME FILE statement for all the files whose name is prefixed with MISSING.

To prepare for a scenario in which you might have to re-create the control file:

- Run the following statement when the database is mounted or open to obtain the CREATE CONTROLFILE syntax:

  ```
  ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
  ```

  The preceding SQL statement produces a trace file that you can edit and use as a script to re-create the control file. You can specify either the RESETLOGS or NORESETLOGS (default) keywords to generate CREATE CONTROLFILE . . . RESETLOGS or CREATE CONTROLFILE . . . NORESETLOGS versions of the script.

All the restrictions related to read-only files in CREATE CONTROLFILE statements also apply to offline normal tablespaces, except that you need to bring the tablespace online after the database is open. You should leave out tempfiles from the CREATE CONTROLFILE statement and add them after opening the database.

> **See Also:** "Backing Up the Control File to a Trace File" on page 27-11 to learn how to make trace backups of the control file

Re-Creating Datafiles When Backups Are Unavailable

If a datafile is damaged and no backup of the file is available, then you can still recover the datafile if:

- All archived log files written after the creation of the original datafile are available

- The control file contains the name of the damaged file (that is, the control file is current, or is a backup taken after the damaged datafile was added to the database)

> **Note:** You cannot re-create any of the datafiles for the SYSTEM tablespace by using the CREATE DATAFILE clause of the ALTER DATABASE statement because the necessary redo is not available.

To re-create a datafile for recovery:

1. Create a new, empty datafile to replace a damaged datafile that has no corresponding backup. For example, assume that the datafile /disk1/oradata/trgt/users01.dbf has been damaged, and no backup is available. The following statement re-creates the original datafile (same size) on disk2:

```
ALTER DATABASE CREATE DATAFILE '/disk1/oradata/trgt/users01.dbf' AS
                               '/disk2/users01.dbf';
```

   This statement creates an empty file that is the same size as the lost file. The database looks at information in the control file and the data dictionary to obtain size information. The old datafile is renamed as the new datafile.

2. Perform media recovery on the empty datafile. For example, enter:

```
RECOVER DATAFILE '/disk2/users01.dbf'
```

3. All archived logs written after the original datafile was created must be applied to the new, empty version of the lost datafile during recovery.

Recovering NOLOGGING Tables and Indexes

You can create tables and indexes with the CREATE TABLE AS SELECT statement. You can also specify that the database create them with the NOLOGGING option. When you create a table or index as NOLOGGING, the database does not generate redo log records for the operation. Thus, you cannot recover objects created with NOLOGGING, even if you are running in ARCHIVELOG mode.

> **Note:** If you cannot afford to lose tables or indexes created with NOLOGGING, then make a backup after the unrecoverable table or index is created.

Be aware that when you perform media recovery, and some tables or indexes are created normally whereas others are created with the NOLOGGING option, the NOLOGGING objects are marked logically corrupt by the RECOVER operation. Any attempt to access the unrecoverable objects returns an ORA-01578 error message. Drop the NOLOGGING objects and re-create them if needed.

Because it is possible to create a table with the NOLOGGING option and then create an index with the LOGGING option on that table, the index is not marked as logically corrupt after you perform media recovery. The table was unrecoverable (and thus marked as corrupt after recovery), however, so the index points to corrupt blocks. The index must be dropped, and the table and index must be re-created if necessary.

> **See Also:** *Oracle Data Guard Concepts and Administration* for information about the effect of NOLOGGING on a database

Recovering Transportable Tablespaces

The **transportable tablespace** feature of Oracle Database enables a user to transport a set of tablespaces from one database to another. Transporting a tablespace into a database is like creating a tablespace with preloaded data. Using this feature is often an advantage for the following reasons:

- It is faster than using the Data Pump Export or SQL*Loader utilities because it involves only copying datafiles and integrating metadata
- You can use it to move index data, hence avoiding the necessity of rebuilding indexes

> **See Also:** *Oracle Database Administrator's Guide* for detailed information about using the transportable tablespace feature

Like normal tablespaces, transportable tablespaces are recoverable. However, while you can recover normal tablespaces without a backup, you must have a consistent version of the transported datafiles in order to recover a transported tablespace.

To recover a transportable tablespace, use the following procedure:

1. If the database is open, then take the transported tablespace offline. For example, if you want to recover the `users` tablespace, then issue:

   ```
   ALTER TABLESPACE users OFFLINE IMMEDIATE;
   ```

2. Restore a backup of the transported datafiles with an operating system utility. The backup can be the initial version of the transported datafiles or any backup taken after the tablespace is transported. For example, enter:

   ```
   % cp /backup/users.dbf $ORACLE_HOME/oradata/trgt/users01.dbf
   ```

3. Recover the tablespace as normal. For example, enter:

   ```
   RECOVER TABLESPACE users
   ```

You may see the error `ORA-01244` when recovering through a transportable tablespace operation just as when recovering through a `CREATE TABLESPACE` operation. In this case, rename the unnamed files to the correct locations using the procedure in "Recovering Through an Added Datafile with a Backup Control File" on page 29-4.

Recovering After the Loss of Online Redo Log Files

If a media failure has affected the online redo logs of a database, then the appropriate recovery procedure depends on the following considerations:

- The configuration of the online redo log: mirrored or non-mirrored
- The type of media failure: temporary or permanent
- The types of online redo log files affected by the media failure: current, active, unarchived, or inactive

Table 29–3 displays `V$LOG` status information that can be crucial in a recovery situation involving online redo logs.

Table 29–3 STATUS Column of V$LOG

Status	Description
UNUSED	The online redo log has never been written to.
CURRENT	The online redo log is active, that is, needed for instance recovery, and it is the log to which the database is currently writing. The redo log can be open or closed.
ACTIVE	The online redo log is active, that is, needed for instance recovery, but is not the log to which the database is currently writing.It may be in use for block recovery, and may or may not be archived.
CLEARING	The log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE statement. After the log is cleared, then the status changes to UNUSED.
CLEARING_CURRENT	The current log is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch such as an I/O error writing the new log header.
INACTIVE	The log is no longer needed for instance recovery. It may be in use for media recovery, and may or may not be archived.

Recovering After Losing a Member of a Multiplexed Online Redo Log Group

If the online redo log of a database is multiplexed, and if at least one member of each online redo log group is not affected by the media failure, then the database continues functioning as normal, but error messages are written to the log writer trace file and the alert_*SID*.log of the database.

Solve the problem by taking one of the following actions:

- If the hardware problem is temporary, then correct it. The log writer process accesses the previously unavailable online redo log files as if the problem never existed.
- If the hardware problem is permanent, then drop the damaged member and add a new member by using the following procedure.

> **Note:** The newly added member provides no redundancy until the log group is reused.

1. Locate the filename of the damaged member in V$LOGFILE. The status is INVALID if the file is inaccessible:

```
SELECT GROUP#, STATUS, MEMBER
FROM V$LOGFILE
WHERE STATUS='INVALID';

GROUP#    STATUS       MEMBER
-------   -----------  ---------------------
0002      INVALID      /disk1/oradata/trgt/redo02.log
```

2. Drop the damaged member. For example, to drop member redo02.log from group 2, issue:

```
ALTER DATABASE DROP LOGFILE MEMBER '/disk1/oradata/trgt/redo02.log';
```

3. Add a new member to the group. For example, to add redo02.log to group 2, issue:

```
ALTER DATABASE ADD LOGFILE MEMBER '/disk1/oradata/trgt/redo02b.log'
  TO GROUP 2;
```

If the file you want to add already exists, then it must be the same size as the other group members, and you must specify `REUSE`. For example:

```
ALTER DATABASE ADD LOGFILE MEMBER '/disk1/oradata/trgt/redo02b.log'
  REUSE TO GROUP 2;
```

Recovering After the Loss of All Members of an Online Redo Log Group

If a media failure damages all members of an online redo log group, then different scenarios can occur depending on the type of online redo log group affected by the failure and the archiving mode of the database.

If the damaged online redo log group is current and active, then it is needed for crash recovery; otherwise, it is not.

Table 29–4 Recovering After the Loss of an Online Redo Log Group

If the group is . . .	Then . . .	And you should . . .
Inactive	It is not needed for crash recovery	Clear the archived or unarchived group.
Active	It is needed for crash recovery	Attempt to issue a checkpoint and clear the log; if impossible, then you must either use Flashback Database or restore a backup and perform incomplete recovery up to the most recent available redo log.
Current	It is the redo log that the database is currently writing to	Attempt to clear the log; if impossible, then you must either use Flashback Database or restore a backup and perform incomplete recovery up to the most recent available redo log.

Your first task is to determine whether the damaged group is active or inactive.

1. Locate the filename of the lost redo log in `V$LOGFILE` and then look for the group number corresponding to it. For example, enter:

```
SELECT GROUP#, STATUS, MEMBER FROM V$LOGFILE;

GROUP#    STATUS       MEMBER
-------   -----------  ---------------------
0001                    /oracle/dbs/log1a.f
0001                    /oracle/dbs/log1b.f
0002      INVALID       /oracle/dbs/log2a.f
0002      INVALID       /oracle/dbs/log2b.f
0003                    /oracle/dbs/log3a.f
0003                    /oracle/dbs/log3b.f
```

2. Determine which groups are active.

   For example, execute the following SQL query (sample output included):

```
SELECT GROUP#, MEMBERS, STATUS, ARCHIVED
FROM V$LOG;

GROUP#  MEMBERS           STATUS     ARCHIVED
------  -------           ---------  -----------
 0001   2                 INACTIVE   YES
```

```
0002    2                  ACTIVE     NO
0003    2                  CURRENT    NO
```

3. Perform one of the following actions:
   - If the affected group is inactive, then follow the procedure in Losing an Inactive Online Redo Log Group on page 29-13.
   - If the affected group is active (as in the preceding example), then follow the procedure in "Losing an Active Online Redo Log Group" on page 29-14.

Losing an Inactive Online Redo Log Group

If all members of an online redo log group with `INACTIVE` status are damaged, then the procedure depends on whether you can fix the media problem that damaged the inactive redo log group. If the failure is temporary, then fix the problem. The log writer can reuse the redo log group when required. If the failure is permanent, then the damaged inactive online redo log group eventually halts normal database operation. Reinitialize the damaged group manually by issuing the `ALTER DATABASE CLEAR LOGFILE` statement as described in this section.

Clearing Inactive, Archived Redo You can clear an inactive redo log group when the database is open or closed. The procedure depends on whether the damaged group has been archived.

To clear an inactive, online redo log group that has been archived:

1. If the database is shut down, then start a new instance and mount the database:

   ```
   STARTUP MOUNT
   ```

2. Reinitialize the damaged log group. For example, to clear redo log group `2`, issue the following statement:

   ```
   ALTER DATABASE CLEAR LOGFILE GROUP 2;
   ```

Clearing Inactive, Unarchived Redo Clearing a not-yet-archived redo log allows it to be reused without archiving it. This action makes backups unusable if they were started before the last change in the log, unless the file was taken offline prior to the first change in the log. Hence, if you need the cleared log file for recovery of a backup, then you cannot recover that backup. Also, it prevents complete recovery from backups due to the missing log.

To clear an inactive, online redo log group that has not been archived:

1. If the database is shut down, then start a new instance and mount the database:

   ```
   SQL> STARTUP MOUNT
   ```

2. Clear the log using the `UNARCHIVED` keyword.

   For example, to clear log group `2`, issue the following SQL statement:

   ```
   SQL> ALTER DATABASE CLEAR LOGFILE UNARCHIVED GROUP 2;
   ```

   If there is an offline datafile that requires the cleared log to bring it online, then the keywords `UNRECOVERABLE DATAFILE` are required. The datafile and its entire tablespace have to be dropped because the redo necessary to bring it online is being cleared, and there is no copy of it. For example, enter:

   ```
   SQL> ALTER DATABASE CLEAR LOGFILE UNARCHIVED GROUP 2 UNRECOVERABLE DATAFILE;
   ```

3. Immediately back up all datafiles in the database with an operating system utility, so that you have a backup you can use for complete recovery without relying on the cleared log group. For example, enter:

```
% cp /disk1/oracle/dbs/*.dbf /disk2/backup
```

4. Back up the database's control file with the ALTER DATABASE statement. For example, enter:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/dbs/cf_backup.f';
```

Failure of CLEAR LOGFILE Operation The ALTER DATABASE CLEAR LOGFILE statement can fail with an I/O error due to media failure when it is not possible to:

- Relocate the redo log file onto alternative media by re-creating it under the currently configured redo log filename
- Reuse the currently configured log filename to re-create the redo log file because the name itself is invalid or unusable (for example, due to media failure)

In these cases, the ALTER DATABASE CLEAR LOGFILE statement (before receiving the I/O error) would have successfully informed the control file that the log was being cleared and did not require archiving. The I/O error occurred at the step in which the CLEAR LOGFILE statement attempts to create the new redo log file and write zeros to it. This fact is reflected in V$LOG.CLEARING_CURRENT.

Losing an Active Online Redo Log Group

If the database is still running and the lost active redo log is *not* the current log, then issue the ALTER SYSTEM CHECKPOINT statement. If successful, then the active redo log becomes inactive, and you can follow the procedure in "Losing an Inactive Online Redo Log Group" on page 29-13. If unsuccessful, or if your database has halted, then perform one of procedures in this section, depending on the archiving mode.

The current log is the one LGWR is currently writing to. If a LGWR I/O fails, then LGWR terminates and the instance crashes. In this case, you must restore a backup, perform incomplete recovery, and open the database with the RESETLOGS option.

Recovering from Loss of Active Logs in NOARCHIVELOG Mode In this scenario, the database archiving mode is NOARCHIVELOG.

To recover from loss of an active online log group in NOARCHIVELOG mode:

1. If the media failure is temporary, then correct the problem so that the database can reuse the group when required.
2. Restore the database from a consistent, whole database backup (datafiles and control files). For example, enter:

```
% cp /disk2/backup/*.dbf $ORACLE_HOME/oradata/trgt/
```

3. Mount the database:

```
STARTUP MOUNT
```

4. Because online redo logs are not backed up, you cannot restore them with the datafiles and control files. In order to allow the database to reset the online redo logs, you must first mimic incomplete recovery:

```
RECOVER DATABASE UNTIL CANCEL
CANCEL
```

5. Open the database using the RESETLOGS option:

```
ALTER DATABASE OPEN RESETLOGS;
```

6. Shut down the database consistently. For example, enter:

```
SHUTDOWN IMMEDIATE
```

7. Make a whole database backup.

If the media failure is temporary, then correct the problem so that the database can reuse the group when required. If not temporary, then use the following procedure.

Recovering from Loss of Active Logs in ARCHIVELOG Mode In this scenario, the database archiving mode is ARCHIVELOG.

To recover from loss of an active online redo log group in ARCHIVELOG mode:

1. Begin incomplete media recovery, recovering up through the log before the damaged log.
2. Ensure that the current name of the lost redo log can be used for a newly created file. If not, then rename the members of the damaged online redo log group to a new location. For example, enter:

```
ALTER DATABASE RENAME FILE "/disk1/oradata/trgt/redo01.log" TO
"/tmp/redo01.log";
ALTER DATABASE RENAME FILE "/disk1/oradata/trgt/redo01.log" TO
"/tmp/redo02.log";
```

3. Open the database using the RESETLOGS option:

```
ALTER DATABASE OPEN RESETLOGS;
```

> **Note:** All updates executed from the endpoint of the incomplete recovery to the present must be re-executed.

Loss of Multiple Redo Log Groups

If you have lost multiple groups of the online redo log, then use the recovery method for the most difficult log to recover. The order of difficulty, from most difficult to least difficult, is as follows:

1. The current online redo log
2. An active online redo log
3. An unarchived online redo log
4. An inactive online redo log

Recovering from a Dropped Table Without Using Flashback Features

One common error is the accidental dropping of a table from your database. In general, the fastest and simplest solution is to use the flashback drop feature to reverse the dropping of the table. If you cannot use flashback table (for example, because flashback drop is disabled or the table was dropped with the PURGE option), then you can perform the procedure in this section.

In this scenario, assume that you do not have the flashback database functionality enabled, so FLASHBACK DATABASE is not an option. However, you do have physical

backups of the database. If possible, keep the database that experienced the user error online and available for use.

> **Note:** Grant powerful privileges only to appropriate users to minimize user errors that require recovery.

To recover a table that has been accidentally dropped:

1. Back up all datafiles of the existing database in case an error is made during the remaining steps of this procedure.
2. Restore a partial backup of the database to an alternative location. At minimum, restore the following:
   - `SYSTEM` and `SYSAUX` tablespaces
   - Tablespaces containing undo or rollback segments
   - Self-contained tablespaces that contain the data to be retrieved
3. Perform incomplete recovery of this backup using a restored backup control file, to the point just before the table was dropped.
4. Export the lost data from the temporary, restored version of the database using Data Pump Export. In this case, export the accidentally dropped table.

   > **Note:** System audit options are exported.

5. Use the Data Pump Import utility to import the data back into the production database.
6. Delete the files of the temporary copy of the database to conserve space.

> **See Also:** *Oracle Database Utilities* for more information about the Oracle Data Pump

Dropping a Database with SQL*Plus

You may need to remove a database, that is, the database files that form the database, from the operating system. For example, this scenario can occur when you create a test database and then no longer have a use for it. The SQL statement `DROP DATABASE` can perform this function.

> **See Also:** "Dropping a Database" on page 11-24 to learn how to use the equivalent RMAN command `DROP DATABASE`

1. After connecting to the database with administrator privileges, ensure that the database is either mounted or open in restricted mode with no users connected.

   For example, enter the following command:

   ```
   SQL> STARTUP RESTRICT FORCE MOUNT
   ```

2. Remove the datafiles and control files from the operating system.

   For example, enter the following command:

   ```
   SQL> DROP DATABASE; # deletes all database files, both ASM and non-ASM
   ```

If the database is on raw disk, then the command does not delete the actual raw disk special files.

3. Use an operating system utility to delete all backups and archived logs associated with the database.

   For example, enter the following command:

   ```
   % rm /backup/* /disk1/oradata/trgt/arch/*
   ```

Glossary

active database duplication

A **duplicate database** that is created over a network without restoring backups of the **target database**. This technique is an alternative to **backup-based duplication**.

ancestor incarnation

The **parent incarnation** is the database **incarnation** from which the **current incarnation** branched following an `OPEN RESETLOGS` operation. The parent of the parent incarnation is an ancestor incarnation. Any parent of an ancestor incarnation is also an ancestor incarnation.

archival backup

A database backup that is exempted from the normal backup and recovery strategy. Typically, these backups are archived onto separate storage media and retained for long periods.

archived redo log

A copy of one of the filled members of an online redo log group made when the database is in `ARCHIVELOG` mode. After the LGWR process fills each online redo log with redo records, the archiver process copies the log to one or more redo log archiving destinations. This copy is the archived redo log. Note that RMAN does not distinguish between an original archived redo log and an image copy of an archived redo log; both are considered image copies.

archived redo log deletion policy

A configurable, persistent RMAN policy that governs when archived redo logs can be deleted. You can configure the policy with the `CONFIGURE ARCHIVELOG DELETION POLICY` command.

archived redo log failover

An RMAN features that enables RMAN to complete a backup even when some archived log destinations are missing logs or have logs with corrupt blocks. For example, if you back up logs in the flash recovery area that RMAN determines are corrupt, RMAN can search for logs in other archiving locations and back them up instead if they are intact.

ARCHIVELOG mode

The mode of the database in which Oracle Database copies filled online redo logs to disk. Specify the mode at database creation or with the `ALTER DATABASE ARCHIVELOG` statement.

See Also: **archived redo log**, **NOARCHIVELOG mode**

archiving

The operation in which a filled online redo log file is copied to an offline log archiving destination. An offline copy of an online redo logs is called an **archived redo log**. You must run the database in ARCHIVELOG mode to archive redo logs.

asynchronous I/O

A server process can begin an I/O and then perform other work while waiting for the I/O to complete while RMAN is either reading or writing data. RMAN can also begin multiple I/O operations before waiting for the first I/O to complete.

automatic channel allocation

The ability of RMAN to perform backup and restore tasks without requiring the use of the ALLOCATE CHANNNEL command. You can use the CONFIGURE command to specify disk and tape channels. Then, you can issue commands such as BACKUP and RESTORE at the RMAN command prompt without manually allocating channels. RMAN uses whatever configured channels that it needs in order to execute the commands.

Automatic Diagnostic Repository (ADR)

A system-managed repository for storing and organizing database trace files and other diagnostic data. ADR provides a comprehensive view of all the serious errors encountered by the database and maintains all relevant data needed for problem diagnostic and their eventual resolution. The repository contains data describing incidents, traces, dumps, alert messages, data repair records, **data integrity check** records, SQL trace information, core dumps, and so on.

The initialization parameter DIAGNOSTIC_DEST specifies the location of the ADR base, which is the directory that contains one or more ADR homes. Each ADR home is used by a product or a product instance to store diagnostic data in well-defined subdirectories. For example, diagnostic data for an Oracle database instance is stored in its ADR home, which includes an `alert` subdirectory for alert messages, a `trace` subdirectory for trace files, and so on. The easiest way to locate trace files and the alert log is to run the following SQL query: SELECT NAME, VALUE FROM V$DIAG_INFO.

Automatic Storage Management (ASM)

A vertical integration of both the file system and the volume manager built specifically for Oracle database files. ASM consolidates storage devices into easily managed disk groups and provides benefits such as mirroring and striping without requiring a third-party logical volume manager.

automatic undo management mode

A mode of the database in which undo data is stored in a dedicated **undo tablespace**. The only undo management that you must perform is the creation of the undo tablespace. All other undo management is performed automatically.

auxiliary channel

An RMAN channel that is connected to an auxiliary instance. An auxiliary channel is specified with the AUXILIARY keyword of the ALLOCATE CHANNEL or CONFIGURE CHANNEL command.

auxiliary database

(1) A database created from target database backups with the RMAN DUPLICATE command.

(2) A temporary database that is restored to a new location and then started with a new instance name during **tablespace point-in-time recovery (TSPITR)**. A TSPITR auxiliary database contains the **recovery set** and **auxiliary set**.

auxiliary destination

In a **transportable tablespace** operation, the location on disk where auxiliary set files such as the parameter file, datafiles (other than those of the tablespaces being transported), control files, and online redo logs of the auxiliary instance can be stored.

auxiliary instance

The Oracle instance associated with an auxiliary database, or the temporary instance used in tablespace point-in-time recovery (TSPITR) or a transportable tablespace operation.

auxiliary set

In TSPITR, the set of files that is not in the recovery set but which must be restored in the **auxiliary database** for the TSPITR operation to be successful. In a transportable tablespace operation, the auxiliary set includes datafiles and other files required for the tablespace transport but which are not themselves part of the **recovery set**.

backup

(1) A backup **copy** of data, that is, a database, tablespace, table, datafile, control file, or archived redo log. Backups can be physical (at the database file level) or logical (at the database object level). Physical backups can be created by using RMAN to back up one or more datafiles, control files or archived redo log files. You can create logical backups with Data Pump Export.

(2) In an RMAN context, the output of the `BACKUP` command. The output format of a backup can be a **backup set**, **proxy copy**, or **image copy**. Logs archived by the database are considered copies rather than backups.

backup and recovery

The set of concepts, procedures, and strategies involved in protecting the database against data loss due to media failure or users errors.

backup control file

A backup of the control file. You can back up the control file with the RMAN `backup` command or with the SQL statement `ALTER DATABASE BACKUP CONTROLFILE TO 'filename'`.

backup encryption

The encryption of backup sets by using one of the algorithms listed in `V$RMAN_ENCRYPTION_ALGORITHMS`. RMAN can transparently encrypt data written to backup sets and decrypt those backup sets when they are needed in a `RESTORE` operation. RMAN offers three modes of encryption: transparent, password-protected, and dual-mode.

backup mode

The database mode (also called **hot backup mode**) initiated when you issue the `ALTER TABLESPACE ... BEGIN BACKUP` or `ALTER DATABASE BEGIN BACKUP` command before taking an **online backup**. You take a tablespace out of backup mode when you issue the `ALTER TABLESPACE ... END BACKUP` or `ALTER DATABASE END BACKUP` command.

When making a user-managed backup of datafiles in an online tablespace, you must place the tablespace in backup mode to protect against the possibility of a **fractured block**. In backup mode, updates to the database create more than the usual amount of redo. Each time a block in the buffer cache becomes dirty, the database must write an image of the changed block to the redo log file, in addition to recording the changes to the data. RMAN does *not* require you to put the database in backup mode.

See Also: corrupt block

backup optimization

A configuration enabling RMAN to automatically skip backups of files that it has already backed up. You enable and disable backup optimization with the `CONFIGURE` command.

backup piece

The physical file format used to store an **RMAN backup set**. Each logical backup set contains one or more physical backup pieces.

backup retention policy

A user-defined policy for determining how long backups and archived logs need to be retained for media recovery. You can define a retention policy in terms of backup redundancy or a **recovery window**. RMAN retains the datafile backups required to satisfy the current retention policy, and any archived redo logs required for complete recovery of those datafile backups.

backup set

A backup of one or more datafiles, control files, server parameter files, and archived redo log files. Each backup set consists of one or more binary files. Each binary file is called a **backup piece**. Backup pieces are written in a proprietary format that can only be created or restored by RMAN.

Backup sets are produced by the RMAN `BACKUP` command. A backup set usually consists of only one backup piece. RMAN divides the contents of a backup set among multiple backup pieces only if you limit the backup piece size using the `MAXPIECESIZE` option of the `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` command.

See Also: unused block compression, multiplexing, RMAN

backup undo optimization

The exclusion of undo not needed for recovery of an RMAN backup because they describe already-committed transactions. For example, a user updates the `salaries` table in the `USERS` tablespace. The change is written to the `USERS` tablespace, while the before image of the data is written to the `UNDO` tablespace. A subsequent RMAN backup of the `UNDO` tablespace may not include the undo for the salary change. Backup undo optimization is built-in RMAN behavior and cannot be disabled.

backup window

A period of time during which a backup activity must complete.

backup-based duplication

A **duplicate database** that is created by restoring and recovering backups of the **target database**. This technique is an alternative to **active database duplication**.

base recovery catalog

The entirety of the **recovery catalog** schema. The base recovery catalog is distinguished from a **virtual private catalog**, which is a subset of a recovery catalog.

binary compression

A technique whereby RMAN applies a compression algorithm to data in backup sets.

block change tracking

A database option that causes Oracle to track datafile blocks affected by each database update. The tracking information is stored in a block change tracking file. When block change tracking is enabled, RMAN uses the record of changed blocks from the change tracking file to improve incremental backup performance by only reading those blocks known to have changed, instead of reading datafiles in their entirety.

block change tracking file

A binary file used by RMAN to record changed blocks to improve **incremental backup** performance. You create and rename this file with the `ALTER DATABASE` statement.

block media recovery

The recovery of specified blocks within a datafile with the Recovery Manager `RECOVER ... BLOCK` command. Block media recovery leaves the affected datafiles online and restores and recovers only the damaged or corrupted blocks.

breaking a mirror

The termination of a disk mirroring procedure so that a mirror image is no longer kept up-do-date.

channel

An RMAN channel represents one stream of data to or from a backup device. A channel can either be a `DISK` channel (used to perform disk I/O) or an **SBT** channel (used to perform I/O through a third-party **media manager**). Each allocated channel starts a new Oracle server session. The server session then performs backup, restore, and recovery operations.

See Also: **target database**

channel parallelism

Allocating multiple channels for RMAN operations.

data integrity check

An invocation of a checker, which is a diagnostic procedure registered with the Health Monitor.

checkpoint

A data structure that defines an SCN in the redo thread of a database. Checkpoints are recorded in the control file and each datafile header, and are a crucial element of recovery.

checksum

A number calculated by the database from all the bytes stored in a data or redo block. If the `DB_BLOCK_CHECKSUM` initialization parameter is enabled, then the database calculates the checksum for every datafile or online redo log block and stores it in the

block header when writing to disk. The database can use the checksum value to check consistency.

circular reuse records

Control file records containing information used by RMAN for backups and recovery operations. These records are arranged in a logical ring. When all available record slots are full, Oracle either expands the control file to make room for a new records or overwrites the oldest record. The `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter controls how many days records must be kept before they can be overwritten. The default for `CONTROL_FILE_RECORD_KEEP_TIME` is 7 days.

See Also: **noncircular reuse records**

closed backup

A backup of one or more database files taken while the database is closed. Typically, closed backups are whole database backups. If you closed the database consistently, then all the files in the backup are consistent. Otherwise, the backups are inconsistent.

See Also: **consistent shutdown**, **consistent backup**

cold backup

See **closed backup**

command file

In an RMAN context, a client-side text file containing a sequence of RMAN commands. You can run command files with the `@` or `@@` commands from within RMAN or from the operating system prompt with the `@` or `CMDFILE` parameters.

complete recovery

Recovery of one or more datafiles that applies all redo generated after the restored backup. Typically, you perform complete **media recovery** when **media failure** damages one or more datafiles or control files. You fully recover the damaged files using all redo generated since the restored backup was taken.

See Also: **incomplete recovery**

consistent backup

A **whole database backup** that you can open with the `RESETLOGS` option without performing media recovery. In other words, you do not need to apply redo to this backup to make it consistent. Unless you apply the redo generated since the consistent backup was created, however, you lose all transactions since the time of the consistent backup.

You can only take consistent backups after you have performed a **consistent shutdown** of the database. The database must not be re-opened until the backup has completed.

See Also: **fuzzy file**, **inconsistent backup**

consistent shutdown

A database shut down with the `IMMEDIATE`, `TRASACTIONAL`, or `NORMAL` options of the statement. A database shut down cleanly does not require recovery; it is already in a consistent state.

control file autobackup

The automatic backup of the current control file and server parameter file that RMAN makes after backups and, if the database is in `ARCHIVELOG` mode, after structural changes.

The control file autobackup has a default filename that allows RMAN to restore it even if the control file and recovery catalog are lost. You can override the default filename.

convert script

A script generated by the `CONVERT DATABASE` command that you can use to convert datafile formats on the destination host.

copy

To back up a bit-for-bit image of an Oracle file (Oracle datafiles, control files, and archived redo logs) onto disk. You can copy in two ways:

- Using operating system utilities (for example, the UNIX `cp` or `dd`)
- Using the RMAN `BACKUP AS COPY` command

See Also: **backup**

corrupt block

An Oracle block that is not in a recognized Oracle format, or whose contents are not internally consistent. Typically, corruptions are caused by faulty hardware or operating system problems. Oracle identifies corrupt blocks as either logically corrupt (an Oracle internal error) or media corrupt (the block format is not correct).

You can repair a media corrupt block with **block media recovery**, or dropping the database object that contains the corrupt block so that its blocks are reused for another object. If media corruption is due to faulty hardware, then neither solution will work until the hardware fault is corrected.

crash recovery

The automatic application of online redo records to a database after either a single-instance database crashes or all instances of an Oracle Real Applications Cluster configuration crash. Crash recovery only requires redo from the online logs: archived redo logs are not required.

See Also: **recover**

crosscheck

A check to determine whether files on disk or in the **media management catalog** correspond to the data in the **RMAN repository**. Because the **media manager** can mark tapes as expired or unusable, and because files can be deleted from disk or otherwise become corrupted, the RMAN repository can contain outdated information about backups. Run the `CROSSCHECK` command to perform a crosscheck.

See Also: **validation**

cumulative incremental backup

An **incremental backup** that backs up all the blocks changed since the most recent backup at level 0. When recovering with cumulative incremental backups, only the most recent cumulative incremental backup needs to be applied.

See Also: **differential incremental backup**, **incremental backup**

current incarnation

The database **incarnation** in which the database is currently generating redo.

current online redo log

The **online redo log** file in which the LGWR background process is currently logging redo records.

See Also: redo log, **redo log group**

data repair

The use of **media recovery** or **Oracle Flashback Technology** to recover lost or corrupted data.

Data Recovery Advisor

An Oracle Database tool that automatically diagnoses persistent data failures, presents repair options to the user, and executes repairs at the user's request.

database area

A location for the Oracle-managed datafiles, control files, and online redo log files. The database area is specified by the `DB_CREATE_FILE_DEST` initialization parameter.

database checkpoint

The thread checkpoint that has the lowest SCN. All changes in all enabled redo threads with SCNs prior to the database checkpoint SCN are guaranteed to have been written to disk.

See Also: checkpoint, **datafile checkpoint**

database identifier

See DBID

database point-in-time recovery (DBPITR)

The recovery of an entire database to a specified past target time, SCN, or log sequence number.

See Also: incomplete recovery, **tablespace point-in-time recovery (TSPITR)**

database registration

See **registration**

datafile checkpoint

A data structure that defines an SCN in the redo thread of a database for a particular datafile. Every datafile has a **checkpoint** SCN, which you can view in `V$DATAFILE.CHECKPOINT_CHANGE#`. All changes with an SCN lower than this SCN are guaranteed to be in the datafile.

datafile media recovery

The application of redo records to a restored datafile in order to roll it forward to a more current time. Unless you are doing **block media recovery**, the datafile must be offline while being recovered.

DBID

An internal, uniquely generated number that differentiates databases. Oracle creates this number automatically when you create the database.

destination host

The computer on which a duplicate database resides.

destination platform

When using the RMAN `CONVERT` command, the platform on which the destination database is running. The destination database is the database into which you are transporting data.

differential incremental backup

A type of **incremental backup** that backs up all blocks that have changed since the most recent backup at level 1 or level 0. For example, in a differential level 1 backup RMAN determines which level 1 or **level 0 incremental backup** is most recent and then backs up all blocks changed since that backup. Differential backups are the default type of incremental backup. When recovering using differential incremental backups, RMAN must apply all differential incremental level 1 backups since the restored datafile backup.

See Also: **cumulative incremental backup**, **incremental backup**

direct ancestral path

When multiple `OPEN RESETLOGS` operations have been performed, the incarnation path that includes the **parent incarnation** of the current database incarnation as well as each **ancestor incarnation** of the **current incarnation**.

disaster recovery

A strategic response to the loss of all data associated with a database installation. For example, a fire may destroy a server in a data center, forcing you to reinstall Oracle Database on a new server and recover the lost database from backups.

disk controller

A hardware component that is responsible for controlling one or more disk drives.

disk group

A collection of disks that are managed as a unit by **Automatic Storage Management (ASM)**. The components of a disk group include disks, files, and allocation units.

disk quota

A user-specified limit to the size of the **flash recovery area**. When the disk quota is reached, Oracle automatically deletes files that are no longer needed.

duplexed backup set

In RMAN, a duplexed **backup set** is an RMAN-generated identical copy of a backup set. Each backup piece is in the original backup set is copied, with each copy getting a unique copy number (for example, `0tcm8u2s_1_1` and `0tcm8u2s_1_2`).

duplicate database

A database created from target database backups using the RMAN duplicate command.

See Also: **auxiliary database**

expired backup

A backup whose status in the RMAN repository is `EXPIRED`, which means that the backup was not found. RMAN marks backups and copies as expired when you run a `CROSSCHECK` command and the files are absent or inaccessible.

export

The extraction of logical data (that is, not physical files) from a database into a binary file using Data Pump Export. You can then use Data Pump Import to import the data into a database.

See Also: **logical backup**

export dump file

A file created by the Data Pump Export utility. The dump file set is made up of one or more disk files that contain table data, database object metadata, and control information. The files are written in a proprietary, binary format.

failure

In the context of **Data Recovery Advisor**, a failure is a persistent data corruption that has been diagnosed by the database. A failure can manifest itself as observable symptoms such as error messages and alerts, but a failure is different from a symptom because it represents a diagnosed problem. Failures are recorded in a repository for diagnostic data located outside of the database.

For each failure, Data Recovery Advisor generates a problem statement that unambiguously describes it. Examples of failures include inaccessible datafiles and corrupted undo segments. Data Recovery Advisor maps every failure to a **repair option** or set of repair options.

failure priority

The priority of a **failure** diagnosed by **Data Recovery Advisor**. Every failure that is not closed has `CRITICAL`, `HIGH`, or `LOW` status. You can manually change the status of `HIGH` and `LOW` failures with the `CHANGE` command.

failure status

The status of a **failure** diagnosed by **Data Recovery Advisor**. Every failure has `OPEN` or `CLOSED` status.

file section

A contiguous range of blocks in a datafile. A **multisection backup** processes a large file in parallel by copying each section to a separate **backup piece**.

flash recovery area

An optional disk location that you can use to store recovery-related files such as control file and online redo log copies, archived redo log files, **flashback logs**, and RMAN backups. Oracle Database and RMAN manage the files in the flash recovery area automatically. You can specify the **disk quota**, which is the maximum size of the flash recovery area.

flashback data archive

A historical repository of transactional changes to every record in a table for the duration of the record's lifetime. A flashback data archive enables you to use some of the **logical flashback features** to transparently access historical data from far in the past.

flashback database window

The range of SCNs for which there is currently enough flashback log data to support the `FLASHBACK DATABASE` command. The flashback database window cannot extend further back than the earliest SCN in the available **flashback logs**.

flashback logs

Oracle-generated logs used to perform flashback database operations. The database can only write flashback logs to the flash recovery area. Flashback logs are written sequentially and are not archived. They cannot be backed up to disk.

flashback retention target

A user-specified time or SCN that specifies how far into the past you want to be able to perform a flashback of the database.

foreign archived redo log

An archived redo log received by a logical standby database for a LogMiner session. Unlike normal archived logs, foreign archived logs have a different DBID. For this reason, they cannot be backed up or restored on a logical standby database.

fractured block

A block in which the header and footer are not consistent at a given SCN. In a **user-managed backup**, an operating system utility can back up a datafile at the same time that DBWR is updating the file. It is possible for the operating system utility to read a block in a half-updated state, so that the block that is copied to the backup media is updated in its first half, while the second half contains older data. In this case, the block is fractured.

For non-RMAN backups, the `ALTER TABLESPACE ... BEGIN BACKUP` or `ALTER DATABASE BEGIN BACKUP` command is the solution for the fractured block problem. When a tablespace is in **backup mode**, and a change is made to a data block, the database logs a copy of the entire block image before the change so that the database can reconstruct this block if media recovery finds that this block was fractured.

full backup

A non-incremental RMAN backup. Note that "full" does not refer to how much of the database is backed up, but to the fact that the backup is not incremental. Consequently, you can make a full backup of one datafile.

full resynchronization

An RMAN operation that updates the **recovery catalog** with all changed metadata in the database's control file. You can initiate a full catalog **resynchronization** by issuing the RMAN command `RESYNC CATALOG`. (Note that it is rarely necessary to use `RESYNC CATALOG` because RMAN automatically performs resynchronizations when needed.)

fuzzy file

A datafile that contains at least one block with an SCN greater than or equal to the checkpoint SCN in the datafile header. Fuzzy files are possible because database writer does not update the SCN in the file header with each file block write. For example, this situation occurs when Oracle updates a datafile that is in **backup mode**. A fuzzy file that is restored always requires **media recovery**.

guaranteed restore point

A **restore point** for which the database is guaranteed to retain the **flashback logs** for an **Oracle Flashback Database** operation. Unlike a **normal restore point**, a guaranteed restore point does not age out of the control file and must be explicitly dropped. Guaranteed restore points utilize space in the **flash recovery area**, which must be defined.

hot backup

See **online backup**

hot backup mode

See **backup mode**

image copy

A bit-for-bit **copy** of a single datafile, archived redo log file, or control file that is:

- Usable as-is to perform recovery (unlike a backup set, which uses **unused block compression** and is in an RMAN-specific format)
- Generated with the RMAN `BACKUP AS COPY` command, an operating system command such as the UNIX `cp`, or by the Oracle archiver process

incarnation

A separate version of a database. The incarnation of the database changes when you open it with the `RESETLOGS` option, but you can recover backups from a prior incarnation so long as the necessary redo is available.

incomplete recovery

A synonym for **database point-in-time recovery (DBPITR)**.

See Also: **complete recovery**, **media recovery**, **recover**

inconsistent backup

A backup in which some of the files in the backup contain changes that were made after the files were checkpointed. This type of backup needs recovery before it can be made consistent. Inconsistent backups are usually created by taking online database backups. You can also make an inconsistent backup by backing up datafiles while a database is closed, either:

- Immediately after the crash of an Oracle instance (or, in a RAC configuration, all instances)
- After shutting down the database using `SHUTDOWN ABORT`

Inconsistent backups are only useful if the database is in `ARCHIVELOG` mode and all archived redo logs created since the backup are available.

See Also: **consistent backup**, **online backup**, **system change number (SCN)**, **whole database backup**

incremental backup

An RMAN backup in which only modified blocks are backed up. Incremental backups are classified by **level**. A **level 0 incremental backup** performs the same function as a **full backup** in that they both back up all blocks that have ever been used. The difference is that a full backup will not affect blocks backed up by subsequent incremental backups, whereas an incremental backup will affect blocks backed up by subsequent incremental backups.

Incremental backups at level 1 back up only blocks that have changed since previous incremental backups. Blocks that have not changed are not backed up. An incremental backup can be either a **differential incremental backup** or a **cumulative incremental backup**.

incrementally updated backup

An RMAN datafile copy that is updated by means of an incremental backup. An effective backup strategy is to copy a datafile, make an incremental backup, and then merge the incremental backup into the image copy. This strategy reduces the time required for **media recovery** because the image copy is updated with the latest data block changes.

instance failure

The termination of an Oracle instance due to a hardware failure, Oracle internal error, or `SHUTDOWN ABORT` statement. Crash or instance recovery is always required after an instance failure.

instance recovery

In an Oracle RAC configuration, the application of redo data to an open database by an instance when this instance discovers that another instance has crashed.

See Also: **recover**

interblock corruption

A type of block corruption in which the corruption occurs between blocks rather than within the block itself. This type of corruption can only be **logical corruption**.

intrablock corruption

A type of block corruption in which the corruption occurs within the block itself. this type of corruption can be either a **physical corruption** or **logical corruption**.

level 0 incremental backup

An RMAN **incremental backup** that backs up all data blocks in the datafiles being backed up. An incremental backup at level 0 is identical in content to a **full backup**, but unlike a full backup the level 0 backup is considered a part of the incremental backup strategy.

level of multiplexing

The number of input files simultaneously read and then written into the same RMAN **backup piece**.

LogMiner

A utility that enables log files to be read, analyzed, and interpreted by means of SQL statements.

See Also: **archived redo log**

log sequence number

A number that uniquely identifies a set of redo records in a redo log file. When Oracle fills one online redo log file and switches to a different one, Oracle automatically assigns the new file a log sequence number.

See Also: **log switch**, **redo log**

log switch

The point at which LGWR stops writing to the active redo log file and switches to the next available redo log file. LGWR switches when either the active log file is filled with redo records or you force a switch manually.

See Also: **redo log**

logical backup

A backup of database schema objects, such as tables. Logical backups are created and restored with the Oracle Data Pump Export utility. You can restore objects from logical backups using the Data Pump Import utility.

logical flashback features

The set of **Oracle Flashback Technology** features other than **Oracle Flashback Database**. The logical features enable you to view or rewind individual database objects or transactions to a past time.

logical corruption

A type of corruption in which the block has a valid checksum, the header and footer match, and so on, but the contents are logically inconsistent.

long-term backup

A backup that you want to exclude from a backup retention policy, but want to record in the recovery catalog. Typically, long-term backups are snapshots of the database that you may want to use in the future for report generation.

lost write

A write to persistent storage that the database believes has occurred based on information from the I/O subsystem, when in fact the write has not occurred.

Mean Time To Recover (MTTR)

The time required to perform recovery.

media failure

Damage to the disks containing any of the files used by Oracle, such as the datafiles, archived redo log files, or control file. When Oracle detects media failure, it takes the affected files offline.

See Also: **media recovery**

media manager

A third-party networked backup system that can be integrated with Recovery Manager so that database backups can be written directly to tertiary storage.

media manager multiplexing

Multiplexing in which the **media manager** rather than RMAN manages the mixing of blocks during an RMAN **backup**. One type of media manager multiplexing occurs when the media manager writes the concurrent output from multiple RMAN channels to a single sequential device. Another type occurs when a backup mixes database files and non-database files on the same tape.

media management catalog

A catalog of records maintained by a **media manager**. This catalog is completely independent from the RMAN **recovery catalog**. The **Oracle Secure Backup** catalog is an example of a media management catalog.

media management library

A software library that RMAN can use to back up to tertiary storage. An **SBT** interface conforms to a published API and is supplied by a media management vendor. **Oracle Secure Backup** includes an SBT interface for use with RMAN.

media recovery

The application of redo or incremental backups to a restored backup datafile or individual data block.

When performing media recovery, you can recover a database, tablespace, datafile, or set of blocks within a datafile. Media recovery can be either **complete recovery** (in which all changes in the redo logs are applied) or **incomplete recovery** (in which only changes up to a specified point in time are applied). Media recovery is only possible when the database is in `ARCHIVELOG` mode.

See Also: **block media recovery**, **recover**

mirroring

Maintaining identical copies of data on one or more disks. Typically, mirroring is performed on duplicate hard disks at the operating system level, so that if one of the disks is unavailable, the other disk can continue to service requests without interruptions. When mirroring files, Oracle Database writes once while the operating system writes to multiple disks. When **multiplexing** files, Oracle Database writes the same data to multiple files.

MTTR

See **Mean Time To Recover (MTTR)**

multiplexed backup set

A backup set that contains blocks from multiple input files. For example, you could multiplex 10 datafiles into one backup set. Only whole files, never partial files, are included in a backup set.

multiplexing

The meaning of the term depends on which files are multiplexed:

- **online redo logs**

  The automated maintenance of more than one identical copy of the online redo log.

- **control file**

  The automated maintenance of more than one identical copy of a database's control file.

- **backup set**

  The RMAN technique of reading database files *simultaneously* from the disks and then writing the blocks to the *same* backup piece.

- **archived redo logs**

  The Oracle archiver process is able to archive multiple copies of a redo log.

See Also: **mirroring**

multisection backup

An RMAN backup set in which each **backup piece** contains a **file section**, which is a contiguous range of blocks in a datafile. Note that a multisection **backup set** contains multiple backup pieces, but a backup set never contains only a part of a datafile.

You create a multisection backup by specifying the `SECTION SIZE` parameter on the `BACKUP` command. An RMAN **channel** can process each file section independently, either serially or in parallel. Thus, in a multisection backup, multiple channels can back up a single file.

native transfer rate

In a tape drive, the speed of writing to a tape without compression. This speed represents the upper limit of the backup rate.

NOARCHIVELOG mode

The mode of the database in which Oracle does not require filled online redo logs to be archived before they can be overwritten. Specify the mode at database creation or change it with the `ALTER DATABASE NOARCHIVELOG` command.

Note that running in `NOARCHIVELOG` mode severely limits the possibilities for recovery of lost or damaged data.

See Also: **archived redo log**, **ARCHIVELOG mode**

noncircular reuse records

Control file records containing critical information needed by the Oracle database. These records are never automatically overwritten. Some examples of information in noncircular reuse records include the locations of datafiles and online redo logs.

See Also: **circular reuse records**

normal restore point

A label for an SCN or time. For commands that support an SCN or time, you can often specify a **restore point**. Normal restore points exist in the circular list and can be overwritten in the control file. However, if the restore point pertains to an **archival backup**, then it will be preserved in the **recovery catalog**.

obsolete backup

A backup that is not need to satisfy the current backup **retention policy**. For example, if your retention policy dictates that you must maintain one backup of each datafile, but you have two backups of datafile 1, then the second backup of datafile 1 is considered obsolete.

offline normal

A tablespace is offline normal when taken offline with the `ALTER TABLESPACE ... OFFLINE NORMAL` statement. The datafiles in the tablespace are checkpointed and do not require recovery before being brought online. If a tablespace is not taken offline normal, then its datafiles must be recovered before being brought online.

offsite backup

An **SBT** backup that requires retrieval by the **media manager** before RMAN can restore it. You can list offsite backups with `RESTORE ... PREVIEW`.

online backup

A backup of one or more datafiles taken while a database is open and the datafiles are online. When you make a user-managed backup while the database is open, you must put the tablespaces in **backup mode** by issuing an `ALTER TABLESPACE BEGIN BACKUP` command. (You can also use `ALTER DATABASE BEGIN BACKUP` to put all tablespaces in your database into backup mode in one step.)

You should not put tablespaces in backup mode when performing backups with RMAN.

online redo log

The online redo log is a set of two or more files that record all changes made to the database. Whenever a change is made to the database, Oracle generates a redo record in the redo buffer. The LGWR process writes the contents of the redo buffer into the online redo log.

The **current online redo log** is the one being written to by LGWR. When LGWR gets to the end of the file, it performs a **log switch** and begins writing to a new log file. If you run the database in `ARCHIVELOG` mode, then each filled online redo log file must be copied to one or more archiving locations before LGWR can overwrite them.

See Also: **archived redo log**

online redo log group

The Oracle online redo log consists of two or more online redo log groups. Each group contains one or more identical online redo log members. An **online redo log member** is a physical file containing the redo records.

online redo log member

A physical online redo log file within an **online redo log group**. Each log group must have one or more members. Each member of a group is identical.

operating system backup

See **user-managed backup**

operating system backup and recovery

See **user-managed backup and recovery**

Oracle Flashback Database

The return of the whole database to a prior consistent SCN by means of the `FLASHBACK DATABASE` command in RMAN or SQL. A database flashback is different from traditional media recovery because it does not involve the restore of physical files, instead restoring your current datafiles to past states using saved images of changed data blocks. This feature uses **flashback logs** and archived redo logs.

Oracle Flashback Technology

A set of Oracle Database features that provide an additional layer of data protection. These features include Oracle Flashback Query, Oracle Flashback Version Query, Oracle Flashback Transaction Query, Oracle Flashback Transaction, Oracle Flashback Table, Oracle Flashback Drop, and **Oracle Flashback Database**.

You can use flashback features to view past states of data and rewind parts or all of your database. In general, flashback features are more efficient and less disruptive than media recovery in most situations in which they apply.

Oracle-managed file

A database file managed by the Oracle Managed Files feature.

Oracle Managed Files (OMF)

A service that automates naming, location, creation, and deletion of database files such as control files, redo log files, datafiles and others, based on a few initialization parameters. You can use Oracle-managed files on top of a traditional file system supported by the host operating system, for example, VxFS or ODM. It can simplify many aspects of the database administration by eliminating the need to devise your own policies for such details.

Oracle Secure Backup

An Oracle media manager that supplies reliable data protection through file system backup to tape. The Oracle Secure Backup **SBT** interface also enables you to use RMAN to back up Oracle databases. All major tape drives and tape libraries in SAN, Gigabit Ethernet, and SCSI environments are supported.

Oracle VSS writer

A service on Windows systems that acts as coordinator between an Oracle database instance and other **Volume Shadow Copy Service (VSS)** components, enabling data providers to create a shadow copy of files managed by the Oracle instance. For example, the Oracle VSS writer can place datafiles in hot backup mode to provide a recoverable copy of these datafiles in a shadow copy set.

Oracle-suggested backup strategy

A backup strategy available through a wizard in Oracle Enterprise Manager. The strategy involves periodically applying a level 1 **incremental backup** to a level 0 backup to create an **incrementally updated backup**. If run daily, this strategy provides 24 hour **point-in-time recovery** from disk.

orphaned backups

Backups that were not made in the **direct ancestral path** of the **current incarnation** of the database. Orphaned backups cannot be used in the current incarnation.

parallel recovery

A form of recovery in which several processes simultaneously apply changes from redo log files. The `RECOVERY_PARALLELISM` initialization parameter determines the level of parallelism for instance and crash recovery. You can use the `PARALLEL` and `NOPARALLEL` options of the `RECOVER` command to control parallelism for media recovery.

Oracle Database automatically chooses the optimum degree of recovery parallelism. In most cases, manually setting the level of parallelism for instance, crash, or media recovery is not recommended or necessary.

parent incarnation

The database **incarnation** from which the **current incarnation** branched following an `OPEN RESETLOGS` operation.

partial resynchronization

A type of **resynchronization** in which RMAN transfers data about archived logs, backup sets, and datafile copies from the target control file to the **recovery catalog**.

password file

A file created by the `ORAPWD` command, and required if you wish to connect using the `SYSDBA` or `SYSOPER` privileges over a network. For details on password files, see the *Oracle Database Administrator's Guide*.

physical backup

A backup of physical files. A physical backup contrasts with a logical backup such as a table export.

physical corruption

A type of corruption in which the database does not recognize a **corrupt block**. The database may not recognize the block because the checksum is invalid, the block contains all zeros, or the header and footer of the block do not match.

physical schema

The datafiles, control files, and redo logs in a database at a given time. Issue the RMAN `REPORT SCHEMA` command to obtain a list of tablespaces and datafiles.

physical standby database

A copy of a production database that you can use for disaster protection.

point-in-time recovery

The incomplete recovery of database files to a noncurrent time. Point-in-time recovery is also known as **incomplete recovery**.

See Also: media recovery, **recover**

problem

A critical error in the database that is recorded in the **Automatic Diagnostic Repository (ADR)**. Critical errors include internal errors and other severe errors. Each problem has a problem key, which is a set of attributes that describe the problem. The problem key includes the ORA error number, error parameter values, and other information.

proxy copy

A backup in which the **media manager** manages the transfer of data between the media storage device and disk during RMAN backup and restore operations.

raw device

A disk or partition without a file system. Thus, you cannot use `ls`, Windows Explorer, and so on to view their contents. The raw partition appears to Oracle Database as a single file.

recover

To recover a database file or a database is typically to perform **media recovery**, **crash recovery**, or **instance recovery**. This term can also be used generically to refer to reconstructing or re-creating lost data by any means.

See Also: complete recovery, **incomplete recovery**

recovery

When used to refer to a database file or a database, the application of redo data or incremental backups to database files in order to reconstruct lost changes. The three types of recovery are **instance recovery**, **crash recovery**, and **media recovery**. Oracle

performs the first two types of recovery automatically using online redo records; only media recovery requires you to restore a backup and issue commands.

See Also: **complete recovery**, **incomplete recovery**

recovery catalog

A set of Oracle tables and views used by RMAN to store RMAN repository information about one or more Oracle databases. RMAN uses this metadata to manage the backup, restore, and recovery of Oracle databases.

Use of a recovery catalog is optional unless you use RMAN in a Data Guard environment, in which case it is required. The primary storage for RMAN repository information for a database is always in the control file of the database. A recovery catalog is periodically updated with RMAN repository data from the control file. In the event of the loss of your control file, the recovery catalog can provide most or all of the lost metadata required for restore and recovery of your database. The recovery catalog can also store records of archival backups and RMAN stored scripts for use with target databases.

See Also: **recovery catalog database**

recovery catalog database

An Oracle database that contains a recovery catalog schema. You should not store the recovery catalog in the target database.

recovery catalog schema

The **recovery catalog database** schema that contains the **recovery catalog** tables and views.

Recovery Manager (RMAN)

The primary utility for physical backup and recovery of Oracle databases. RMAN keeps records of Oracle databases in its own structure called an RMAN repository, manages storage of backups, validates backups. You can use it with or without the central information repository called a **recovery catalog**. If you do not use a recovery catalog, then RMAN uses the database's control file to store information necessary for backup and recovery operations. You can use RMAN in conjunction with third-party media management software to back up files to tertiary storage.

See Also: **backup piece**, **backup set**, **copy**, **media manager**, **recovery catalog**

recovery set

One or more tablespaces that are being recovered to an earlier point in time during **tablespace point-in-time recovery (TSPITR)**. After TSPITR, all database objects in the recovery set have been recovered to the same point in time.

See Also: **auxiliary set**

recovery window

A recovery window is one type of RMAN backup **retention policy**, in which the DBA specifies a period of time and RMAN ensures retention of backups and archived redo logs required for point-in-time recovery to any time during the recovery window. The interval always ends with the current time and extends back in time for the number of days specified by the user.

For example, if the retention policy is set for a recovery window of seven days, and the current time is 11:00 AM on Tuesday, RMAN retains the backups required to allow point-in-time recovery back to 11:00 AM on the previous Tuesday.

recycle bin

A data dictionary table containing information about dropped objects. Dropped tables and any associated objects such as indexes, constraints, nested tables, and so on are not removed and still occupy space. The Flashback Drop feature uses the recycle bin to retrieve dropped objects.

redo log

A redo log can be either an **online redo log** or an **archived redo log**. The online redo log is a set of two or more redo log groups that records all changes made to Oracle datafiles and control files. An archived redo log is a copy of an online redo log that has been written to an offline destination.

redo log group

Each online redo log member (which corresponds to an online redo log file) belongs to a redo log group. Redo log groups contain one or more members. A redo log group with more than one member is called a multiplexed redo log group. The contents of all members of a redo log group are identical.

redo thread

The redo generated by an instance. If the database runs in a single instance configuration, then the database has only one thread of redo.

redundancy

In a **retention policy**, the setting that determines many copies of each backed-up file to keep. A redundancy-based retention policy is contrasted with retention policy that uses a **recovery window**.

redundancy set

A set of backups enabling you to recover from the failure or loss of any Oracle database file.

registration

In RMAN, the execution of a `REGISTER DATABASE` command in order to record the existence of a target database in the **recovery catalog**. A **target database** is uniquely identified in the catalog by its **DBID**. You can register more than one database in the same catalog, and also register the same database in multiple catalogs.

repair

In the context of **Data Recovery Advisor**, a repair is an action or set of actions that fixes one or more **failure**s. Examples repairs include **block media recovery**, **datafile media recovery**, **Oracle Flashback Database**, and so on.

repair option

In the context of **Data Recovery Advisor**, one possible technique for repairing a **failure**. Different repair options are intended to fix the same problem, but represent different advantages and disadvantages in terms of repair time and data loss.

RESETLOGS

A technique for opening a database that archives any current online redo logs (if using `ARCHIVELOG` mode), resets the log sequence number to 1, and clears the online redo logs. An `ALTER DATABASE OPEN RESETLOGS` statement begins a new database incarnation. The starting SCN for the new incarnation, sometimes called the

RESETLOGS SCN, is the incomplete recovery SCN of the media recovery preceding the OPEN RESETLOGS, plus one.

An ALTER DATABASE OPEN RESETLOGS statement is required after incomplete recovery or recovery with a backup control file. An OPEN RESETLOGS operation does not affect the recoverability of the database. Backups from before the OPEN RESETLOGS operation remain valid and can be used along with backups taken after the OPEN RESETLOGS operation to repair any damage to the database.

resilvering a mirror

Informing the operating system or the hardware managing the mirror that you want to refresh a broken mirror from the half that is up-to-date and then maintain both sides of the mirror.

restartable backup

The feature that enables RMAN to back up only those files that have not been backed up since a specified date. The unit of restartability is last completed backup set or image copy. You can use this feature after a backup fails to back up the parts of the database missed by the failed backup.

restore

The replacement of a lost or damaged file with a backup. You can restore files either with commands such as UNIX cp or the RMAN RESTORE command.

restore failover

The automatic search by RMAN for usable backups in a restore operation if a corrupted or inaccessible backup is found.

restore optimization

The default behavior in which RMAN avoids restoring datafiles from backup when possible.

restore point

A user-defined a name associated with an SCN of the database corresponding to the time of the creation of the restore point. A restore point can be a **guaranteed restore point** or a **normal restore point**.

resynchronization

The operation that updates the **recovery catalog** with current metadata from the target database control file. You can initiate a **full resynchronization** of the catalog by issuing a RESYNC CATALOG command. A **partial resynchronization** transfers information to the recovery catalog about archived redo log files, backup sets, and datafile copies. RMAN resynchronizes the recovery catalog automatically when needed.

retention policy

See **backup retention policy**

reverse resynchronization

In a Data Guard environment, the updating of a primary or standby database control file with metadata obtained from the **recovery catalog**. For example, if you configure persistent RMAN settings for a standby database that is not the connected target database, then RMAN performs a reverse resynchronization the next time RMAN connects as target to the standby database. In this way, the recovery catalog keeps the metadata in the control files in a Data Guard environment up to date.

RMAN

See **Recovery Manager (RMAN)**

RMAN backup job

The set of BACKUP commands executed within a single **RMAN session**. For example, assume that you start the RMAN client, execute BACKUP DATABASE, BACKUP ARCHIVELOG, and RECOVER COPY, and then exit the RMAN client. The RMAN backup job consists of the database backup and the archived redo log backup.

RMAN client

An Oracle Database executable that interprets commands, directs server sessions to execute those commands, and records its activity in the target database control file. The RMAN executable is automatically installed with the database and is typically located in the same directory as the other database executables. For example, the RMAN client on Linux is named rman and is located in $ORACLE_HOME/bin.

RMAN job

The set of RMAN commands executed in an **RMAN session**. For example, assume that you start the RMAN client, execute BACKUP DATABASE, BACKUP ARCHIVELOG, and RECOVER COPY, and then exit the RMAN client. The RMAN job consists of the two backups and the roll forward of the datafile copy.

RMAN maintenance commands

Commands that you can use the manage RMAN metadata records and backups. The maintenance commands are CATALOG, CHANGE, CROSSCHECK, and DELETE.

RMAN repository

The record of RMAN metadata about backup and recovery operations on the target database. The authoritative copy of the RMAN repository is always stored in the control file of the target database. A **recovery catalog** can also be used for longer-term storage of the RMAN repository, and can serve as an alternate source of RMAN repository data if the control file of your database is lost.

See Also: **recovery catalog database**, **resynchronization**

RMAN session

An RMAN session begins when the RMAN client is started and ends when you exit from the client or the RMAN process is terminated. Multiple RMAN commands can be executed in a single RMAN session.

rollback segments

Database segments that record the before-images of changes to the database.

rolling back

The use of rollback segments to undo uncommitted changes applied to the database during the **rolling forward** stage of **recover**.

rolling forward

The application of redo records or incremental backups to datafiles and control files in order to **recover** changes to those files.

See Also: **rolling back**

RUN block

A series of RMAN commands that are executed sequentially.

SBT

System Backup to Tape. This term specifies a nondisk backup device type, typically a tape library or tape drive. RMAN supports channels of type disk and SBT.

shadow copy

In the **Volume Shadow Copy Service (VSS)** infrastructure on Windows, a consistent snapshot of a component or volume.

snapshot control file

A copy of a database control file created in an operating system-specific location by Recovery Manager. RMAN creates the snapshot control file so that it has a consistent version of a control file to use when either resynchronizing the recovery catalog or backing up the control file.

source database

The database that you are copying when you create a **duplicate database**.

source host

The host on which a **source database** resides.

source platform

When using the RMAN `CONVERT` command, the platform on which the source database is running. The source database contains the data to be transported to a database running on a different platform.

split mirror backup

A backup of database files that were previously mirrored. Some third-party tools allow you to use **mirroring** a set of disks or logical devices, that is, maintain an exact duplicate of the primary data in another location. Splitting a mirror involves separating the file copies so that you can use them independently. With the `ALTER SYSTEM SUSPEND/RESUME` database feature, you can suspend I/O to the database, split the mirror, and make a backup of the split mirror.

stored script

A sequence of RMAN commands stored in the **recovery catalog**. Stored scripts can be global or local. Global scripts can be shared by all databases registered in the recovery catalog.

synchronous I/O

A server process can perform only one task at a time while RMAN is either reading or writing data.

system change number (SCN)

A stamp that defines a committed version of a database at a point in time. Oracle assigns every committed transaction a unique SCN.

tablespace destination

In a **transportable tablespace** operation, the location on disk which (by default) contains the datafile copies and other output files when the tablespace transport command completes.

tablespace point-in-time recovery (TSPITR)

The recovery of one or more non-`SYSTEM` tablespaces to a noncurrent time. You use RMAN to perform TSPITR.

tag

Identifier for an RMAN backup. If you generate a backup set, then the tag is assigned to each backup piece rather than to the backup set. If you do not specify a tag for a backup, then RMAN assigns one automatically.

target database

In an RMAN environment, the database to which you are connected as `TARGET`. The target database is the database on which you are performing RMAN operations.

target host

The computer on which a **target database** resides.

target instance

In an RMAN environment, the instance associated with a **target database**.

tempfile

A file that belongs to a temporary tablespace and is created with the `TEMPFILE` option. Temporary tablespaces cannot contain permanent database objects such as tables, and are typically used for sorting. Because tempfiles cannot contain permanent objects, RMAN does not back them up. RMAN does keep track of the locations of tempfiles in the control file, however, and during recovery re-creates the tempfiles as needed at those locations.

transport script

A script generated by the `CONVERT DATABASE` command. This script contains SQL statements used to create the new database on the **destination platform**.

transportable tablespace

A feature that transports a set of tablespaces from one database to another, or from one database to itself. Transporting a tablespace into a database is like creating a tablespace with preloaded data.

transportable tablespace set

Datafiles for the set of tablespaces in a transportable tablespace operation, and an export file containing metadata for the set of tablespaces. You use Data Pump Export to perform the export.

trial recovery

A simulated recovery initiated with the `RECOVER ... TEST` command in RMAN or SQL*Plus. A trial recovery applies redo in a way similar to normal **media recovery**, but it never writes its changes to disk and it always rolls back its changes. Trial recovery occurs only in memory.

undo retention period

The minimum amount of time that Oracle Database attempts to retain old undo data in the **undo tablespace** before overwriting it. Old (committed) undo data that is older than the current undo retention period is said to be expired. Old undo data with an age that is less than the current undo retention period is said to be unexpired.

undo tablespace

A dedicated tablespace that stores only undo information when the database is run in **automatic undo management mode**.

unused block compression

A feature by which RMAN reduces the size of datafile backup sets by skipping data blocks. RMAN always skips blocks that have never been used. Under certain conditions, which are described in the `BACKUP AS BACKUPSET` entry in *Oracle Database Backup and Recovery Reference*, RMAN also skips previously used blocks that are not currently being used.

user-managed backup

A backups made using a non-RMAN method, for example, using an operating system utility. For example, you can make a user-managed backup by running the `cp` command on Linux or the `COPY` command on Windows. A user-managed backup is also called an **operating system backup**.

user-managed backup and recovery

A backup and recovery strategy for an Oracle database that does not use RMAN. This term is equivalent to **operating system backup and recovery**. You can back up and restore database files using operating system utilities (for example, the `cp` command in UNIX), and recover using the SQL*Plus `RECOVER` command.

validation

In an RMAN context, a test that checks database files for block corruption or checks a backup set to determine whether it can be restored. RMAN can check for both physical and logical block corruption.

virtual private catalog

A subset of the metadata in a base **recovery catalog** to which a database user is granted access. The owner of a base recovery catalog can grant or revoke restricted access to the recovery catalog to other database users. Each restricted user has full read/write access to his own virtual private catalog.

Volume Shadow Copy Service (VSS)

An infrastructure on Windows server platforms that enables requestors, writers, and providers to participate in the creation of a consistent snapshot called a **shadow copy**. The VSS service uses well-defined COM interfaces. See *Oracle Database Platform Guide for Microsoft Windows* to learn how to use RMAN with VSS.

whole database backup

A **backup** of the control file and all datafiles that belong to a database.

Index

Symbols

%U substitution variable, 8-4

A

ABORT option
  SHUTDOWN statement, 29-1, 29-2
ABORT option, SHUTDOWN statement, 28-16
active database duplication, 4-11, 23-2
Advanced Security Option, 6-7, 7-4
ADVISE FAILURE command, 14-4, 14-10
alert log, 11-5, 22-1
ALLOCATE CHANNEL command, 6-1, 8-4
  MAXPIECESIZE option, 6-4
ALLOCATE command, 5-4
ALLOW ... CORRUPTION clause, RECOVER command, 28-23
ALTER DATABASE statement
  CLEAR LOGFILE clause, 29-14
  END BACKUP clause, 27-8
  OPEN RESETLOGS clause, 12-28
  RECOVER clause, 28-6, 28-10, 28-12
  RESETLOGS option, 28-17
ALTER SYSTEM statement
  KILL SESSION clause, 22-13
  RESUME clause, 27-13
  SUSPEND clause, 27-13
ALTER TABLESPACE statement
  BEGIN BACKUP clause, 27-5, 27-7
  END BACKUP option, 27-7
application errors, 1-3
archival backups, 1-3, 8-23, 11-15
  restoring with DUPLICATE, 23-21
archived redo log deletion policies, 5-26, 5-27, 8-13
archived redo log files
  applying during media recovery, 28-3, 28-5, 28-6
  backing up, 8-12
    using RMAN, 8-11
    with other backups, 8-12
  cataloging, 11-16
  changing default location, 28-5
  corrupted, 28-18
  deleting, 13-5, 28-11
  deletion after backup, 8-11
  failover, 8-11
  incompatible format, 28-18
  location during recovery, 28-3
  loss of, 28-13
  restoring using RMAN, 17-8
ARCHIVELOG mode
  backups in, 2-4
AS SELECT clause
  CREATE TABLE statement, 29-9
authentication, RMAN, 2-3
autobackups, control file, 7-12, 8-8, 8-24
  configuring, 5-7
  format, 5-7
automated repairs
  Data Recovery Advisor, 1-8
automatic channel allocation, 6-1
automatic channels, 3-3, 3-4
  configuring, 6-2
  naming conventions, 3-5
  overriding, 6-1
Automatic Diagnostic Repository (ADR), 5-11, 7-12, 11-4, 14-3, 15-2, 15-3, 22-1
Automatic Storage Management (ASM)
  backups to, 8-3
  duplicating databases and, 23-14
Automatic Workload Repository (AWR), 11-10
AUTORECOVERY option
  SET statement, 28-4
auxiliary channels, 23-2
auxiliary instance parameter file
  with TRANSPORT TABLESPACE, 24-5
availability
  of RMAN backups, 11-14
AVAILABLE option
  of CHANGE command, 11-15

B

backup and recovery
  definition, 1-1
  introduction, 1-1
  new features, xxiii
  solutions, 1-3
  strategy, 1-2
  user-managed, 1-3
  VSS-enabled, xxvi
BACKUP command, 2-4, 2-5, 3-4, 3-9, 5-20, 5-23, 6-2,

6-4, 6-6, 7-1, 7-3, 8-1, 8-13
ARCHIVELOG option, 8-12
AS BACKUPSET option, 7-4
AS COMPRESSION BACKUPSET option, 8-6
AS COPY option, 2-4, 7-8
BACKUPSET option, 6-7, 7-10, 7-11, 8-26, 8-28
CHANNEL option, 5-6
COMPRESSED BACKUPSET option, 8-6
COPIES parameter, 7-10
COPY OF option, 7-10, 7-11, 8-26, 8-29
CURRENT CONTROLFILE option, 8-9
DATABASE option, 8-7
DATAFILE option, 8-8
DB_FILE_NAME_CONVERT parameter, 7-8
DELETE INPUT option, 8-13, 11-20
DELETE option, 8-11
DEVICE TYPE clause, 5-3, 5-23, 8-2, 8-9
DURATION parameter, 9-13
FILESPERSET parameter, 7-7
FOR RECOVER OF COPY option, 8-17
FORMAT parameter, 2-5, 2-7, 5-10, 5-12, 7-5, 7-10, 8-3
INCREMENTAL option, 2-5, 2-6, 2-7, 8-14, 8-16, 8-17
KEEP option, 8-23, 8-25
MAXSETSIZE parameter, 9-1
NOT BACKED UP clause, 8-13
PLUS ARCHIVELOG option, 8-12
PROXY ONLY option, 7-8
PROXY option, 7-8
RECOVERY AREA option, 8-26
SECTION SIZE parameter, 7-3, 9-3
SPFILE option, 8-10
TABLESPACE option, 8-8
TAG parameter, 2-5, 2-7, 8-5
VALIDATE option, 2-7, 14-3, 14-8, 15-4
BACKUP CONTROLFILE clause
ALTER DATABASE statement, 27-2
BACKUP COPIES parameter
CONFIGURE command, 6-5
backup encryption, 6-7, 7-4, 13-3
decrypting backups, 17-10
default algorithm, 6-7
dual-mode, 6-8, 9-11
overview, 9-10
password, 6-8, 9-11
transparent, 6-8, 9-11
backup mode, 7-9
ending with ALTER DATABASE END BACKUP, 27-8
for online user-managed backups, 7-2, 27-5
instance failure, 27-7
backup optimization, 8-13
configuring, 5-23, 9-3
definition, 5-23, 8-13
disabling, 5-23, 5-25
enabling, 5-23, 5-25
redundancy and, 5-24
retention policies and, 5-24
backup pieces, 7-3
definition, 2-4
maximum size, 6-4
names, 7-5
names on tape, 5-12
backup retention policies, 1-3, 3-7, 5-14
affect on backup optimization, 5-24
configuring, 5-21
configuring for redundancy, 5-21
definition, 7-17
disabling, 5-22
exempt backups, 8-23, 11-15
recovery window, 7-17
recovery windows, 5-22
redundancy, 7-17, 7-19
backup sets, 2-4, 7-1
backing up, 7-11, 8-26
compressed, 5-4, 6-6, 7-4, 8-6
configuring as default, 5-4
configuring maximum size, 6-4
crosschecking, 11-12
duplexing, 9-6
how RMAN generates, 7-6
limiting size, 7-6
maximum size, 6-4, 9-1
multiplexed, 2-4, 6-4, 7-6, 8-5, 21-4
naming, 7-5
overview, 7-3
specifying maximum size, 7-5
specifying number, 7-6
testing restore of, 17-8
unused block compression, 7-4
Backup Solutions Program (BSP), 3-7
backup strategy
flash recovery area, 5-14
backup tags, RMAN, 8-4
backup techniques, comparison, 1-4
backup windows, 9-13
backup-based duplication, 23-2
backups
archival, 1-3, 8-23
archived redo logs
using RMAN, 8-11
availability, 11-14
backup sets, 8-26
backups of, 7-11
closed, 27-3
consistent, 27-3
making using RMAN, 7-1
control file, 8-8, 27-10
control files, 27-10
binary, 27-10
correlating RMAN channels with, 22-9, 22-10
crosschecking, 11-12
cumulative incremental, 7-14
datafile
using RMAN, 8-28, 8-29
DBVERIFY utility, 27-17
default type for RMAN, 5-4
determining datafile status, 27-2
duplexing, 6-5, 9-6

excluding tablespaces from backups, 6-6
exempt from retention policy, 11-15
expired, deleting, 11-23
generating reports for, 10-2, 10-11
image copies, 7-8
inconsistent, 27-3
making using RMAN, 7-1
incremental, 7-13, 8-14, 9-7, 9-8
incrementally updated, 8-17
listing files needed, 27-1
logical, 1-2
long-term, 1-3
managing, 11-1
multisection, 3-5, 7-3, 15-5
NOARCHIVELOG mode, 8-10
obsolete, 7-19, 11-23
offline, 27-4
offsite, 17-6
optimizing, 5-23, 8-13
orphaned, 13-7
physical, 1-2
previewing, 17-5
read-only tablespaces, 27-9
recovering pre-RESETLOGS, 16-20
recovery catalog, 12-13
Recovery Manager, 8-1
reporting objects needing backups, 10-11
restartable, 9-12
restoring user-managed, 28-2
server parameter files, 8-10
skipping files during, 9-6
split mirror, 7-9
using RMAN, 9-8
stored scripts, 12-3, 12-15
tablespace, 27-6
using RMAN, 8-8, 8-28, 8-29
testing RMAN, 15-3, 15-4, 15-6
using media manager, 5-11
user-managed, 27-1
validating, 15-4, 15-6
verifying, 27-17
whole database, 8-7, 27-3
BEGIN BACKUP clause
ALTER TABLESPACE statement, 27-5
binary compression for backups, 8-6
block change tracking, 1-4, 7-15, 8-20
disk space used for, 8-21
enabling and disabling, 8-21, 8-22
moving the change tracking file, 8-22
block corruptions, 1-3
stored in
V$DATABASE_BLOCK_CORRUPTION, 15-4
block media recovery, 1-3, 15-3
BSP. *See* Backup Solutions Program (BSP)
BZIP2 compression algorithm, 6-6, 7-4

C

cancel-based media recovery, 28-16
canceling RMAN commands, 22-13
CATALOG command, 11-16
START WITH parameter, 12-9
CHANGE command
AVAILABLE option, 11-15
DB_UNIQUE_NAME parameter, 12-25
RESET DB_UNIQUE_NAME option, 3-8
UNCATALOG option, 11-18
CHANGE FAILURE command, 14-14
channels, RMAN, 3-3
auxiliary, 23-2
configuring, 5-4
configuring advanced options, 6-1
definition, 2-3, 3-3
generic, 5-4
naming conventions, 3-5
Oracle RAC environment, 6-2
parallel, 5-5
character sets
setting for use with RMAN, 4-2
circular reuse records, 11-3
CLEAR LOGFILE clause
of ALTER DATABASE, 29-14
client, RMAN, 2-1, 3-1, 3-6
cold failover cluster
definition, 27-8
command files, RMAN, 2-8
command interface
RMAN, 3-3
commands, Recovery Manager
ADVISE FAILURE, 14-4, 14-10
ALLOCATE CHANNEL, 5-4, 6-1, 6-4, 8-4
BACKUP, 2-4, 2-5, 2-6, 2-7, 3-4, 3-9, 5-3, 5-6, 5-10, 5-12, 5-20, 5-23, 6-2, 6-4, 6-6, 6-7, 7-1, 7-3, 7-4, 7-7, 7-8, 7-10, 7-11, 8-1, 8-2, 8-5, 8-6, 8-7, 8-8, 8-9, 8-10, 8-11, 8-12, 8-13, 8-14, 8-16, 8-17, 8-23, 8-25, 8-26, 8-28, 8-29
PROXY ONLY option, 7-8
PROXY option, 7-8
BACKUP CURRENT CONTROLFILE, 8-9
canceling, 22-13
CATALOG, 11-16
CHANGE, 3-8, 11-12
CHANGE FAILURE, 14-14
CONFIGURE, 3-8, 5-4, 5-21, 5-35, 6-1, 6-4, 6-9, 6-10
CREATE CATALOG, 12-6, 12-11
CREATE SCRIPT, 12-16
CROSSCHECK, 11-12
DELETE, 11-12, 11-16, 11-19
DROP CATALOG, 12-33
DROP DATABASE, 11-24
DUPLICATE, 23-1
EXECUTE SCRIPT, 12-15, 12-17
EXIT, 2-3
FLASHBACK DATABASE, 5-28, 12-28
GRANT, 12-11
how RMAN interprets, 3-3
IMPORT CATALOG, 12-31
LIST, 2-8, 10-2, 10-3, 12-28, 14-6

INCARNATION option, 10-9, 12-28
MAXSETSIZE, 6-4
piping, 4-12
PRINT SCRIPT, 12-19
RECOVER, 13-5
REPAIR FAILURE, 14-12, 14-14
REPLACE SCRIPT, 12-17
REPORT, 2-9, 10-11
NEED BACKUP option, 10-11
RESET DATABASE
INCARNATION option, 12-28
RESTORE, 17-3
RESYNC CATALOG, 12-15, 12-22, 12-24
FROM CONTROLFILECOPY option, 12-15
REVOKE, 12-12
SET, 6-8
SHOW, 2-3, 5-2
SPOOL, 14-13
SWITCH, 17-16
terminating, 22-13
UNREGISTER DATABASE, 12-26
UPGRADE CATALOG, 12-29
VALIDATE, 14-3, 14-8, 15-4
commands, SQL*Plus
RECOVER
UNTIL TIME option, 28-16
SET, 28-4, 28-6, 28-10, 28-12
comments in RMAN syntax, 4-4
COMPATIBLE initialization parameter, 6-7
complete recovery
overview, 17-1
procedures, 28-7
compressed backups, 5-4, 8-6
algorithms, 6-6
CONFIGURE command
AUXNAME option, 6-10
BACKUP OPTIMIZATION option, 5-25
CHANNEL option, 5-4, 6-1
CONTROLFILE AUTOBACKUP option, 7-12, 8-24
DB_UNIQUE_NAME option, 5-35
ENCRYPTION option, 6-9
EXCLUDE option, 6-6
FOR DB_UNIQUE_NAME option, 3-8
MAXPIECESIZE option, 6-4
MAXSETSIZE option, 6-4
RETENTION POLICY clause, 7-17
RETENTION POLICY option, 5-21
configuring media managers, 5-10
installing, 5-9
prerequisites, 5-9
configuring Recovery Manager
autobackups, 5-7, 7-12
backup optimization, 5-23
backup retention policies, 5-21
backup set size, 6-4
default backup type, 5-4
default devices, 5-3
overview, 5-1
shared server, 6-11
snapshot control file location, 6-11
specific channels, 6-2
tablespace exclusion for backups, 6-6
consistent backups, 7-1
using RMAN, 7-1
whole database, 27-3
control file autobackups, 11-6
after structural changes to database, 7-12
configuring, 5-7, 7-12
default format, 7-12
format, 5-7
control files
backups, 27-2, 27-10
binary, 27-10
including within database backup, 8-9
recovery using, 19-4
using RMAN, 8-8
circular reuse records, 11-3
configuring location, 5-19
creating after loss of all copies, 29-7
duplicate database, 23-17
finding filenames, 27-2
multiplexed, 5-14, 5-19, 11-6, 17-3, 27-2, 28-3, 29-1
loss of, 29-1
multiplexing, 11-5
recreated, 29-6
restoring, 19-5, 29-1, 29-2
snapshot, 12-22
specifying location of, 6-11
user-managed restore after loss of all copies, 29-6
CONTROL_FILE_RECORD_KEEP_TIME
initialization parameter, 11-4, 11-5, 12-24
CONTROL_FILES initialization parameter, 5-19, 19-5, 20-19, 29-2
CONVERT command
with tablespaces and datafiles, 25-1
COPIES option
BACKUP command, 9-8
corrupt blocks, 13-1, 15-2, 28-18
recovering, 18-2
RMAN and, 9-12
CREATE CATALOG command, 12-6, 12-11
CREATE DATAFILE clause, ALTER DATABASE statement, 29-9
CREATE SCRIPT command, 12-16
CREATE TABLE statement
AS SELECT clause, 29-9
CREATE TABLESPACE statement, 29-5
CROSSCHECK command, 11-12
crosschecking, RMAN, 2-10, 11-2, 11-12
definition, 11-12
recovery catalog with the media manager, 11-12
cross-platform transportable tablespace, 25-1
cumulative incremental backups, 2-6, 7-13, 7-14

D

data blocks, corrupted, 1-3, 1-4, 2-11, 2-16, 13-1, 14-8, 14-15, 15-4, 18-1, 28-18, 28-19
data dictionary views, 27-4, 27-5, 27-9

Data Guard environment, 3-9
archived log deletion policies, 5-27
changing a DB_UNIQUE_NAME, 12-25
configuring RMAN, 5-35
maintenance commands, 11-2
reporting in a, 10-2
RMAN backups, 8-2
RMAN backups, accessibility of, 3-8
RMAN backups, association of, 3-8
RMAN backups, interchangeability of, 3-8, 8-9
RMAN usage, 3-7
data integrity checks, 1-7, 14-3, 14-8
data preservation, definition of, 1-3
data protection
definition, 1-2
Data Recovery Advisor, 2-11, 10-4, 13-2
automated repairs, 1-8
data integrity checks, 14-3, 14-8
failure consolidation, 14-4
failure priority, 14-4
failures, 14-2, 14-3
feasibility checks, 14-4
overview, 1-7
purpose, 14-1
repair options, 14-10
repairing failures, 14-12
repairs, 14-2, 14-4, 14-5
supported configurations, 14-5
user interfaces, 14-2
data repair
overview, 13-1
techniques, 13-2
data transfer, RMAN, 1-3
database connections
Recovery Manager
auxiliary database, 4-10
hiding passwords, 4-11
without a catalog, 4-7
SYSDBA required for RMAN, 4-8
types in RMAN, 4-7
database point-in-time recovery, 16-14
definition, 16-2
Flashback Database and, 5-28, 16-1
prerequisites, 16-15
user-managed, 28-13
databases
listing for backups, 27-1
media recovery procedures, user-managed, 28-1
media recovery scenarios, 29-1
recovery
after control file damage, 29-1, 29-2
registering in recovery catalog, 12-8
reporting on schemas, 10-14
suspending, 27-12
unregistering from recovery catalog, 12-26
datafiles
backing up, 8-8, 8-28, 8-29, 27-4
determining status, 27-2
duplicate database, 23-18
listing, 27-1
losing, 28-2
recovery
without backup, 29-8
re-creating, 29-8
renaming
after recovery, 29-5
restoring, 13-3
DB_BLOCK_CHECKSUM initialization parameter, 15-2
DB_CREATE_FILE_DEST initialization parameter, 5-19, 8-21, 17-11
DB_FILE_NAME_CONVERT initialization parameter, 20-19
using with RMAN DUPLICATE command, 23-18, 23-19
DB_FLASHBACK_RETENTION_TARGET initialization parameter, 5-16, 5-17, 5-18
DB_LOST_WRITE_PROTECT initialization parameter, 6-13
DB_NAME initialization parameter, 20-18
DB_RECOVERY_FILE_DEST initialization parameter, 2-2, 5-16, 5-19
DB_RECOVERY_FILE_DEST_SIZE initialization parameter, 2-2, 5-16
DB_UNIQUE_NAME initialization parameter, 3-7, 3-8, 5-35, 10-3
DBA_DATA_FILES view, 27-4, 27-5, 27-9
DBID
determining, 17-5
problems registering copied database, 12-2
setting with DBNEWID, 12-7
DBMS_PIPE package, 4-12
DBNEWID utility, 12-7, 23-1
DBPITR. *See* database point-in-time recovery
DBVERIFY utility, 27-17
DELETE command, 11-16, 11-19, 11-22
EXPIRED option, 11-12, 11-23
OBSOLETE option, 7-19, 11-23
deleting backups, 2-10, 11-19, 11-20, 11-22
deletion policies, archived redo log, 5-26
enabling, 5-27
devices, configuring default, 5-3
differential incremental backups, 2-6, 7-13
direct ancestral path, 13-6, 16-12, 16-19
disaster recovery, 1-3
definition, 1-2
disconnecting
from Recovery Manager, 2-3
disk API, 5-10
disk failures, 1-2
disk usage
monitoring, 11-7
DROP DATABASE command, 11-24
dropped tables, retrieving, 16-7
dropping a database, 11-24
dropping the recovery catalog, 12-33
dual mode backup encryption, 6-8
dual-mode backup encryption, 9-11
dummy API, 5-10
duplexing backup sets, 6-5, 7-10, 9-6

DUPLICATE command, 23-1
duplicate databases, 3-2
  active database duplication, 4-11
  creating
    on local host, 23-14
    on remote host with different file system, 23-13
    on remote host with same file system, 23-12
    using CONFIGURE AUXNAME, 23-23
  datafiles, 23-18
  generating control files, 23-17
  generating filenames, 23-6
  how RMAN creates, 23-2
  overview, 23-1
  preparing for duplication, 23-7
  restoring archival backups, 23-21
  skipping offline normal tablespaces, 23-20
  skipping read-only tablespaces, 23-20
duplicating databases
  active database duplication, 23-2
  backup-based duplication, 23-2
DURATION parameter, BACKUP command, 9-13

E

encrypted backups, 9-10, 13-3
  decrypting, 17-10
environment variables
  NLS_DATE_FORMAT, 4-2
  NLS_LANG, 4-2
error codes
  media manager, 22-3
  RMAN, 22-1, 22-2
error messages, RMAN
  interpreting, 22-5
error stacks, RMAN
  interpreting, 22-5
EXECUTE SCRIPT command, 12-17
EXIT command, 2-3
exiting RMAN, 2-3
expired backups, 7-17, 11-13
  deleting, 11-23
EXPIRED option
  DELETE command, 11-23

F

failover, when restoring files, 13-4
failures
  definition, 1-2
  media, 1-2
  See also recovery
failures, Data Recovery Advisor, 14-2, 14-3
  consolidation, 14-4
  priority, 14-4
feasibility checks, Data Recovery Advisor, 14-4
features, new, 0-xxiii
file sections, 7-5, 7-6, 9-2, 15-5
filenames, listing for backup, 27-1
flash recovery area, 3-2, 3-7, 16-3
  autobackups, 5-8
  changing locations, 11-9
  configuring, 5-13
  definition, 2-2
  disabling, 11-9
  effect of retention policy, 7-20
  enabling, 5-16
  flashback database window, 5-29
  maintaining, 11-6
  monitoring disk usage, 11-7
  monitoring usage, 11-7
  Oracle Managed Files, 5-15
  permanent and impermanent files, 5-14
  RMAN files in, 5-20
  setting location, 5-17
  setting size, 5-17
  snapshot control files, 6-11
  space management, 5-15
flashback data archive
  definition, 1-6
Flashback Database, 2-13, 13-2
  configuring, 5-28
  determining the flashback database window, 16-12
  enabling, 5-33
  flashback logs, 1-7, 5-31
  monitoring, 11-10
  overview, 1-7
  prerequisites, 16-11
  purpose, 16-1
  requirements, 5-32
  space management, 11-7
    estimating disk space requirement, 5-19
  tuning performance, 5-34
FLASHBACK DATABASE command, 5-28, 16-12
flashback database window, 5-29
Flashback Drop, 16-3, 16-7
flashback logs, 1-7, 2-13, 5-28, 11-7, 16-3
  guaranteed restore points and, 5-30
flashback retention target, 5-28
Flashback Table, 16-3
  using, 16-4, 16-5
FLASHBACK TABLE statement, 16-4, 16-5
Flashback Technology, 16-2
  logical features, 16-3
  overview, 1-5
flashback undrop
  restoring objects, 16-8
formats, for RMAN backups, 8-3
fractured blocks, 7-2
  detection, 7-2
full backups, 7-13
  incremental backups and, 2-6

G

generic channels
  definition, 5-4
GRANT command, 12-11
groups, redo log, 29-11, 29-12

guaranteed restore points, 1-7, 5-17, 5-26
alternative to storage snapshots, 5-30
compared to storage snapshots, 5-30
creating, 5-33
flashback logs and, 5-30
requirements, 5-32
space usage in flash recovery area, 10-10

H

Health Monitor, 14-3
hot backup mode
failed backups, 27-7, 27-8
for online user-managed backups, 27-6

I

image copies, 2-4, 7-1, 7-8
definition, 7-8
testing restore of, 17-8
IMPORT CATALOG command, 12-31
INCARNATION option
LIST command, 10-9, 12-28
RESET DATABASE command, 12-28
incarnations, database, 10-9, 13-5, 16-12, 16-18
INCLUDE CURRENT CONTROLFILE option
BACKUP command, 8-9
incomplete media recovery, 28-13
incomplete recovery
defined, 16-14
in Oracle Real Application Clusters configuration, 28-5
overview, 13-5
time-based, 28-16
with backup control file, 28-5
inconsistent backups, 7-2
using RMAN, 2-4, 7-1
incremental backups, 2-5, 8-14
block change tracking, 8-20
differential, 7-13
how RMAN applies, 13-5
making, 8-14
multilevel, 7-13
using RMAN, 9-7, 9-8
initialization parameter file, 13-5
initialization parameters
CONTROL_FILES, 19-5, 29-2
DB_FILE_NAME_CONVERT, 20-19
DB_NAME, 20-18
LARGE_POOL_SIZE, 21-14
LOCK_NAME_SPACE, 20-18
LOG_ARCHIVE_DEST_*n*, 28-5
LOG_ARCHIVE_FORMAT, 28-5
LOG_FILE_NAME_CONVERT, 20-19
instance failures
backup mode and, 27-7
integrity checks, 15-1
interpreting RMAN error stacks, 22-5
interrupting media recovery, 28-6
I/O errors
effect on backups, 9-12

J

jobs, RMAN
monitoring progress, 21-10
querying details about, 10-15

K

KEEP option
BACKUP command, 11-15

L

level 0 incremental backups, 2-6, 7-13, 7-15
level 1 incremental backups, 7-13, 7-14
LIST command, 2-8, 10-2, 10-3
FAILURE option, 14-6
INCARNATION option, 12-28
LOCK_NAME_SPACE initialization parameter, 20-18
log sequence numbers, 28-3
LOG_ARCHIVE_DEST_*n* initialization parameter, 5-20, 5-21, 5-26, 17-8, 28-4, 28-5, 28-9, 28-14, 28-16
LOG_ARCHIVE_FORMAT initialization parameter, 28-5
LOG_FILE_NAME_CONVERT initialization parameter, 20-19
logical backups, 1-2
logical block corruption, 15-2
LOGSOURCE variable
SET statement, 28-6, 28-10, 28-12
long waits, 21-13
loss of
inactive log group, 29-13
lost writes, detecting, 6-13

M

maintenance commands, RMAN, 2-10, 3-4, 11-2
Data Guard environment, 11-2
managing RMAN metadata, 10-1, 11-1
MAXPIECESIZE parameter
SET command, 5-13
MAXSETSIZE parameter
BACKUP command, 6-4, 9-1
CONFIGURE command, 6-4
media failures, 1-2
archived redo log file loss, 28-13
complete recovery, 28-7
complete recovery, user-managed, 28-7
control file loss, 29-6
datafile loss, 28-2
definition, 1-2
NOARCHIVELOG mode, 28-16
online redo log group loss, 29-12
online redo log loss, 29-11
online redo log member loss, 29-11
recovery, 28-7

recovery procedures
examples, 28-2
Media Management Layer (MML) API, 3-6, 6-5
media managers, 3-2, 3-4, 3-6
backing up files, 3-6
backup piece names, 5-12
Backup Solutions Program, 3-7
catalog, 3-2
configuring for use with RMAN, 5-10
crosschecking, 11-12
definition, 2-2
error codes, 22-3
file restrictions, 5-12
installing, 5-9
library location, 5-9
linking
testing, 5-10
linking to software, 3-6, 5-9
multiplexing backups, 7-7
prerequisites for configuring, 5-9
sbttest program, 22-11
testing, 5-10
testing backups, 5-11
testing the API, 22-11
third-party, 5-8
troubleshooting, 5-11
media recovery, 7-16
ADD DATAFILE operation, 29-4
after control file damage, 29-1, 29-2
applying archived redo logs, 28-3
cancel-based, 28-13, 28-16
complete, 28-7
closed database, 28-8
complete, user-managed, 28-7
corruption
allowing to occur, 28-21
datafiles
without backup, 29-8
errors, 28-19
incomplete, 28-13
interrupting, 28-6
lost files
lost archived redo log files, 28-13
lost datafiles, 28-2
lost mirrored control files, 29-1
NOARCHIVELOG mode, 28-16
offline tablespaces in open database, 28-11
online redo log files, 29-10
parallel, 28-7
problems, 28-18, 28-19, 28-20
restarting, 28-6
restoring
whole database backups, 28-16
resuming after interruption, 28-6
roll forward phase, 28-3
scenarios, 29-1
time-based, 28-13
transportable tablespaces, 29-10
trial, 28-23
troubleshooting, 28-18, 28-19
undamaged tablespaces online, 28-11
user-managed, 28-1
using Recovery Manager, 13-5
metadata, RMAN, 3-5, 10-1, 11-1, 12-1
mirrored files
backups using, 9-8
online redo log
loss of, 29-11
splitting, 27-12
suspend/resume mode, 27-12
using RMAN, 9-8
monitoring flash recovery area usage, 11-7
monitoring RMAN, 22-7
MTTR, 14-1
multiplexed backup sets, 6-4, 7-6, 8-5, 21-4
multiplexed control files, 5-14, 5-19, 11-5, 11-6, 17-3, 27-2, 28-3, 29-1
multisection backups, 3-5, 7-3, 7-5, 7-6, 9-2, 15-5

N

naming backup sets, 7-5
new features, xxiii
NLS_DATE_FORMAT environment variable, 4-2
NLS_LANG environment variable, 4-2
NOARCHIVELOG mode
backing up, 8-10
disadvantages, 28-16
recovery, 28-16

O

obsolete backups, 7-17
definition, 7-17
deleting, 2-11, 7-19, 11-23
offsite backups, 17-6
online redo logs, 29-13
active group, 29-11, 29-12
applying during media recovery, 28-3
archived group, 29-11, 29-12
clearing
failure, 29-14
clearing inactive logs
archived, 29-13
unarchived, 29-13
configuring location, 5-19
current group, 29-11, 29-12
determining active logs, 29-12
inactive group, 29-11, 29-12
loss of, 29-13
active group, 29-14, 29-15
all members, 29-12
group, 29-12
mirrored members, 29-11
recovery, 29-10
loss of group, 29-14, 29-15
multiple group loss, 29-15
replacing damaged member, 29-11
status of members, 29-11, 29-12
OPEN RESETLOGS clause

ALTER DATABASE statement, 12-28, 13-5, 16-12, 16-17
ORA-01578 error message, 29-9
Oracle Backup Solutions Program (BSP), 3-7
Oracle Data Pump, 1-2, 16-14
Oracle Encryption Wallet
and backups, 6-8
Oracle Flashback Database. *See* Flashback Database
Oracle Flashback Drop, 1-6
Oracle Flashback Query, 1-5
Oracle Flashback Table, 1-6
Oracle Flashback Transaction, 1-6
Oracle Flashback Transaction Query, 1-6
Oracle Flashback Version Query, 1-6
Oracle Managed Files
flash recovery, 5-15
Oracle Real Application Clusters (Oracle RAC)
RMAN channels and, 6-2
Oracle Secure Backup, 3-6, 5-8
Oracle VSS writer, 5-15
Oracle wallet, 6-8
orphaned backups, 13-7

P

packages
DBMS_PIPE, 4-12
password backup encryption, 6-8
password-mode encryption, 9-11
passwords
connecting to RMAN, 4-11
performance tuning
short waits
definition of, 21-13
performance tuning, RMAN
backup performance, 21-13
LARGE_POOL_SIZE initialization parameter, 21-14
long waits, 21-13
physical backups, 1-2
physical block corruption, 15-2
pipe interface, RMAN, 4-12
point of recoverability
recovery window, 7-17
point-in-time recovery, 28-13
performing
with current control file, 16-15
tablespace, 16-2
PREVIEW option, RESTORE command, 10-2
previewing backups, 17-5
PRINT SCRIPT command, 12-19
proxy copies, 3-6, 7-8
PROXY option
BACKUP command, 7-8

Q

QUIT command, 2-3
quitting RMAN, 2-3

R

raw devices
backing up to, 27-14
UNIX backups, 27-14
Windows backups, 27-16
RC_ARCHIVED_LOG view, 10-18
RC_BACKUP_FILES view, 10-19
RC_BACKUP_PIECE view, 10-17
RC_BACKUP_SET view, 11-19
read-only tablespaces
backups, 27-9
RECOVER clause
ALTER DATABASE statement, 28-6, 28-10, 28-12
RECOVER command, 13-5
COPY option, 8-17
PARALLEL and NOPARALLEL options, 28-7
TEST option, 15-7
unrecoverable objects and standby databases, 29-9
UNTIL TIME option, 28-16
USING BACKUP CONTROLFILE clause, 29-5
recovery
ADD DATAFILE operation, 29-4
automatically applying archived logs, 28-4
cancel-based, 28-16
complete, 17-1, 28-7
closed database, 28-8
offline tablespaces, 28-11
corruption
intentionally allowing, 28-21
database
in NOARCHIVELOG mode, 19-1
database files
how RMAN applies changes, 13-5
overview, 13-5
database point-in-time, 16-14
datafiles, 28-2
disaster using RMAN, 19-8
dropped table, 29-16
errors, 28-19
failures requiring, 1-2
interrupting, 28-6
media, 28-1, 28-17, 29-1
multiple redo threads, 28-5
of lost or damaged recovery catalog, 12-15
online redo logs, 29-10
losing member, 29-11
loss of group, 29-12
parallel, 28-7
preparing for, 17-3
problems, 28-18
fixing, 28-20
investigating, 28-19
stuck, 28-18
time-based, 28-16
transportable tablespaces, 29-10
trial, 28-23
explanation, 28-23
overview, 28-23
troubleshooting, 28-18

user errors, 29-15
user-managed, 28-1, 28-17, 29-1
using backup control file, 19-4
without recovery catalog, 19-6
using logs in a nondefault location, 28-5
using logs in default location, 28-5
using logs in nondefault location, 28-6
without a recovery catalog, 11-5
recovery catalog, 3-5, 12-1
backing up, 12-13
cataloging backups, 11-16, 12-9
centralization of metadata, 12-2
creating, 12-4
crosschecking, 11-12
DBID problems, 12-2
definition, 2-2, 3-1
deleting backups, 11-19
deleting records, 11-22
dropping, 12-33
log switch record, 11-16
managing size of, 12-24
operating with, 3-5
purpose of, 12-1
recovery of, 12-15
refreshing, 12-22
registering databases, 12-2, 12-7, 12-8
resynchronizing, 12-22
space requirements, 12-5
stored scripts, 12-15
creating, 12-16
synchronization, 12-22
unregistering databases, 12-26
updating
after operating system deletions, 11-19
upgrading, 12-29
views, querying, 10-17
virtual private catalogs, 3-5
recovery catalogs
backing up, 12-13
base recovery catalog, 12-9
dropping, 12-33
importing, 12-31
moving, 12-33
virtual private catalogs, 12-9
Recovery Manager
allocating tape buffers, 21-6
archived redo logs
backups, 8-11
authentication, 2-3
backups, 8-1
archived redo logs, 8-11
backing up, 7-11, 8-26
batch deletion of obsolete, 7-19
control files, 8-8
datafile, 8-8, 8-28, 8-29
duplexed, 7-10
image copy, 7-8
incremental, 8-14, 9-7, 9-8
optimization, 5-23, 8-13
tablespace, 8-28, 8-29
testing, 15-3, 15-4, 15-6
validating, 15-4, 15-6
whole database, 8-7
channels, 3-3
naming conventions, 3-5
client, 2-1
connecting to databases, 2-2
corrupt datafile blocks
handling I/O errors and, 9-12
crosschecking recovery catalog, 11-12
database character set, 4-2
database connections, 4-7
auxiliary database, 4-10
duplicate database, 4-11
hiding passwords, 4-11
SYSDBA required for target, 4-8
without a catalog, 4-7
DBMS_PIPE package, 4-12
definition, 2-1
disconnecting from, 2-3
duplicate databases, overview of, 23-2
error codes
message numbers, 22-2
errors, 22-1, 22-2
interpreting, 22-5
file deletion, 11-20
fractured block detection in, 7-2
image copy backups, 7-8
incremental backups
cumulative, 7-14
differential, 7-13
level 0, 7-13
integrity checking, 15-1
jobs, monitoring progress, 21-10
jobs, querying details of, 10-15
lists, 10-3
maintenance commands, 2-10
media management
backing up files, 3-6
Backup Solutions Program (BSP), 3-7
crosschecking, 11-12
media manager, linking with a, 5-9
metadata, 3-5, 10-1, 11-1, 12-1
monitoring, 22-7
overview, 2-1, 3-3
performance
monitoring, 22-7
pipe interface, 4-12
proxy copy, 3-6
recovery
after total media failure, 19-8
recovery catalog, 12-1
backing up, 12-13
crosschecking, 11-12
managing the size of, 12-24
operating with, 3-5
recovering, 12-15
registration of target databases, 12-2, 12-7, 12-8
resynchronizing, 12-22

synchronization, 12-22
upgrading, 12-29
reports, 10-11
database schema, 10-14
objects needing a backup, 10-11
obsolete backups, 10-13
repository, 3-5
restoring
archived redo logs, 17-8
datafiles, 13-3
retention policies
configuring, 5-21
return codes, 22-7
setting time parameters, 4-2
snapshot control file location, 6-11
starting, 2-2
synchronous and asynchronous I/O, 21-5, 21-7
terminating commands, 22-13
test disk API, 5-10
types of backups, 7-8
using RMAN commands, 3-3
recovery window, 5-21
point of recoverability, 7-17
RECOVERY WINDOW parameter
CONFIGURE command, 5-22
recovery windows
configuring for retention policy, 5-22
definition, 7-17
RECOVERY_CATALOG_OWNER role, 12-11
recycle bin, 16-3, 16-8
restoring objects from, 16-8
redo logs
incompatible format, 28-18
naming, 28-5
parallel redo, 28-18
redo records
problems when applying, 28-18
REGISTER command, 12-8
REPAIR FAILURE command, 14-12, 14-14
repair options, Data Recovery Advisor, 14-10
repairs, Data Recovery Advisor, 14-2
consolidation of, 14-5
manual and automatic, 14-4
REPLACE SCRIPT command, 12-17
REPORT command, 2-9, 10-2, 10-11
NEED BACKUP option, 10-11
OBSOLETE option, 7-19
reports, RMAN, 2-8, 10-2, 10-11
backup jobs, 10-15
database schema, 10-14
files needing backups, 10-11
obsolete backups, 10-13
unrecoverable backups, 10-13
repository, RMAN, 3-5
RESET DATABASE command
INCARNATION option, 12-28
RESETLOGS operation
when necessary, 13-6
RESETLOGS option
of ALTER DATABASE, 28-17
restartable backups, 9-12
RESTORE command, 13-3, 17-3
FORCE option, 13-4
PREVIEW option, 10-2, 17-5
VALIDATE HEADER option, 10-2, 17-5
restore optimization, 13-4
restore points, 1-7, 2-13
creating, 5-33
dropping, 11-9
flashing back to, 16-19
guaranteed, 1-7, 5-30
compared to storage snapshots, 5-30
listing, 10-9
requirements, 5-32
restore validation, 17-8
restoring
control files, 19-5
to default location, 29-1
to nondefault location, 29-2
database
to default location, 28-16
database files, 13-3, 13-4
server parameter files, 19-2
testing, 15-6, 17-8
user-managed backups, 28-2
RESUME clause
ALTER SYSTEM statement, 27-13
resuming recovery after interruption, 28-6
RESYNC CATALOG command, 12-22, 12-24
FROM CONTROLFILECOPY option, 12-15
resynchronizing the recovery catalog, 3-8, 12-22, 12-24
retention policies. *See* backup retention policies
return codes
RMAN, 22-7
REVOKE command, 12-12
RMAN repository, 1-3, 2-1
RMAN. *See* Recovery Manager
RMAN sessions, 2-12, 3-4

S

SBT, 3-4, 5-13
sbtio.log
and RMAN, 22-2
sbttest program, 22-11
scenarios, Recovery Manager
NOARCHIVELOG backups, 8-10
recovering pre-resetlogs backup, 16-20, 19-1
recovery after total media failure, 19-8
scripts, RMAN, 2-8
substitution variables in, 8-24
server parameter files
autobackups, 7-12
backups, 8-10
configuring autobackups, 5-7, 7-12
restoring, 19-2
server sessions, Recovery Manager, 3-3
session architecture, Recovery Manager, 3-3
SET command

DBID option, 3-8
ENCRYPTION option, 6-8
MAXCORRUPT option, 15-3
SET statement
AUTORECOVERY option, 28-4
LOGSOURCE variable, 28-6, 28-10, 28-12
shadow copies, 8-16
shared server
configuring for use with RMAN, 6-11
configuring RMAN, 6-11
short waits
definition, 21-13
SHOW command, 2-3, 5-2
SHUTDOWN statement
ABORT option, 28-16, 29-1, 29-2
size of backup sets, setting, 7-5
skipping files in RMAN backups, 9-6
snapshot control files, 6-11, 12-22
specifying location, 6-11
split mirrors
suspend/resume mode, 27-12
using as backups, 9-8
SPOOL command, 14-13
standby databases, 3-2
creating with DUPLICATE, 23-2
statements, SQL
ALTER DATABASE, 28-6, 28-10, 28-12
storage snapshots, 5-30
stored scripts, 3-5, 8-24, 12-3, 12-15, 12-31
creating RMAN, 12-16
deleting, 12-20, 12-21
dynamic, 12-18
executing, 12-21
listing names of, 12-20
managing, 12-15
printing, 12-19
substitution variables in, 12-18
stuck recovery, 28-18
substitution variables, FORMAT parameter, 5-12, 7-5, 7-8
substitution variables, stored scripts, 12-18
SUSPEND clause
ALTER SYSTEM statement, 27-13
suspending a database, 27-12
suspend/resume mode, 27-12
SWITCH command, 17-16
SYSDBA privileges, 2-2
system backup to tape. *See* SBT
system time
changing
effect on recovery, 28-16

T

tables, recovery of dropped, 29-16
tablespace point-in-time recovery, 16-2
configuring datafile names, 6-10
planning, 20-5
preparing the auxiliary instance, 20-18
restrictions, 20-4
why perform, 20-1
tablespaces
backups, 8-28, 8-29, 27-6
offline, 27-4
online, 27-6
backups using RMAN, 8-8
excluding from backups, 6-6
excluding from RMAN backups, 6-6
read-only
backing up, 27-9
read/write
backing up, 27-5
recovering accessible
when database is open, 17-13
recovering offline in open database, 28-11
transporting with RMAN, 24-1
tape devices, 3-6
target database
connecting to, 2-2
definition, 2-1, 3-1
terminating RMAN commands, 22-13
test disk API, 5-10
testing RMAN
backups, 15-3, 15-4, 15-6
with media management API, 22-11
time format
RECOVER DATABASE UNTIL TIME statement, 28-16
time parameters
setting for Recovery Manager use, 4-2
time-based recovery, 28-16
trace files, RMAN, 22-2
transparent backup encryption, 6-8
transparent-mode backup encryption, 9-11
transportable tablespaces
creating with RMAN, 24-1
and Data Pump Export, 24-9
and past points in time, 24-8
auxiliary destination, 24-3
auxiliary instance parameter file, 24-5, 24-6
file locations, 24-10
initialization parameters, 20-16, 24-5
Shared Pool Size, 24-5
cross-platform, 25-1
recovery, 29-10
transporting tablespaces, 24-1
trial recovery, 15-7, 28-23
tuning Recovery Manager
V$ views, 22-7

U

UNAVAILABLE option
of CHANGE, 11-14
UNCATALOG option
CHANGE command, 11-18
deleting repository records, 11-18
undo optimization, backup, 5-23, 7-4
unrecoverable objects
recovery, 29-9

UNREGISTER DATABASE command, 12-26
unregistering databases, 12-26
UNTIL TIME option
  RECOVER command, 28-16
unused block compression, 7-4
upgrading the recovery catalog, 12-29
user errors
  definition, 1-2
  recovery from, 29-15
user-managed backups, 27-1, 27-3
  backup mode, 27-5, 27-7
  control files, 27-10
  definition, 7-9
  determining datafile status, 27-2
  hot backups, 7-2, 27-8
  listing files before, 27-1
  offline tablespaces, 27-4
  read-only tablespaces, 27-9
  tablespace, 27-6
  verifying, 27-17
  whole database, 27-3
user-managed recovery, 28-13
  ADD DATAFILE operation, 29-4
  complete, 28-7
  incomplete, 28-13
  interrupting, 28-6
  scenarios, 29-1
user-managed restore operations, 28-2

V

V$ARCHIVED_LOG view, 5-18, 10-18, 16-15
  listing all archived logs, 27-11
V$BACKUP view, 27-2
V$BACKUP_ASYNC_IO view, 21-12
V$BACKUP_DATAFILE view, 8-15, 11-17
V$BACKUP_FILES view, 5-22, 11-12, 11-17
V$BACKUP_PIECE view, 10-17, 11-17
V$BACKUP_REDOLOG view, 11-17
V$BACKUP_SET view, 11-17, 11-19
V$BACKUP_SPFILE view, 11-17
V$BACKUP_SYNC_IO view, 21-12
V$BLOCK_CHANGE_TRACKING view, 8-22
V$CONTROLFILE view, 8-9
V$DATABASE view, 10-19, 16-5, 16-17
V$DATABASE_BLOCK_CORRUPTION view, xxv, 1-5, 2-16, 15-3, 15-4, 18-1, 18-2, 18-4, 18-5
V$DATABASE_INCARNATION view, 12-28
V$DATAFILE view, 17-5, 20-19, 27-1
  listing files for backups, 27-1
V$DATAFILE_HEADER view, 10-2, 17-4
V$DIAG_INFO view, 2-16, 18-4
V$EVENT_NAME view, 22-8
V$FLASH_RECOVERY_AREA_USAGE view, 11-7
V$FLASHBACK_DATABASE_LOG view, 5-19, 16-12, 16-17
V$FLASHBACK_DATABASE_STAT view, 11-10
V$INSTANCE view, 17-4
V$LOG_HISTORY view
  listing all archived logs, 28-8
V$LOGFILE view, 20-19, 29-11, 29-12
V$PARAMETER view, 16-5
V$PROCESS view, 10-2, 22-7, 22-9
V$PROXY_ARCHIVEDLOG view, 7-8
V$PROXY_DATAFILE view, 7-8
V$RECOVER_FILE view, 17-5, 28-8
V$RECOVERY_FILE_DEST, 11-7
V$RECOVERY_FILE_DEST view, 11-7
V$RECOVERY_LOG view
  listing logs needed for recovery, 28-8
V$RESTORE_POINT view, 10-10, 16-5
V$RMAN_BACKUP_JOB_DETAILS view, 10-15
V$RMAN_BACKUP_SUBJOB_DETAILS view, 10-15
V$RMAN_ENCRYPTION_ALGORITHMS view, 6-7, 6-10, 21-6
V$RMAN_OUTPUT view, 10-19
V$RMAN_STATUS view, 22-1
V$SESSION view, 6-12, 10-2, 22-7, 22-9
V$SESSION_LONGOPS view, 21-10
V$SESSION_WAIT view, 22-7, 22-8
V$SGASTAT view, 21-14
V$SYSSTAT view, 11-10
V$TABLESPACE view, 17-5, 27-1
VALIDATE command, 14-3, 14-8, 15-4, 17-3
  SECTION SIZE parameter, 15-5
VALIDATE HEADER option, RESTORE command, 10-2
validation, RMAN, 14-8
  backups, 2-8, 15-4, 15-6
  database files, 2-7, 15-4
  restore operations, 17-8
views, recovery catalog, 10-2, 10-17
virtual private catalogs, 3-5
  creating, 12-9
  dropping, 12-12
Volume Shadow Copy Service (VSS), 5-15, 8-16

W

wallet, 6-8
whole database backups
  ARCHIVELOG mode, 27-3
  inconsistent, 27-3
  NOARCHIVELOG mode, 27-3
  preparing for, 27-3
  using RMAN, 8-7

Z

ZLIB compression algorithm, 6-6, 7-4